ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

[HRY 419]-Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων

Αναφορά 5ου εργαστηρίου

Ιωάννης Περίδης
A.M. 2018030069

22 Μαΐου 2022

1 Εισαγωγή:

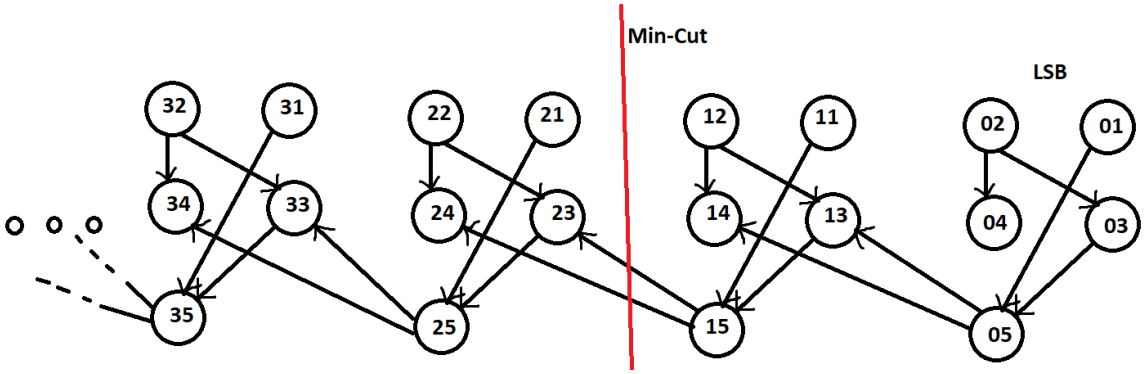
Στην εργαστηριακή άσκηση αυτή , μας ζητήθηκε να βρούμε μια ντετερμινιστική μέθοδο για την τοποθέτηση και διασύνδεση των standard shells. Συγκεκριμένα, αντί να γίνονται τυχαίες αντιμεταθέσεις πυλών και πολλά routing, γίνονται πιο έξυπνες και στοχευμένες αντιμεταθέσεις , χρησιμοποιώντας την θεωρία γράφων.

2 Θεωρία Γράφων:

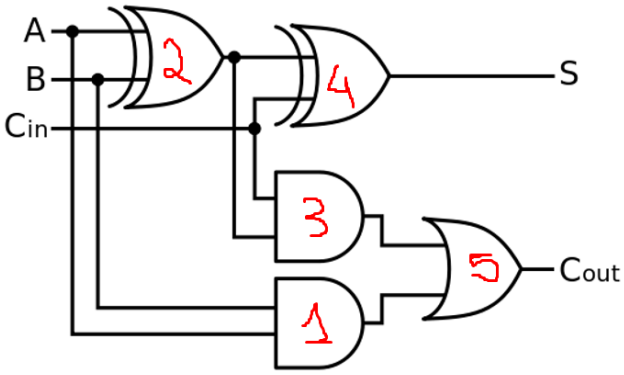
Η θεωρία γράφων, μπορεί να γίνει ένα πολύ χρήσιμο εργαλείο για την εύρεση μιας αρκετά καλής λύσης σε αυτό το πρόβλημα NP-complete της τοποθέτησης των πυλών με τον πιο αποδοτικό τρόπο.

Αρχίζοντας, θα απεικονίσουμε ολόκληρο το κύκλωμα σαν έναν γράφο με όλες τις διασυνδέσεις πυλών. Σε αυτόν τον γράφο κάθε πύλη θα απεικονίζει έναν κόμβο και κάθε σύνδεση μεταξύ πυλών θα απεικονίζει μια ακμή. Ύστερα, μέσα σε αυτόν τον γράφο των 40 κόμβων , αφού έχουμε 8 Bit adder, θα ψάξουμε τους 4 υπογράφους των 20 κόμβων ο καθένας , οι οποίοι θα έχουν την μεγαλύτερη εντός κάθε υπογράφου διασυνδεσιμότητα και την μικρότερη σε σχέση αναμεταξύ τους. Η τεχνική αυτή είναι παρόμοια με τον αλγόριθμο Mincut , οποίος προσπαθεί να κόψει τους υπογράφους στα σημεία εκείνα που έχουν την ελάχιστη συσχέτιση μεταξύ τους. Ιδανικά, γνωρίζοντας ότι έχουμε έναν Ripple Carry Adder των 8 Bit, το mincut θα ήταν να κόψουμε κάθε 1 ή παραπάνω bit μετά την πύλη OR η οποία μεταδίδει το κρατούμενο στο επόμενο Bit και έτσι θα είχαμε μεταξύ των υπογράφων μονάχα 1 σύρμα που θα πέρναγε σε 2 εισόδους. Παρόλα αυτά, είναι πολύ δύσκολο να βρεθεί η βέλτιστη λύση με αντιμεταθέσεις χωρίς να γνωρίζουμε τι κύκλωμα έχουμε με κάποιον γενικό κανόνα.

Παρακάτω φαίνεται σχηματικά πως θα ήταν το ιδανικό χώρισμα των υπογράφων (φαίνονται μόνο οι 2 υπογάφοι από τους 4, οι άλλοι θα ήταν αντίστοιχα στην σειρά ο ένας μετά τον άλλο) :



Κάθε κόμβος αναπαριστά μια πύλη με τους αριθμούς που φαίνονται στο παρακάτω σχήμα. Κάθε 5 κόμβοι αποτελούν 1 FA, δηλαδή 1 bit του αθροιστή:



3 Υλοποίηση Γράφου:

Η υλοποίηση του ολοκληρωμένου γράφου αυτού σε κώδικα είχε ως εξής. Αρχικά, δημιουργήθηκε ένας πίνακας graphcConnections[][] μήκους όσο το πλήθος των shells και στις 2 διαστάσεις . Ο πίνακας αυτός , αναπαριστά τις διασυνδέσεις των κόμβων. Αν 2 κόμβοι (row,col), συνδέονται τότε το αντίστοιχο κελί περιέχει ‘1’, αν δεν συνδέονται περιέχει ‘0’. Ο γράφος μας, θα είναι μη κατευθυντικός , δηλαδή θα υπάρχει ‘1’ στην περίπτωση οποιαδήποτε σύνδεσης, όχι μόνο όταν μια έξοδος κατευθύνεται κάπου. Αυτό διότι , κατ’ αυτόν τον τρόπο, παίρνουμε πολύ περισσότερη πληροφορία από τον γράφο , καθώς και αν ήταν μη κατευθυντικός , οι κόμβοι των πυλών X4 που δίνουν το sum, θα είχαν κενές τις γραμμές τους στον πίνακα (όλο ‘0’). Για την διαδικασία γεμίσματος όλων των 40χ40 κελιών του πίνακα, μέσα σε έναν βρόγχο, έγινε το γέμισμα των κελιών όλων των συνδέσεων των 5 πυλών από 1 Bit του αθροιστή και ύστερα απλά γεμίστηκαν και τα υπόλοιπα κελιά επαναλαμβάνοντας την διαδικασία άλλες 7 φορές , απλά μετακινώντας τις αντίστοιχες σειρές και στήλες.

Ύστερα, δημιουργήθηκε ο τελικός πίνακας totalGraph[][], που έχει το ίδιο μήκος με τον παραπάνω πίνακα με τις συνδέσεις, απλά έχουν προστεθεί στην πρώτη γραμμή και στην πρώτη στήλη , ο αριθμός κάθε κόμβου (κάθε πύλης), για να φαίνονται οπτικά ποιες είναι οι συνδέσεις. Στην πρώτη γραμμή, τοποθετήθηκαν οι 40 πύλες του αθροιστή με την σειρά, όπως δίνονται από το routing του netlist, ενώ στην πρώτη στήλη, τοποθετήθηκαν οι πύλες έτσι όπως δίνονται στο placement του netlist, ανακατεμένες. Προφανώς αρχικά οι συνδέσεις του γράφου ήταν στην σειρά, με τις οποίες τοποθετήθηκαν, έτσι έγινε μέσω της συνάρτησης swapRows η αντιστοίχιση κάθε γραμμής συνδέσεων του πίνακα, με την αντίστοιχη πύλη που βρισκότανε. Προφανώς το πρώτο κελί του πίνακα το [0][0] είναι ένα κενό στοιχείο.

Παρακάτω φαίνεται εκτυπωμένος ο τελικός γράφος του κυκλώματος που έχει αναπαρασταθεί σαν τον παραπάνω πίνακα που αναλύθηκε. Με κίτρινο χρώμα σχεδιάστηκε από πάνω μια διαχωριστική γραμμή για να φαίνονται καθαρότερα τα όρια του πίνακα :

Οι συνδέσεις προφανώς τώρα θα είναι εντελώς τυχαία κατανεμημένες με τεράστια διασπορά σε όλον τον πίνακα. Αυτό συμβαίνει διότι οι πύλες που συνδέονται μεταξύ τους, δεν βρίσκονται κοντά η μια στην άλλη.

```
***This is the Circuit Represented as a Graph***
==================================================
```

0	1	2	3	4	5	11	12	13	14	15	21	22	23	24	25	31	32	33	34	35	41	42	43	44	45	51	52	53	54	55	61	62	63	64	65	71	72	73	74	75
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1		
54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0		
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
75	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0		
72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0			
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0			
53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0		
15	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
25	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
3	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		
12	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
5	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
24	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
13	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0		
62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0		
73	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
22	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
4	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	
74	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
23	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Παρακάτω φαίνεται εκτυπωμένος ο πίνακας, που απεικονίζει πως θα ήταν ο βέλτιστος γράφος, δηλαδή όταν οι πύλες ήταν τοποθετημένες στην σειρά:

0	1	2	3	4	5	11	12	13	14	15	21	22	23	24	25	31	32	33	34	35	41	42	43	44	45	51	52	53	54	55	61	62	63	64	65	71	72	73	74	75
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0
61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0
71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
73	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
74	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Οι χοντρές κίτρινες γραμμές , είναι εκείνες που δείχνουν που θα διαχωρίζονταν οι γράφοι στην άσκησή μας. Ο κάθε ένας από τους 4 γράφους , περιέχει 2 Bits, τα οποία φαίνονται στα τετράγωνα. Τέλος, οι διασυνδέσεις μεταξύ των Bit είναι κυκλωμένες (τα γωνιακά σχήματα). Οι πιο σημαντικές διασυνδέσεις είναι εκείνες στις πύλες X5 με (X+1)3 και (X+1)4, οι οποίες μεταδίδουν το κρατούμενο και είναι αυτές που ανα 2 κόβουμε σαν Mincut για να έχουμε το βέλτιστο αποτέλεσμα.

Παρατηρείται πως για την βέλτιστη λύση, έχουμε όλους τους '1' μαζεμένους κοντά ο ένας στον άλλο με όσο πιο μικρή διασπορά γίνεται. Με αυτόν τον τρόπο πετυχαίνουμε μεγάλη διασυνδεσιμότητα εντός υπογράφων και ελάχιστη διασυνδεσιμότητα των μεταξύ τους υπογράφων .Επίσης, παρατηρούμε πως τα bits, έχουν τις συνδέσεις τους , όλα στην κύρια διαγώνιο του πίνακα. Αυτό συμβαίνει γιατί έτσι πετυχαίνουμε το ελάχιστο ολικό εμβαδόν και ίσο με $\mathbf{5^2}\ \boldsymbol{\pi\acute{\upsilon}\lambda\varepsilon\varsigma} * \mathbf{8}\ \boldsymbol{bits} = \mathbf{200}$.

Αυτή η λύση φυσικά, είναι η βέλτιστη και την δημιουργήσαμε μονάχα γνωρίζοντας το τελικό αποτέλεσμα. Στον παρακάτω αλγόριθμό μας, θα προσπαθήσουμε να πετύχουμε μια καλή λύση , αρχίζοντας από ένα τελείως τυχαίο Placement και χωρίς να χρησιμοποιήσουμε τακτικές που γνωρίζουμε ότι θα μας φέρουνε στην καλύτερη λύση, αλλά θα χρησιμοποιήσουμε αλγόριθμο τέτοιον ώστε να μπορεί να δουλέψει σε ένα οπουδήποτε τυχαίο κύκλωμα (φυσικά αυτός δεν θα είναι βέλτιστος , αλλά θα αποτελεί μια καλύτερη λύση στα σίγουρα).

4 Αλγόριθμος Ντετερμινιστικού Placement:

Ο αλγόριθμος που υλοποιήθηκε είχε την εξής λογική. Είναι φανερό , πως θέλουμε να μειώσουμε την διασπορά του πίνακα σε '1' και να φέρουμε πιο κοντά όσες πύλες συνδέονται μεταξύ τους. Αυτό θα γίνει, αλλάζοντας 2 κάθε φορά γραμμές (πύλες δηλαδή) αναμεταξύ τους, με σκοπό να τις φέρουμε πιο κοντά. Θέλει μεγάλη προσοχή η επιλογή των σωστών πυλών προς αντιμετάθεση, γιατί κάθε φορά που γίνεται μια αντιμετάθεση για να έρθει μια πύλη κοντά στην άλλη, μπορεί αντίστοιχα στον πίνακα να αλλάξουν 2 πύλες που ήταν ήδη κοντά μεταξύ τους και να απομακρυνθούνε. Συγκεκριμένα , θέλουμε να επιλέξουμε να φέρουμε κοντά εκείνες τις πύλες που έχουν την μεγαλύτερη διασυνδεσιμότητα. Οι πύλες αυτές όπως αναφέρθηκε και πριν είναι εκείνες που περνάνε το κρατούμενο από Bit σε Bit οι X5 και έχουν τον μέγιστο αριθμό ακμών στον κόμβο και είναι ίσος με 4 (2 ακμές πάνε στις πύλες (X+1)3 και (X+1)4 και 2 ακμές έρχονται από τις X3 και X1).

Τελικά, αυτό που έγινε ήταν το εξής, βρέθηκε για κάθε στήλη ο μέγιστος αριθμός συνδέσεων της (γιατί είπαμε πως θέλουμε να δουλεύει για κάθε κύκλωμα και όχι μόνο για αυτό που ξέρουμε την λύση).Αυτό υλοποιήθηκε, αθροίζοντας τους '1' κάθε στήλης και αποθηκεύοντας το αποτέλεσμα σε ένα διάνυσμα sum[].

Ύστερα, βρέθηκε σε ποια θέση βρίσκεται , δηλαδή σε ποια γραμμή του πίνακα βρίσκεται η πρώτη , η δεύτερη ,η τρίτη και η τέταρτη σύνδεση και κρατήθηκαν σε κάποιες μεταβλητές. Αυτό έγινε , εξερευνώντας τις στήλες του πίνακα και κρατώντας την γραμμή όταν βρίσκαμε κελί ίσο με '1'.

Έπειτα, γίνεται για την πρώτη σύνδεση , αλλαγή γραμμής (δηλαδή αλλαγή των πυλών) της γραμμής που βρισκόταν η δεύτερη σύνδεση και της γραμμής που βρισκόταν 2 σειρές πάνω από την πρώτη σύνδεση. Αντίστοιχα γίνεται για την δεύτερη σύνδεση , αλλαγή γραμμής της γραμμής που βρίσκεται η Τρίτη σύνδεση και της γραμμής που βρίσκεται 1 σειρά τώρα πάνω από την πρώτη σύνδεση. Τέλος για την τέταρτη σύνδεση , γίνεται αλλαγή γραμμής αυτής που βρίσκεται η τέταρτη σύνδεση με την γραμμή που βρίσκεται 3 σειρές κάτω από την πρώτη σύνδεση. Επομένως , επιτυγχάνεται να έρθουν οι πύλες που συνδέονται κοντά. Τα ακριβής νούμερα αυτά , αποδείχθηκαν μετά από δοκιμές τα καλύτερα όσον αφορά την αποτελεσματικότητα της λύσης, δηλαδή το πόσο κοντά έρχονται οι συνδεδεμένες πύλες ή

πόσο απομακρύνονται. Πρέπει να αναφέρουμε πως κάθε φορά που κάνουμε μια αλλαγή που μας επιφέρει καλό αποτέλεσμα, μπορεί η ίδια αλλαγή να επιφέρει ταυτόχρονα και κακό αποτέλεσμα, απομακρύνοντας 2 κοντινές πύλες, οπότε δεν είναι εφικτό να βρούμε την βέλτιστη λύση. Παρόλα αυτά , προσπαθούμε να βρούμε μια όσο το δυνατόν καλύτερη.

Παρακάτω φαίνεται εκτυπωμένος ο πίνακας, που απεικονίζει το κύκλωμα, μετά τις διαρρυθμίσεις των γραμμών(αλλαγές πυλών-κόμβων):

```
***This is the Graph After the Re-Arrangement of the Nodes-Gates***
=====================================================================
 0  1  2  3  4  5 11 12 13 14 15 21 22 23 24 25 31 32 33 34 35 41 42 43 44 45 51 52 53 54 55 61 62 63 64 65 71 72 73 74 75
71  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
73  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1
63  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  1
54  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0
53  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0
64  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1
 4  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
61  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
51  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
24  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
72  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0
45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0
23  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
15  0  0  0  0  0  1  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
25  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
14  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
11  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
12  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 5  1  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
21  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 2  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
42  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
34  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
13  0  0  0  0  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
75  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0
52  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0
55  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  1  1  0  0  0  0  0
62  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0
41  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
22  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
74  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
33  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
65  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  1  1  0
44  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
35  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0
 3  0  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Παρατηρούμε πως πράγματι σε αρκετά σημεία έχει μειωθεί η διασπορά των ‘1’ και έχουν έρθει πιο κοντά αναμεταξύ τους.

Εφόσον πλέον έχει γίνει η αναδιάταξη των κόμβων, χωρίζουμε τον ολικό γράφο σε 4 υπογράφους. Πρακτικά γεμίζουμε 4 διανύσματα subGraph[][] με τον αριθμό των κόμβων , κάθε ένα θα περιέχει 10 γραμμές του πίνακα με την σειρά. Στο σχήμα φαίνεται ο διαχωρισμός τους με την κόκκινη γραμμή. Στόχος μας τώρα είναι να γεμίσουμε έναν τελικό πίνακα Placement[] με τους 4 υπογράφους και να κάνουμε το routing και το compaction για να βρούμε την τελική μας λύση. Έχει σημασία όμως με ποια σειρά θα τοποθετήσουμε τους 4 υπογράφους στο τελικό placement, καθώς και θα δοθούν αρκετές διαφορετικές λύσεις και εμείς θέλουμε να κρατήσουμε την καλύτερη. Όλες οι πιθανές λύσεις είναι διατάξεις των 4 ανα 4 , χωρίς επαναλήψεις, δηλαδή 4!=24 πιθανοί συνδυασμοί. Είναι πολύ απλό λοιπόν να δοκιμάσουμε και τους 24 αυτούς συνδυασμούς να κάνουμε routing για τον καθένα από αυτούς και να κρατήσουμε την καλύτερη λύση. Στην δικιά μας περίπτωση η καλύτερη λύση ήταν η τοποθέτηση στον πίνακα Placement ξεκινώντας από το στοιχείο 0, πρώτα τον υπογράφο , 4 μετά τον 2, μετά τον 1 και τέλος τον 3. Έχει σχέση κάθε φορά με το που έχουν μαζευτεί τα Bits Που θα είναι πρώτα και που τα τελευταία στην άθροιση σειριακά.

(Δεν έχει προστεθεί ο κώδικας που κάνει τα πολλά routing στην λύση , καθώς και ήταν απλά 24 φορές ένα Loop που γεμίζει το Placement και τρέχει την συνάρτηση route and compact)

5 Αποτελέσματα & Σύγκριση:

Παρακάτω θα συγκριθούν οι 2 λύσεις που βρέθηκαν , αρχικά κάνοντας απλά routing and compaction στο αρχικό placement που είναι τυχαίο από το netlist Askhsh_2_FA_8bit_v2 και η λύση που μας δίνει το Placement αφού έχει αναδιοργανωθεί, χρησιμοποιώντας την θεωρία των γράφων. Συγκεκριμένα θα συγκρίνουμε τις γραμμές που χρειάζεται το τελικό κύκλωμα, το μήκος του μακρύτερου σύρματος , μετρημένο σε απόσταση μεταξύ std shells και τέλος τον μέσο όρο μήκους όλων των συρμάτων.

Παρακάτω φαίνονται όλα τα αποτελέσματα , του route and compact του αρχικού τυχαίου Placement:

```
***This is the Random Starting Placement Given from the Netlist:***
-----------------------------------------------------------------------------------------------------------------
52|51|44|34|1|21|63|54|42|75|72|45|53|15|25|3|33|12|5|24|2|13|71|41|11|14|55|62|73|22|31|4|65|74|61|32|23|35|64|43|
-----------------------------------------------------------------------------------------------------------------
```

```
***The Length of the Biggest Wire (measured in std shells distance) is 35***
***The Mean Length of All Wires   (measured in std shells distance) is  6***
================================================================================

*****ROWS IN FINAL SOLUTION*****
==========================================
Number of rows after the Place & Route algorithm:19
```

Παρακάτω φαίνονται όλα τα αποτελέσματα , του route and compact του placement , αφού έχει γίνει η επεξεργασία με τους γράφους:

```
***This is the Placement After the Graph-Theory Based Re-Arrangement of the Gates:***
-----------------------------------------------------------------------------------------------------------------
74|33|65|44|31|32|1|35|43|3|72|45|23|15|25|14|11|12|5|21|71|73|63|54|53|64|4|61|51|24|2|42|34|13|75|52|55|62|41|22|
-----------------------------------------------------------------------------------------------------------------
```

```
***The Length of the Biggest Wire (measured in std shells distance) is 28***
***The Mean Length of All Wires   (measured in std shells distance) is  4***
================================================================================

*****ROWS IN FINAL SOLUTION*****
==========================================
Number of rows after the Place & Route algorithm:14
```

Παρατηρούμε πως τα αποτελέσματα είναι επιτυχής. Έχουμε μείωση των γραμμών της λύσης από 19 σε 14, δηλαδή ένα ποσό περίπου του 26,3%. Ακόμη , υπάρχει μείωση στο μήκος του μεγαλύτερου σύρματος από 35 σε 28 και μια γενική μείωση στον μέσο όρο μηκών των συρμάτων από 6 σε 4 , γεγονός που είναι αρκετά καλό.

Συμπερασματικά, μπορεί να μην πετυχαίνουμε μια τόσο καλύτερη λύση , αλλά χρησιμοποιούμε την θεωρία που γνωρίζουμε για να κάνουμε έξυπνες αλλαγές και όχι τυχαίες και έτσι γλιτώνουμε πολλά βήματα , πολλές επαναλήψεις και πολλά route and compact. Συγκεκριμένα σε σχέση με τον αλγόριθμο simulated annealing, για το netlist αυτό, χρειάζονταν για να πέσει στις 11 γραμμές λύση 336 επαναλήψεις του αλγορίθμου route and compact. Αντίθετα τώρα, χρησιμοποιώντας την θεωρία γράφων και 24 μόνο επαναλήψεις του

αλγορίθμου route and compact για να βρεθεί η βέλτιστη τοποθέτηση των γράφων στο Placement, βρήκαμε μια λύση στις 14 σειρές, αρκετά κοντά στις 11 με πολύ λιγότερο κόστος,