ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

13/04/2021

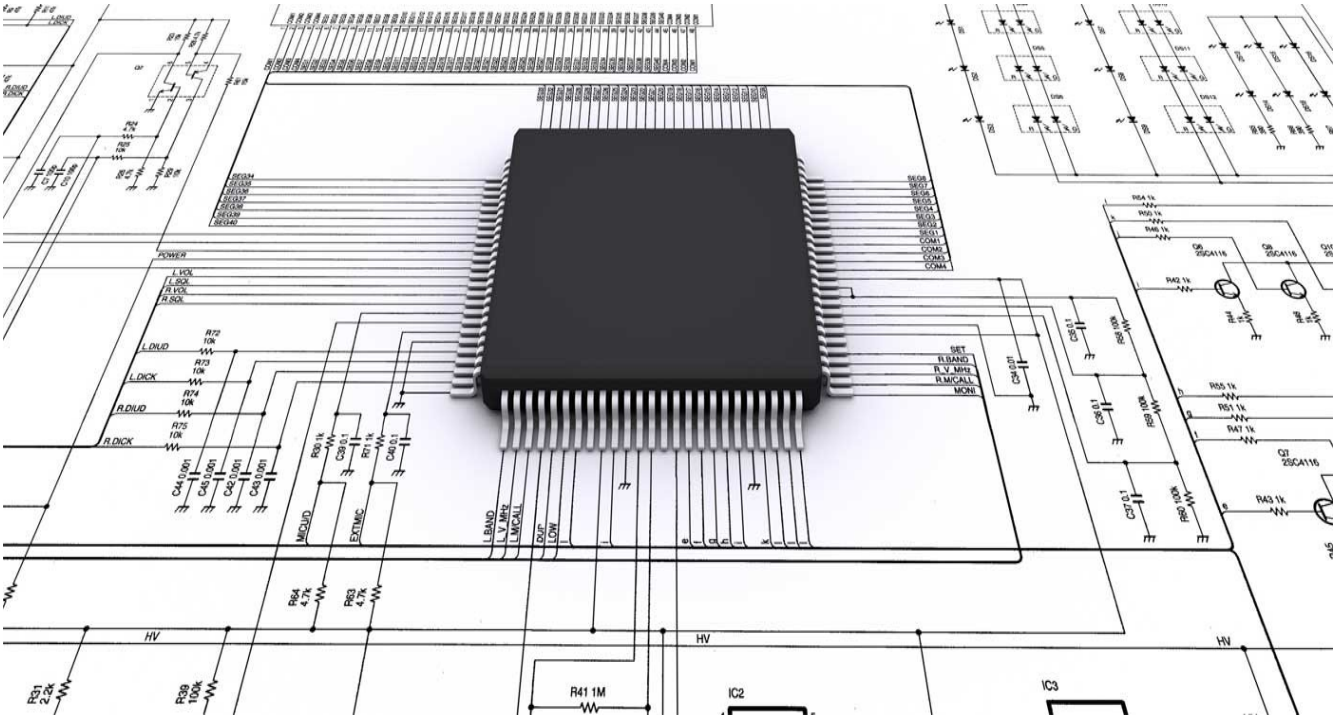
ΗΡΥ608/419-ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΩΝ CAD ΓΙΑ ΣΧΕΔΙΑΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2021

ΑΣΚΗΣΗ 4
ΙΕΡΑΡΧΙΚΗ ΣΧΕΔΙΑΣΗ – ΔΗΜΙΟΥΡΓΙΑ ΓΡΑΦΟΥ, ΕΥΡΕΣΗ BIPARTITE ΓΡΑΦΟΥ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ ΔΥΟ DATASET ΑΠΟ ΕΝΑ

ΑΝΑΦΟΡΑ

Μυλωνάκης
Εμμανουήλ
2015030079



ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σκοπός.

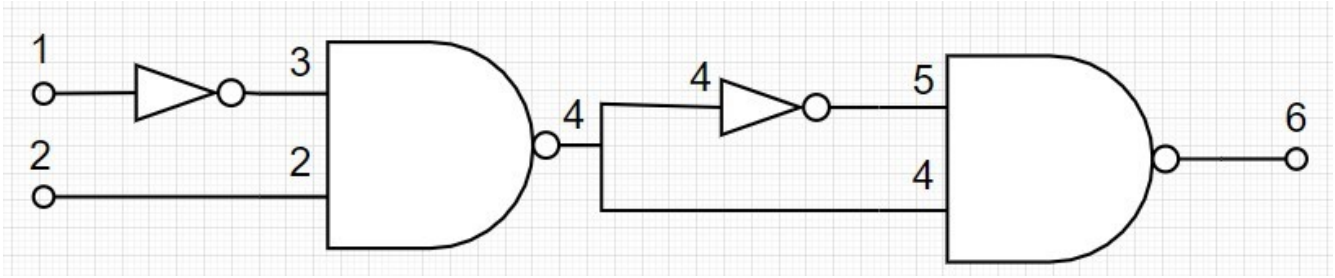
Σκοπός της τέταρτης εργαστηριακής άσκησης είναι η δημιουργία δύο επίπεδων netlist από ένα. Αξιοποιούμε το γεγονός ότι, το netlist το οποίο διαβάζουμε είναι μία μαθηματική δομή κατευθυνόμενου μη κυκλικού γράφου. Με χρήση κατάλληλων συνθηκών επί του κυκλώματος, δηλαδή του γράφου, παράγουμε δύο καινούργια κυκλώματα από το αρχικό, τα οποία είναι ικανά να προσομοιωθούν ξεχωριστά από τον απλό προσομοιωτή του εργαστηρίου δύο.

Υλοποίηση-Περιγραφή.

Αρχικά, αποθηκεύουμε από το αρχείου εισόδου χρήσιμες πληροφορίες όπως: οι κόμβοι τάσης/γείωσης, του κόμβους εισόδου/εξόδου του κυκλώματος και ολόκληρο το netlist σε ένα πίνακα με όνομα **DAG** διαστάσεων [**#CMOS]x[6]**. Σε κάθε γραμμή του πίνακα αποθηκεύετε ένα transistor του netlist, από 1η έως την 4η στήλη κρατάμε όλες τις πληροφορίες για το transistor αυτό, δηλαδή | (1η){PMOS/NMOS} | (2η) {Gate_Node} | (3η) {Source/Drain_Node} | (4η) {Drain/Source_Node} |, στην 5η την πληροφορία για το αν μπορούμε να διαχωρίσουμε τον γράφο στο αντίστοιχο transistor και στην 6η στήλη αν ο διαχωρισμός γίνεται πριν ή μετά το transistor αυτό.

Η επεξήγηση του αλγορίθμου θα γίνει σύμφωνα με το κύκλωμα στην Εικόνα 1 του οποίου το παραγώμενο netlist απ'το εργαστηρίο 3 φαίνεται στην Εικόνα 2.

Εικόνα 1.



Εικόνα 2.

```
## RAILS
VCC 11 ; GND 14
## INPUTS
12 ; 25 ;
## OUTPUTS
42 ;
## NETLIST
U1 PMOS 12 11 13
U2 NMOS 12 13 14
U3 PMOS 25 11 22
U4 PMOS 13 11 22
U5 NMOS 25 22 23
U6 NMOS 13 23 14
U7 PMOS 22 11 33
U8 NMOS 22 33 14
U9 PMOS 22 11 42
U10 PMOS 33 11 42
U11 NMOS 22 42 43
U12 NMOS 33 43 14
```

Εφόσον έχουμε αποθηκεύσει όλες τις πληροφορίες που προαναφέρθηκαν από το αρχείο, καλείται η συνάρτηση **initiate_DAG(int x)** η οποία σημειώνει όλα τα υποψήφια transistor του netlist που μπορεί να γίνει διαχωρισμός. Με βάση το συγκεκριμένο παράδειγμα, ο πίνακας DAG έχει τις παρακάτω τιμές:

DAG

P	**12**	**11**	**13**	**-1**	**-1**
N	**12**	**13**	**14**	**-1**	**-1**
P	**25**	**11**	**22**	**-1**	**-1**
P	**13**	**11**	**22**	**S**	**-1**
N	**25**	**22**	**23**	**-1**	**-1**
N	**13**	**23**	**14**	**S**	**-1**
P	**22**	**11**	**33**	**S**	**-1**
N	**22**	**33**	**14**	**S**	**-1**
P	**22**	**11**	**42**	**S**	**-1**
P	**33**	**11**	**42**	**S**	**-1**
N	**22**	**42**	**43**	**S**	**-1**
N	**33**	**43**	**14**	**-1**	**-1**

Η συνάρτηση αυτή λειτουργεί αναδρομικά. Ξεκινάει από την αρχή του netlist και τερματίζει στην προ-τελευταία γραμμή του πίνακα. Με τον δείκτη x ως όρισμα γνωρίζουμε σε κάθε κλήση της, μέχρι πιο σημείο του πίνακα έχει διασχίσει. Έτσι κατά την αναδίπλωση της αναδρομής εκτελείται μια επαναληπτική διαδικασία που ελέγχει αν ο κόμβος Gate του transistor που δείχνει ο δείκτης x, συμμετέχει σε προηγούμενη συνδεσμολογία ως κόμβος Source ή Drain. Όλα τα transistor αυτά σημειώνονται με τον χαρακτήρα **S** των οποίων οι πύλες ουσιαστικά αφορούν την συνθήκη "κοψίματος" **(D,D,...,S,S,...)-->(G)**. Όλα τα υπόλοιπα σημειώνονται με **-1**.

Στην συνέχεια, καλείται η συνάρτηση **find_correct_splitting()**. Σε αυτήν γίνεται άλλο ενα πέρασμα του πίνακα DAG και απορρίπτονται όσα **S** από την προηγούμενη συνάρτηση θα μας δημιουργήσουν πρόβλημα αν επιλεχθούν για διαχωρισμό. Επιπλέον, στα αποδεκτά **S** σημειώνεται η πληροφορία για το αν θα συμπεριλάβουμε το συγκεριμένο transistor στο πρώτο αρχείο ή όχι, με τους χαρακτήρες **A** (από After) ή **B** (από Before) αντίστοιχα. Από αυτήν την συνάρτηση έχουμε τα εξής αποτελέσματα:

DAG

U1	**P**	**12**	**11**	**13**	**-1**	**-1**
U2	**N**	**12**	**13**	**14**	**-1**	**-1**
U3	**P**	**25**	**11**	**22**	**-1**	**-1**
U4	**P**	**13**	**11**	**22**	**-1**	**-1**
U5	**N**	**25**	**22**	**23**	**-1**	**-1**
U6	**N**	**13**	**23**	**14**	**S**	**A**
U7	**P**	**22**	**11**	**33**	**S**	**B**
U8	**N**	**22**	**33**	**14**	**S**	**A**
U9	**P**	**22**	**11**	**42**	**S**	**B**
U10	**P**	**33**	**11**	**42**	**-1**	**-1**
U11	**N**	**22**	**42**	**43**	**-1**	**-1**
U12	**N**	**33**	**43**	**14**	**-1**	**-1**

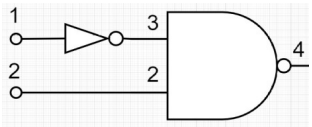
Οι τιμές UX δεν είναι αποθηκευμένες στον πίνακα, απλώς αναγράφονται σε αυτό το σημείο για την επεξήγηση.

Σε αυτήν την συνάρτηση για κάθε τρανζίστορ που σημειώθηκε με 'S' από την προηγούμενη, ελέγχουμε αν κάποιος από τους κόμβους Source ή Drain συμμετέχει σε κάποιο συνδεσμολογία ως Source ή Drain σε κάποιο σημείο του κυκλώματος πρίν ή μετά απο αυτό. Οι κόμβοι τάσης και γείωσης αγνοούνται. Κατά αυτόν τον τρόπο, απορρίπτονται τα transistor που παίζουν ρόλο σε κάποια εσωτερική συνδεσμολογία μιας πύλης. Για παράδειγμα, το τρανζίστορ U4 απορρίφθηκε, καθώς ο κόμβος του Drain 22, συμμετέχει ως Drain στο τρανζίστορ U3 (πιο πριν) και ως Drain στο τρανζίστορ U5 (πιο μετά). Για το τρανζίστορ U6, ο κόμβος του Source 14 αγνοείται καθώς είναι VCC, ενώ, ο κόμβος του Drain 23 συμμετέχει στο τρανζίστορ U5 ως Source 23 (πιο πριν) και δεν μπορούμε να κόψουμε πριν απο αυτό, αλλά, δεν συμμετέχει ως Source η Drain μετέπειτα στο κύκλωμα. Για αυτό σημειώνεται ο χαρακτήρας **A**. Με αυτήν την λογική διαμορφόνεται ο υπόλοιπος πίνακας. Τα δύο κοψίματα στο U6 και U7 είναι απολύτως εισοδύναμα, καθώς αν επιλεχθεί το U6 θα το συμπεριλάβουμε

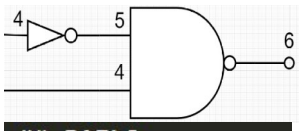
στο πρώτο αρχείο, ενώ αν επιλεχθεί το U7 δεν θα το συμπεριλάβουμε πρώτο αρχείο. Αυτό είναι και το νόημα της σημασιολογίας **A**fter και **B**efore. Ο λόγος για τον οποίο οι κόμβοι τάσης/γείωσης αγνοούνται σε αυτήν την διαδικασία, είναι επειδή πάντα θα βρεθεί ο αριθμός του κόμβου τους μετέπειτα ή προηγουμένος από το σημείο κοπής που έχει οριστεί σε κάποιο κόμβο του γράφου. Έτσι ο αλγόριθμος θα απορρίψει όλα τα σημεία κοπής. Συνοψίζοντας την παραπάνω λογική, γίνεται το πρώτο πέρασμα του γράφου και σημειώνονται οι υποψήφιοι κόμβοι κοψίματος που ικανοποιούν την αναγκαία και ικανή συνθήκη (οσαδήποτε S και D) **-->** (G και μόνο G). Στο επόμενο βήμα, **επί των κόμβων που έχουν ικανοποιήση την συνθήκη αυτή** γίνονται οι ελέγχοι που περιγράφηκαν. Οι κόμβοι αυτοί αφορούν Gates, άρα, για τα τρανζίστορ με τα συγκεκριμένα Gates ελέγχεται εάν τα Source ή Drain τους συμμετέχουν σε κάποια συνδεσμολογία πρίν ή μετά από αυτό το transistor. Με τον τρόπο που έχουμε ορίσει τις βιβλιοθήκες και παράγουμε το netlist είναι εγγυημένο ότι, αν από ένα σημείο μετά ή πριν στον γράφο δεν εμφανίζονται αυτοί οι κόμβοι ως Source η Drain, δεν πρόκειται να κόψουμε εντός κάποιας πύλης και θα βρισκόμαστε πάντα στο σημείο εισόδου ή εξόδου της. Με άλλα λόγια και όπως έχει αναφερθεί και στην θεωρία, αυτή είναι η περίπτωση της απαγορευμένης όδευσης κοψίματος **VCC --> (οσαδήποτε S και D) --> GND**, που ελέγχεται με έναν πιο έμμεσο τρόπο.

Συνεχίζοντας, καλείται η συνάρτηση **int find_splitting_point()** η οποία μας επιστρέφει το σημείο "κοπής" του κυκλώματος. Λειτουργεί πολύ απλά, διαιρεί το πλήθος των transistor του κυκλώματος διά δύο. Αν το σημείο αυτό έχει μαρκαριστεί με **S** το επιστρέφει, αλλιώς, διασχίζει το κύκλωμα από το σημείο αυτό προς στην αρχή και προς το τέλος μέχρις ώτου να βρεί κάποιο splitting point και επιστρέφει αυτό για το οποίο διένυσε την μικρότερη απόσταση. Έπειτα, όριζουμε τις καινούργιες εισόδους και εξόδους για κάθε ένα από τα δύο κύκλωματα μέσο της συνάρτησης **define_new_INs_OUTs(int splitting_point).** Αρχικά, αναζητάει με βάση το splitting point ποιες από τις εισόδους και εξόδους του αρχικού κυκλώματος συμμετέχουν στο πρώτο και ποιες στο δεύτερο κύκλωμα και αποθηκεύονται κατάλληλα. Τέλος, γίνεται μία αναζήτηση για ασύνδετα Source ή Drain στον πρώτο κύκλωμα τα οποίο όμως είναι κόμβοι Gate στο δεύτερο. Αυτές είναι οι ακμές που "κόπηκαν" λόγο του διαχωρισμού του γράφου, αναφέρονται σε ισοδύναμους κόμβους, χαρακτηρίζουν αυτό που ονομάζουμε bipartite και είναι οι κόμβοι που ορίζουν καινούργιες εξόδους για το πρώτο κύκλωμα και καινούργιες εισόδους για το δεύτερο.

Με βάση όσα περιγράφηκαν, έχουμε των παρακάτω διαχωρισμό για το κύκλωμα του παραδείγματος:

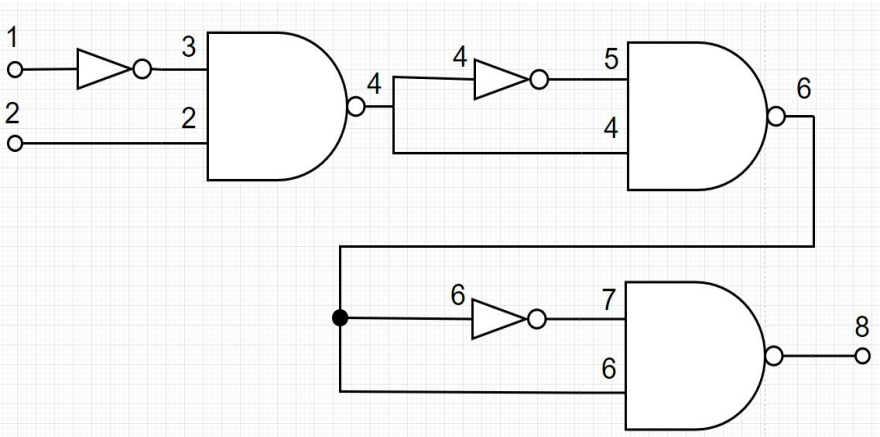



```
## RAILS
VCC 11 ; GND 14
## INPUTS
12 ; 25 ;
## OUTPUTS
22 ;
## NETLIST
U1 PMOS 12 11 13
U2 NMOS 12 13 14
U3 PMOS 25 11 22
U4 PMOS 13 11 22
U5 NMOS 25 22 23
U6 NMOS 13 23 14
```




```
## RAILS
VCC 11 ; GND 14
## INPUTS
22 ;
## OUTPUTS
42 ;
## NETLIST
U7 PMOS 22 11 33
U8 NMOS 22 33 14
U9 PMOS 22 11 42
U10 PMOS 33 11 42
U11 NMOS 22 42 43
U12 NMOS 33 43 14
```

Όσον αφορά το κόψιμο σε πολλαπλές ακμές έχουμε το παρακάτω διάγραμμα με το αντίστοιχο netlist.




```
## RAILS
VCC 11 ; GND 14
## INPUTS
12 ; 25 ;
## OUTPUTS
62 ;
## NETLIST
U1 PMOS 12 11 13
U2 NMOS 12 13 14
U3 PMOS 25 11 22
U4 PMOS 13 11 22
U5 NMOS 25 22 23
U6 NMOS 13 23 14
U7 PMOS 22 11 33
U8 NMOS 22 33 14
U9 PMOS 22 11 42
U10 PMOS 33 11 42
U11 NMOS 22 42 43
U12 NMOS 33 43 14
U13 PMOS 42 11 53
U14 NMOS 42 53 14
U15 PMOS 42 11 62
U16 PMOS 53 11 62
U17 NMOS 42 62 63
U18 NMOS 53 63 14
```

Σε αυτήν την περίπτωση επιλέγεται από τον αλγόριθμο ο κόμβος 5 μετά την 2η πύλη NOT όπου λόγο του εργαστηρίου 3 στο netlist δεξιά αναγράφεται ως 33. Έτσι στο πρώτο αρχείο θα έχουμε έως το transistor U8 και στο δεύτερο από το U9 έως τέλος. Όμως ο κόμβος 22 που αντιπροσωπεύει τον κόμβο 4 του διαγράμματος κανονικά οδηγεί στην άλλη είσοδο της 2ης NAND. Ως αποτέλεσμα για να διατηρηθεί ορθά η έννοια του bipartite ορίζονται δύο καινούργιες εξόδοι/εισόδοι για το πρώτο/δεύτερο κύκλωμα, που αναγράφονται στο netlist ως 22, 33 και αφουρόυν τους κόμβους του διαγράμματος 4,5 αντίστοιχα. Τα netlist αυτά παρουσιάζονται στις παρακάτω εικόνες.

```
## RAILS
VCC 11 ; GND 14
## INPUTS
12 ; 25 ;
## OUTPUTS
33 ; 22 ;
## NETLIST
U1 PMOS 12 11 13
U2 NMOS 12 13 14
U3 PMOS 25 11 22
U4 PMOS 13 11 22
U5 NMOS 25 22 23
U6 NMOS 13 23 14
U7 PMOS 22 11 33
U8 NMOS 22 33 14
```

```
## RAILS
VCC 11 ; GND 14
## INPUTS
33 ; 22 ;
## OUTPUTS
62 ;
## NETLIST
U9 PMOS 22 11 42
U10 PMOS 33 11 42
U11 NMOS 22 42 43
U12 NMOS 33 43 14
U13 PMOS 42 11 53
U14 NMOS 42 53 14
U15 PMOS 42 11 62
U16 PMOS 53 11 62
U17 NMOS 42 62 63
U18 NMOS 53 63 14
```

Στα παραδοτέα αρχεία περιλαμβάνεται αυτό το κύκλωμα του παραπάνω διαγράμματος με την ονομασία NOT_NAND.txt. Επιπλέον περιέχεται και το κύκλωμα του RCA 8 bit καθώς επίσης και το Lab_2.c με το εκτελέσιμο του. Έγινε μία μικρή αλλαγή σε αυτό, καθώς είχε ξεχαστεί να κληθεί η συνάρτηση fclose(FILE *fp) και για μεγάλου όγκου αρχείο δεν εκτυπωνόντουσαν ορισμές γραμμές στο τέλος του αρχείου εξόδου. Πάρθηκε η πρωτοβουλία να αναπαράγεται ένα testbench με όλα τα πιθανά διανύσματα εισόδου. Έτσι έχει γίνει πλήρης προσομοίωση των δύο RCA 4bit που παράγονται από το εργαστήριο αυτό. Με first_X.txt/second_X.txt ονομάζονται τα αρχεία για το πρώτο και δεύτερο κύκλωμα αντίστοιχα. Τα αρχεία Result_X.txt αφορούν τα αποτελέσματα του απλού μας προσομοιωτή.