ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

3/04/2021

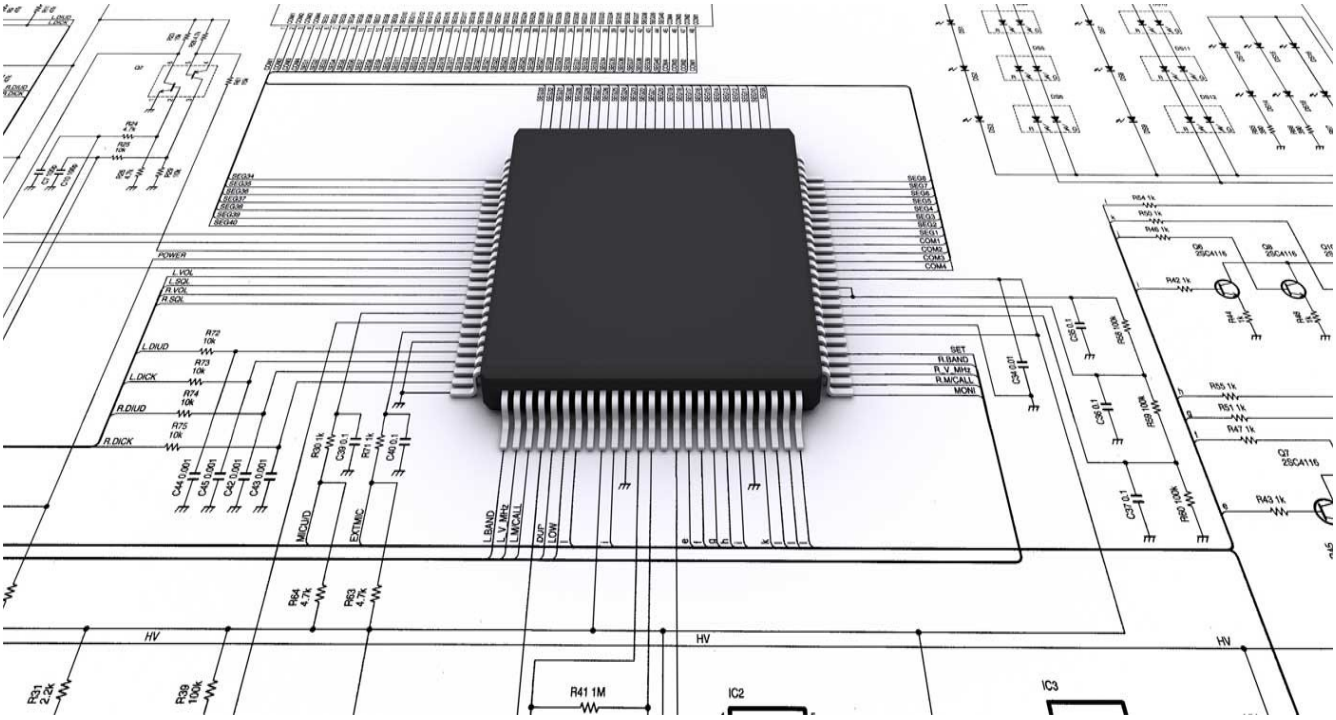
ΗΡΥ608/419-ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΩΝ CAD ΓΙΑ ΣΧΕΔΙΑΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2021

ΑΣΚΗΣΗ 3
ΙΕΡΑΡΧΙΚΗ ΣΧΕΔΙΑΣΗ – ΔΗΜΙΟΥΡΓΙΑ DATASET ΓΙΑ ΤΟΝ ΑΠΛΟΣ ΠΡΟΣΟΜΟΙΩΤΗ CMOS

ΑΝΑΦΟΡΑ

Μυλωνάκης
Εμμανουήλ
2015030079



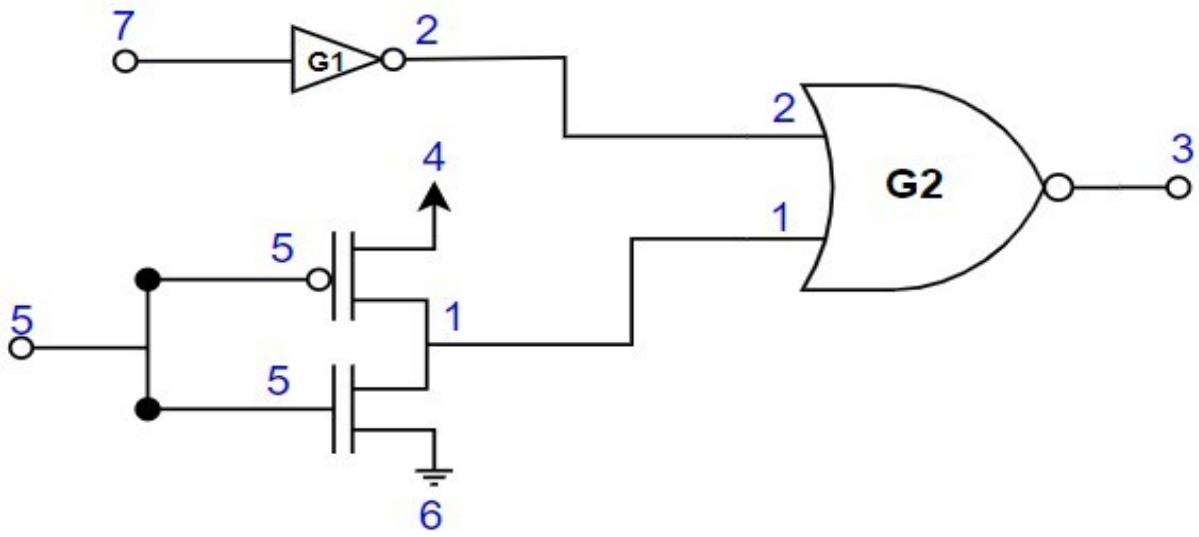
Σκοπός-Περιγραφή.

Σκοπός της τρίτης εργαστηριακής είναι η δημιουργία ενός Datasheet για τον προσομοιωτή που υλοποιήθηκε στην άσκηση 2. Διαβάζοντας ως είσοδο ένα λογικό κύκλωμα πυλών, μεταφράζοντας τις πύλες αυτές σε κύκλωμα τρανζίστορ με χρήση κατάλληλης βιβλιοθήκης και κάνοντας κατάλληλες συνδέσεις μεταξύ των κόμβων του κυκλώματος, παραγόταν αρχείο εξόδου με κύκλωμα που αποτελούταν αποκλειστικά απο τρανζίστορ και πραγματοποιώταν προσομοίωση σε flat επίπεδο.

Υλοποίηση-Κώδικας.

Αρχικά να αναφερθούμε στο ότι δεν πραγματοποιήθηκε καμία αλλαγή ή διόρθωση στον κώδικα του εργαστηρίου δύο, επομένως ο προσομοιωτής λειτουργεί ακριβώς όπως έχει περιγραφεί. Ο κώδικας του τρίτου εργαστηρίου πραγματοποιήθηκε πάνω σε αυτό του δεύτερου και υπάρχει διαχώριση αυτών με κατάλληλα σχόλια. Ξεκινώντας, διαβάζουμε το καινούργιο αρχείο και καλείται η συνάρτηση **void derive_output_file_for_lab_2(FILE *fp).** Στην συνάρτηση αυτή, αποθηκεύουμε από το αρχείο εισόδου σε global μεταβλητές πληροφορίες όπως, το όνομα του αρχείου βιβλιοθήκης, τιμές VCC και GND εάν υπάρχουν και τους κόμβου εισόδου/εξόδου του κυκλώματος. Έπειτα, καλείται η συνάρτηση **int create_CMOS_NET(FILE *fp)** που δημιουργεί το κύκλωμα αποτελούμενο μόνο από CMOS, της οποίας η ανάλυση έπεται στην συνέχεια, και επιστρέφει το πλήθος των transistor του καινούργιου netlist. Τέλος, παράγεται το αρχείο εξόδου και προσομοιώνεται άμμεσα.

Ο αλγόριθμος θα αναλυθεί παραστατικά με χρήση του κυκλώματος της παρακάτω εικόνας.



Είναι το παράδειγμα της εκφώνησης αντικαθιστώντας μία πύλη NOT με τρανζίστορ ώστε να εξηγηθεί/αποδειχθεί η λειτουργηκότητα του αλγορίθμου για ένα Netlist που αποτελείται και από τρανζίστορς. Έχουμε τυχαία αρίθμηση κόμβων καθώς δεν επιρεάζεται ο αλγόριθμος από αυτήν.

Παραθέτουμε άλλες τρεις εικόνες με το αντίστοιχο αρχείο εισόδου και βιβλιοθήκης.

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αρχείου εισόδου

```
## LIBRARY
GATES.LIB
## RAILS
VCC 4 ; GND 6
## INPUTS
7, 5
## OUTPUTS
3
## NETLIST
G1 NOT ; IN 7 ; OUT 2
U1 PMOS 5 4 1
U2 NMOS 5 1 6
G2 NOR_2 ; IN 2, 1 ; OUT 3
```

Βιβλιοθήκη Πύλη NOT

```
## GATE NOT
## RAILS
VCC 1 ; GND 4
## INPUTS
2
## OUTPUTS
3
## NETLIST
U1 PMOS 2 1 3
U2 NMOS 2 3 4
## END_GATE
```

Βιβλιοθήκη Πύλη NOR

```
## GATE NOR_2
## RAILS
VCC 1 ; GND 6
## INPUTS
2 ; 3
## OUTPUTS
5
## NETLIST
U1 PMOS 2 1 4
U2 PMOS 3 4 5
U3 NMOS 2 5 6
U4 NMOS 3 5 6
## END_GATE
```

Ο αλγόριθμος διαβάζει γραμμή προς γραμμή το αρχείο εισόδου και για κάθε γραμμή αποθηκεύει τους κόμβους που αναγράφονται. Οι κόμβοι αυτοί αφορούν, αν πρόκειται για πύλη τις εισόδους/εξόδους της, ενώ αν πρόκειται για transistor κάθε pin του. Έχουμε τις εξής μεταβλητές και δομές δεδομένων (οι πιο βασικές):

- **Offset:** Ξεκινάει με την τιμή '1' και αυξάνεται κατά '1' για κάθε καινούργια γραμμή στο Netlist του αρχείου εισόδου. Συνενώνεται (concatenate) στην αρχή του αριθμού του κόμβου.

- **CMOS_NET[X][4]:** Δισδιάστατος πίνακας όπου Χ το πλήθος των τρανζίστορ του καινούργιου netlist. Όπως δηλώνει και το όνομά του, αποθηκεύει όλα τα transistor του παραγόμενου netlist. Η στήλη Χ είναι στατική με τιμή ίση 1024.

- **EQUAL_NODES[2048][32]:** Στην πρώτη στήλη του αποθηκεύουμε τις τιμές των κόμβων που αναγράφονται στο netlist του αρχείου εισόδου. Σε αντιστοιχία κατά γραμμή, αποθηκεύονται οι ισοδύναμοι κόμβοι που δημιουργήθηκαν με χρήση του offset. Χρήσιμος για την διασύνδεση των κόμβων στο τέλος της δημιουργίας του παραγώμενου netlist από τρανζίστορ.

- **VCC_lib[256]/GND_lib[256]:** Αποθηκεύονται όλοι οι υποψήφιοι κόμβοι Τάσης/Γείωσης που παράχθηκαν με χρήση του offset.

- ***INPUT_new/*OUTPUT_new:** Αποθηκεύονται οι καινούργιοι κόμβοι εισόδου/εξόδου. Δυναμική παραχώρηση μνήμης καθώς βόλευε να αξιοποιηθεί συνάρτηση από το εργαστήριο νούμερο 2.

Έτοιμοι να προχωρήσουμε στην αναπαράσταση.

<u>1η Επανάληψη:</u> line_buffer = "G1 NOT ; IN 7 ; OUT 2"
offset = 1.
VCC_lib = [**11**]. Από βιβλιοθήκη, στο pin με αριθμό 1 εφαρμόζεται τάση. Αποθηκεύται με το offset.
GND_lib = [**14**]. Ομοίως για το pin στο οποίο εφαρμόζεται η γείωση.
INPUT_new = [**12**]. Από βιβλιοθήκη, το pin με αριθμό 2 είναι εισόδου. Αποθηκεύται με το offset.
OUTPUT_new = []. Κανένα pin από τα τρανζίστορ της πύλης αφόρα εξόδο κυκλώματος.

CMOS_NET

'P'	**12**	**11**	**13**
'N'	**12**	**13**	**14**

Αποθηκεύτηκε όλο το netlist της πύλης από την βιβλιοθήκη με καινούργιες τιμές που προκύπτουν με την χρήση του **offset**.

EQUAL_NODES

7	**12**	**-1**
2	**13**	**-1**
-1	**X**	**X**

Pin 2 είσοδος πύλης. Κόμβος netlist στο αρχείο εισόδου **7. (IN 7)**
Pin 3 έξοδος πύλης. Κόμβος netlist στο αρχείο εισόδου **2. (OYT 2)**
Με την τιμή -1 γνωρίζουμε που τελείωνουν οι πληροφορίες. Δεν αρχικοποιείται όλος ο πίνακας με -1 αλλά μόνο η πρώτη τετράδα του ύπο-πίνακα 2x2.

Αυτή η τετράδα αρχικοποίησης μας είναι αρκετή και δεν χάνουμε άδικο χρόνο πράτοντάς την με μία διπλή εμφολευμένη επανάληψη. Για αυτό αυτό υπάρχουν και οι τιμές **X**, αλλά δεν μας παρουσιάζουν κανένα πρόβλημα. Η πρώτη γραμμή του πίνακα ενημερώνεται από την συνάρτηση **void update_EQUAL_NODES_col1(int *A, int size)** χωρίς διπλότυπα. Παίρνει όρισμα έναν πίνακα και το μέγεθός του. Είναι χρήσιμο σε πύλες με πολλαπλές εισόδους (δηλαδή όλες εκτός την NOT), ώστε να ενημερωθούν όλοι οι κόμβοι με μία κλήση της συνάρτησης. Για τους ισοδύναμους κόμβους που ενημερώνονται σε αντιστοιχία γραμμών, καλείται η συνάρτηση ***void* update_EQUAL_NODES_row(*int Node*, *int Node_New*).** Αναζητεί τον κόμβο **Node** στην πρώτη στήλη, έπειτα προχωράει στην αντίστοιχη γραμμή μέχρι την τιμή -1 και ενημερώνει με τον κόμβο **Node_New** χωρίς διπλότυπα.

<u>2η Επανάληψη:</u> line_buffer = "U1 PMOS 5 4 1"
offset = 2.
VCC_lib = [**11, 24**]. Στο pin 4 εφαρμόζεται τάση σύμφωνα με το αρχείο εισόδου.
GND_lib = [**14**]. Δεν υπάρχει Pin γείωσης.
INPUT_new = [**12, 25**]. Το pin 5 είναι είσοδος κυκλώματος.
OUTPUT_new = []. Δεν υπάρχει Pin ως έξοδος κυκλώματος.
CMOS_NET

'P'	**12**	**11**	**13**
'N'	**12**	**13**	**14**
'P'	**25**	**24**	**21**

Αποθηκεύεται στο καινούργιο netlist.

EQUAL_NODES

7	**12**	**-1**
2	**13**	**-1**
5	**25**	**-1**
1	**21**	**-1**
-1	**X**	**X**

Κάθε pin του τρανζιστορ εισάγεται σε αυτόν τον πίνακα εκτός αν σε αυτό εφαρμόζεται τάση ή γείωση. Για τον λόγο αυτό δεν εισάχθηκε το pin **4**

3η Επανάληψη: line_buffer = "U2 NMOS 5 1 6"
offset = 3.
VCC_lib = [**11, 24**]. Δεν υπάρχει Pin τάσης.
GND_lib = [**14, 36**]. Στο pin 6 εφαρμόζεται γείωση σύμφωνα με το αρχείο εισόδου.
INPUT_new = [**12, 25**]. Δεν υπάρχει Pin ως είσοδο κυκλώματος Το pin 5 είναι ήδη συμπεριληφθεί.
OUTPUT_new = []. Δεν υπάρχει Pin ως έξοδος κυκλώματος.

CMOS_NET

'P'	**12**	**11**	**13**
'N'	**12**	**13**	**14**
'P'	**25**	**24**	**21**
'N'	**35**	**31**	**36**

Αποθηκεύεται στο καινούργιο netlist.

EQUAL_NODES

7	**12**	**-1**	**X**
2	**13**	**-1**	**X**
5	**25**	**35**	**-1**
1	**21**	**31**	**X**
-1	**X**	**X**	**X**

Έχουμε ισοδυναμία κόμβου 25 με 35 καθώς αυτοί επικοινωνούν σύμφωνα με το netlist του κυκλώματος. Ομοίως για 31, 21

4η Επανάληψη και τελευταία: line_buffer = "G2 NOR_2 ; IN 2,1 ; OUT 3"
offset = 4.
VCC_lib = [**11, 24, 41**]. Από αρχείο βιβλιθήκης για την πύλη NOR, κόμβος-τάσης **1.**
GND_lib = [**14, 36, 46**]. Ομοίως γείωσης.
INPUT_new = [**12, 25**]. Δεν υπάρχει κόμβος ως είσοδο κυκλώματος.
OUTPUT_new = [**45**]. Κόμβος **5** έξοδος πύλης και κυκλώματος.

CMOS_NET

'P'	**12**	**11**	**13**
'N'	**12**	**13**	**14**
'P'	**25**	**24**	**21**
'N'	**35**	**31**	**36**
'P'	**42**	**41**	**44**
'P'	**43**	**44**	**45**
'N'	**42**	**45**	**46**
'N'	**43**	**45**	**46**

Αποθηκεύεται στο καινούργιο netlist όλο το netlist της βιβλιοθήκης.

EQUAL_NODES

7	**12**	**-1**	**X**
2	**13**	**42**	**X**
5	**25**	**35**	**-1**
1	**21**	**31**	**43**
3	**45**	**-1**	**X**

Ο κόμβος **3** εισάγεται στην πρώτη στήλη σύμφωνα με το netlist του αρχείου. Έχουμε **IN 2,1** και **OUT 3** άρα ενημερώνουμε κατάλληλα του ισοδύναμους κόμβους. Προσοχή, στην βιβλιοθήκη αναγράφεται ## INPUT 2;3. Άρα ο κόμβος **2** της βιβλιοθήκης αντιστοιχή με τον **2** στο αρχείο εισόδου και ο **3** με τον **1.** Αποθηκεύονται με το **offset** ως **42 και 43.**

Τέλος τοποθετούμε και τον κόμβο εξόδου. Σε περίπτωση κάποιας ανάδρασης από έξοδο πύλης ή μία εισόδου που καταλήγει σε πολλαπλές πύλες έχει αξία να τους προσθέτουμε στον πίνακα αυτό. Στην προκειμένη περίπτωση δεν δημιουργείται κάποια ισοδυναμία.

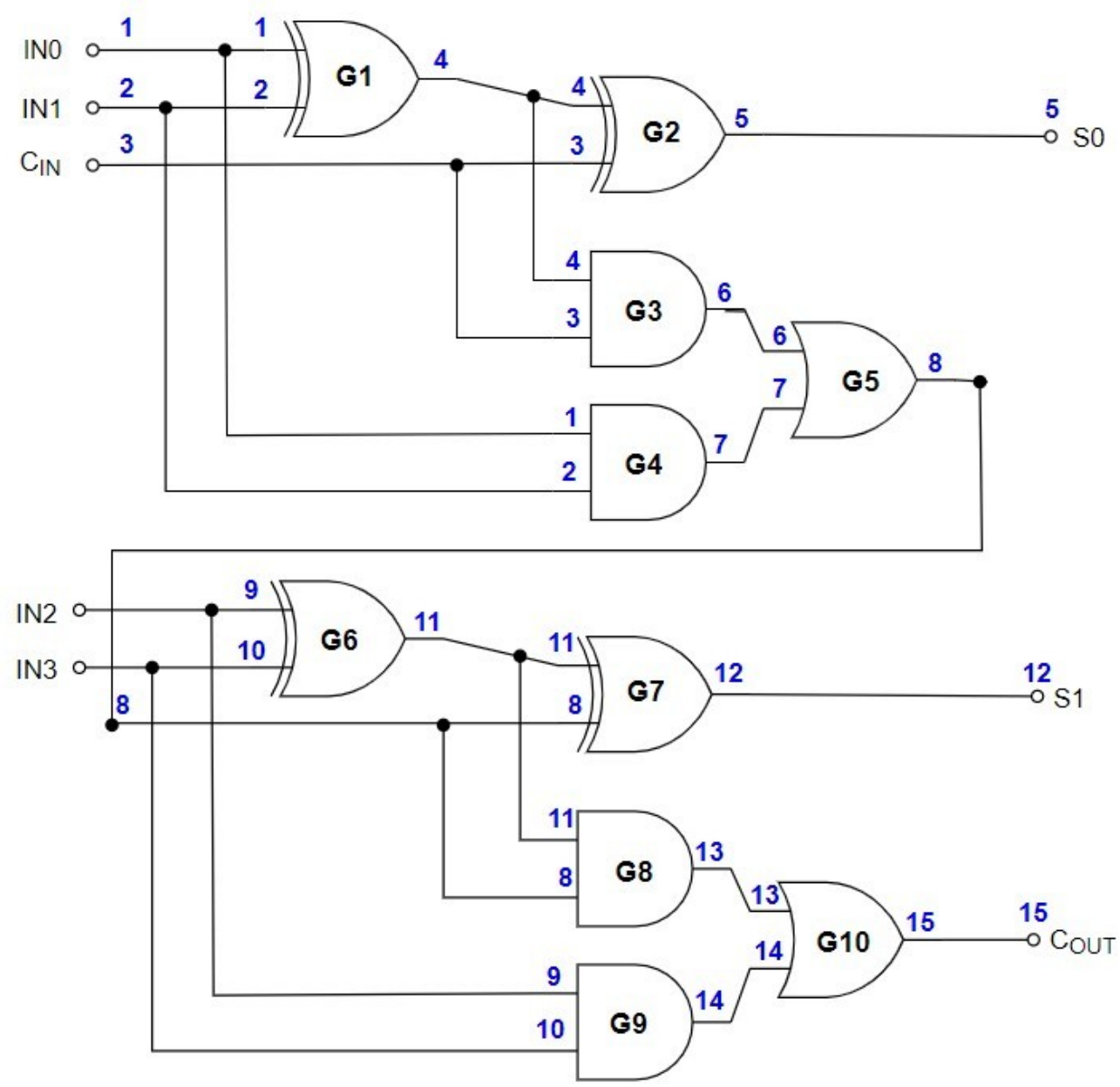
Τέλος έρχεται η συνάρτηση **final_configuration()** να συνδυάσει όλες τις πληροφορίες από τους πίνακες VCC_lib, GND_lib και EQUAL_NODES και να ενημερώσει κατάλληλα τον πίνακα CMOS_NET. Οι πίνακες INPUT_new και OUTPUT_new αξιοποιούνται κατά την δημιουργεία του αρχείου εξόδου. Η προαναφερθέντα συνάρτηση, διαλέγει τα VCC_lib[0] και GND_lib[0], δηλαδή τα πρώτα στοιχεία των πινάκων αυτών, και τα αντικαθιστά στον CMOS_NET όπου "βλέπει" τους ισοδύναμους κόμβους τάσης και γείωσης. Όμοιως, από τον πίνακα EQUAL_NODES διαλέγει την δεύτερη στήλη, δηλαδή την EQUAL_NODES[i][1], και αντικαθιστά με όλες τις ισοδυναμίες **από τους κόμβους που παράχθηκαν με την χρήση offset, όχι δηλαδη της πρώτης στήλης !** Κατά αυτόν το τρόπο ο αλγόριθμος εγγυάται ότι δεν θα πανωγραφούν κόμβοι. Πιο συγκεκριμένα, χάρης την χρήση του offset, πρώτον, στον πίνακα CMOS_NET δεν υπάρχει περίπτωση να εμφανιστούν διπλότυπα σε αυτόν, και δεύτερον, οι πίνακες VCC_lib, GND_lib και EQUAL_NODES **έχουν τομή το κενό.** Έτσι κατά την σύνθεση των αποτεσμάτων με την χρήση των παραπάνω πινάκων, δεν επρόκειτο να πανωγραφούν κόμβοι τάσης, γείωσης ή συνδέσεων.

final_configuration():

CMOS_NET Before Configuration				**EQUAL_NODES**				**CMOS_NET After Configuration.**			
'P'	12	11	13	7	**12**	-1	X	'P'	12	11	13
'N'	12	13	14	2	**13**	**42**	X	'N'	12	13	14
'P'	25	**24**	21	5	**25**	**35**	-1	'P'	25	**11**	21
'N'	**35**	**31**	**36**	1	**21**	**31**	**43**	'N'	**25**	**21**	**14**
'P'	**42**	**41**	44	3	**45**	-1	X	'P'	**13**	**11**	44
'P'	**43**	44	45	**VCC_lib** = [**11**, **24**, **41**] **GND_lib** = [**14**, **36**, **46**]				'P'	**21**	44	45
'N'	**42**	45	**46**					'N'	**13**	45	**14**
'N'	**43**	45	**46**					'N'	**21**	45	**14**

Αποτελέσματα.

Για εξακριβωθεί η ορθή λειτουργικότητα του αλγορίθμου παραγωγής Datasheet για τον απλό προσομοιωτή μας, δημιουργήθηκε NETLIST ενός RCA σύμφωνα με το παρακάτω διάγραμμα και δοκιμάστικε για 32 διανύσματα εισόδου.



Ας δούμε μερικά αποτελέσματα.

```
Input Vector = <1, 1, 1, 1, 1>
```
<=> **<IN0, IN1, C_{IN}, IN2, IN3>**

```
 24 : 1 OUTPUT
```
<=> **Σήμα S0,** δηλαδή ο κόμβος **24**

```
 74 : 1 OUTPUT
104 : 1 OUTPUT
```
<=> **Σήμα S1,** δηλαδή ο κόμβος **74**

<=> **Σήμα C_{out},** δηλαδή ο κόμβος **104**

Όντως, **$(11)_2$ + $(11)_2$ με Cin = $(11)_2$ με Cout.**

```
Input Vector = <1, 1, 0, 1, 1>
```
<=> **<IN0, IN1, C_{IN}, IN2, IN3>**

```
24 : 0 OUTPUT
```
<=> **Σήμα S0,** δηλαδή ο κόμβος **24**

```
74 : 1 OUTPUT
104 : 1 OUTPUT
```
<=> **Σήμα S1,** δηλαδή ο κόμβος **74**

<=> **Σήμα C_{out},** δηλαδή ο κόμβος **104**

Όντως, $(11)_2$ + $(11)_2$ χωρίς Cin = $(10)_2$ με Cout.

```
Input Vector = <1, 1, 1, 0, 0>
```
<=> **<IN0, IN1, C_{IN}, IN2, IN3>**

```
 24 : 1 OUTPUT
```
<=> **Σήμα S0,** δηλαδή ο κόμβος **24**

```
 74 : 1 OUTPUT
104 : 0 OUTPUT
```
<=> **Σήμα S1,** δηλαδή ο κόμβος **74**

<=> **Σήμα C_{out},** δηλαδή ο κόμβος **104**

Όντως, $(01)_2$ + $(01)_2$ με Cin = $(11)_2$ χωρίς Cout.

Επιπλέον, αλλάξαμε τον κόμβο OUT της πύλης G5 σε 18. Οι είσοδοι στις πύλες G7 και G8 παρέμειναν στην τιμή 8 και έχουμε τα εξής αποτελέσματα:

```
Input Vector = <0, 0, 0, 1, 1>
```
<=> **<IN0, IN1, C_{IN}, IN2, IN3>**

```
 24 : 0 OUTPUT
```
<=> **Σήμα S0,** δηλαδή ο κόμβος **24**

```
 74 : Z OUTPUT
104 : 1 OUTPUT
```
<=> **Σήμα S1,** δηλαδή ο κόμβος **74**

<=> **Σήμα C_{out},** δηλαδή ο κόμβος **104**

Έχουμε, $(10)_2$ + $(10)_2$ χωρίς Cin. Όντως το S0 = 0_2 και Cout = 0 αλλά λόγο της αλλαγής το σήμα S1 είναι ασύνδετο και ανυχνεύεται, όχι μόνο για αυτό το διάνυσμα εισόδου αλλά και για τα 32.

Για ένα διαφορετικό διάνυσμα εισόδου έχουμε:

```
Input Vector = <0, 0, 1, 0, 1>
```
<=> <IN0, IN1, C_{IN}, IN2, IN3>

```
24 : 1 OUTPUT
```
<=> Σήμα S0, δηλαδή ο κόμβος **24**

```
74 : Z OUTPUT
```
<=> Σήμα S1, δηλαδή ο κόμβος **74**

```
104 : Z OUTPUT
```
<=> Σήμα C_{out} , δηλαδή ο κόμβος **104**

Υπο κανονικές συνθήκες, η πράξη είναι **$(00)_2$ + $(10)_2$ με Cin = $(10)_2$ χωρίς Cout.** Άρα στον κόμβο **104** ανεμέναμε την τιμή **0.** Το γεγονός που σε αυτήν την περίπτωση έχουμε ως **Cout** το χρώμα **Z,** ενώ στην προηγούμενη περίπτωση ενημερώθηκε κανονικά με το χρώμα **1**, οφείλεται σε μία ιδιατερότητα της πύλης XOR εν αντιθέση των υπολοίπων πυλών. Η πύλη XOR για να δώσει έξοδο χρειάζεται αναγκαστικά και στις δύο εισόδους της τιμή. Αντιθέτως, η πύλη AND αρκεί μία της είσοδος να είναι μηδέν για να αποδώσει έξοδο **0** και η OR αρκεί να έχει έναν άσσο για να αποδώσει **1.** Όντως, στην προηγούμενη περίπτωση για **IN2 = 1 και IN3 = 1** η πύλη η πύλη G9 AND (βλ. διάγραμμα) παράγει στην έξοδο άσσο, ο οποίος μέσο της G10 OR μεταφέρεται στο σήμα Cout. Στη συγκεκριμένη περίπτωση τώρα, έχουμε **IN2 = 0, IN3 = 1 και Cin = X** για τον δεύτερο FA (κόμβος 8 που άλλαξαμε σε 18.) Ως αποτέλεσμα, οι δύο πύλες AND δεν μπορούν να δώσουν τιμή εξόδου και κατ'επέκασιν ούτε η πύλη OR, πράγμα που σημαίνει ανοιχτοκύκλωμα στην έξοδο.

Παραδοτέα

1. **RCA_AND_OR_V32.txt:** Η υλοποίηση RCA όπου το τμήμα για τον υπολογισμό του κρατούμενου είναι με συνδεσμολογία AND-OR. Έστω **X** το όνομα αυτού του αρχείου.

2. **RCA_AND_OR_V32.txt**: Η υλοποίηση RCA όπου το τμήμα για τον υπολογισμό του κρατούμενου είναι με συνδεσμολογία NAND-NAND. Έστω **Y** το όνομα αυτού του αρχείου.

3. **LAB_3_OUTPUT_[X/Y]:** Τα αντίστοιχα παραγώμενα αρχεία εξόδου με NETLIST αποτελούμενο μόνο από τρανζιστορς για τα παραπάνω κυκλώματα. Έστω **O_X/O_Y** τα ονόματα των αρχείων αυτών.

4. **Result_[O_X/O_Y]:** Τα αποτελέσματα του προσομοιωτή μας.