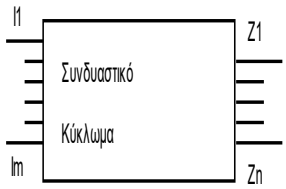
7. ΣΧΕΔΙΑΣΗ ΣΥΝΔΥΑΣΤΙΚΩΝ ΚΥΚΛΩΜΑΤΩΝ

7.1 Εισαγωγή

Ένα συνδυαστικό κύκλωμα παριστάνεται με ένα απλό block διάγραμμα όπου με "**m"** σημειώνουμε το πλήθος εισόδων και με "**n**" το πλήθος των εξόδων του, όπου κάθε μια από αυτές είναι συνάρτηση μερικών ή όλων των εισόδων.

Δηλαδή Fi=f(I1,...,Im) με i=1,2,3,..,n.



Ένα τέτοιο κύκλωμα στο οποίο κάθε έξοδός του εξαρτάται μόνο από τις τιμές των εισόδων, λέγεται "σ**υνδυαστικό κύκλωμα**" (combinational circuit). Τα σήματα εισόδου και εξόδου εμφανίζονται σχεδόν ταυτόχρονα ή στην πραγματικότητα με μια μικρή χρονική καθυστέρηση **δt**. Ο χρόνος **δt** εξαρτάται από την καθυστέρηση προσπέλασης των πυλών του και από το πλήθος των πυλών που συναντά το σήμα από την είσοδο μέχρι την έξοδο.

Στα ολοκληρωμένα κυκλώματα της οικογένειας TTL, η χρονική καθυστέρηση για ένα αντιστροφέα είναι 5ns και 10ns για την άνοδο και την κάθοδο του παλμού εισόδου.

Μερικά βασικά κυκλώματα συνδυαστικής λογικής είναι κωδικοποιητές, αποκωδικοποιητές, συγκριτές, ημιαθροιστές, αθροιστές, πολυπλέκτες, αποπολυπλέκτες κ.λ.π.

7.2 Σχεδίαση κυκλωμάτων Κωδικοποιητών (Encoders)

Ο κωδικοποιητής είναι ένα συνδυαστικό κύκλωμα με **m-**αριθμό εισόδων από τις οποίες **μόνο μία** είναι ένα-"1" και δίδει στην έξοδό του ένα κωδικό Ν δυαδικών ψηφίων, ο οποίος εξαρτάται από το ποια είσοδος είναι ένα-"1".

Πίνακας Εισόδων - Εξόδων

D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	E3	E2	E1	E0
0	0	0	0	0	0	0	0	0	**1**	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	**1**	0	0	0	0	1	1
0	0	0	0	0	0	**1**	0	0	0	0	0	1	0
0	0	0	0	0	**1**	0	0	0	0	0	1	1	0
0	0	0	0	**1**	0	0	0	0	0	0	1	1	1
0	0	0	**1**	0	0	0	0	0	0	0	1	0	1
0	0	**1**	0	0	0	0	0	0	0	0	1	0	0
0	**1**	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

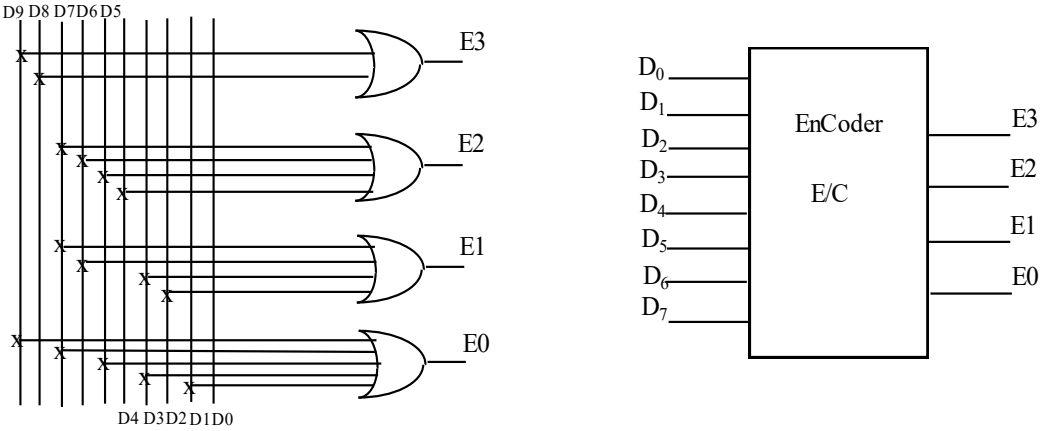
m - είσοδοι (μόνο μία "1") n- έξοδοι

Αν **n** είναι το πλήθος των εξόδων του κωδικοποιητή τότε το πλήθος **m** των εισόδων του δίδεται από την σχέση $2^{\mathbf{n}} \geq \mathbf{m}$. Αν θέλουμε να κωδικοποιήσουμε τα ψηφία του δεκαδικού συστήματος (π.χ στον κώδικα BCD-8421), ο κωδικοποιητής θα έχει **m**=10 το πλήθος εισόδους και **n**=4 το πλήθος εξόδους ($2^{\mathbf{n}}\geq\mathbf{m}$ άρα $2^{\mathbf{4}}=\mathbf{10}$), οι οποίες θα δίνουν τον κώδικα BCD-8421, και από τις οποίες **μια** μόνο είσοδος θα είναι ένα-"1" και οι υπόλοιπες είναι μηδέν-"0". Δηλαδή E3=1 όταν D8=1 ή D9=1.

Οι εξισώσεις που προκύπτουν είναι :

E3=D8+D9 E2=D4+D5+D6+D7 E1=D2+D3+D6+D7 E0=D1+D3+D5+D7+D9

Το κύκλωμα του κωδικοποιητή με βάση τις εξισώσεις σχεδιάζεται όπως στο σχήμα.



Άλλο παράδειγμα είναι η κωδικοποίηση του πληκτρολογίου των Η/Υ, ώστε με το πάτημα ενός πλήκτρου να παράγεται ένας δυαδικός κώδικας οκτώ ψηφίων (8 bits). Στο πληκτρολόγιο (KBD) υπάρχουν 26 μικρά και 26 κεφαλαία γράμματα, 10 αριθμητικά ψηφία και 22 ειδικοί χαρακτήρες και το σύνολο είναι 84 πλήκτρα (εκτός των πλήκτρων ειδικών Λειτουργιών). Ο κώδικας που θα χρησιμοποιηθεί πρέπει να έχει τουλάχιστον 7 ψηφία, ώστε να ισχύει η σχέση $\mathbf{2^6<84<2^7}$. Τέτοιος κώδικας είναι ο γνωστός στους υπολογιστές ASCII. Οι κωδικοποιητές αποτελούν τμήματα μνήμης ROM μόνο τεχνολογίας LSI.

7.3 Σχεδίαση Κυκλωμάτων Μετατροπής Κωδίκων

Στα κυκλώματα εφαρμογής χρησιμοποιούνται διάφοροι λογικοί κώδικες οι οποίοι αντιπροσωπεύουν αριθμούς και απαιτείται η μετατροπή τους από τον ένα κώδικα σε ένα άλλο κώδικα, π.χ. από δεκαδικό σε δυαδικό, από δυαδικό σε Gray, από δυαδικό σε BCD ή από BCD σε κώδικα απεικόνισης επτά (7) τμημάτων για την εμφάνιση δεδομένων που αναγνωρίζονται από τον άνθρωπο κ.τ.λ. Γενικά η σχεδίαση των κυκλωμάτων μετατροπής των κωδίκων γίνεται αφού πρώτα απλοποιηθούν οι λογικές

συναρτήσεις που προκύπτουν από τους πίνακες αληθείας των κωδίκων συνήθως με την χρήση Καρνώ, επειδή μεταφέρονται εύκολα σε αυτούς οι όροι των συναρτήσεων και λαμβάνεται υπ' όψιν και η βοήθεια των αδιάφορων όρων, εάν υπάρχουν, ώστε να προκύψουν οι απλούστερες τελικές συναρτήσεις.

7.3.1 Κύκλωμα Μετατροπής δυαδικού κώδικα σε κώδικα Gray

Ας δούμε με παράδειγμα την μετατροπή ενός δυαδικού κώδικα με τέσσερα ψηφία σε κώδικα GRAY ο οποίος έχει επίσης τέσσερα ψηφία, τη βασική διαδικασία της.

Κατ' αρχή γράφουμε τους κώδικες στον πίνακα εισόδων-εξόδων. Στο τμήμα εισόδων τον δυαδικό κώδικα και στο τμήμα εξόδων τον κώδικα Gray. (Στο αριστερό μέρος δηλαδή ο κώδικας που θα μετατρέψουμε και δεξιά ο κώδικας που θέλουμε να πάρουμε).

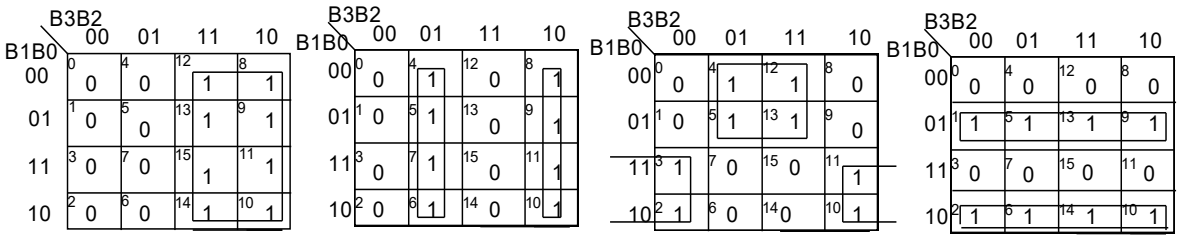
Πίνακας Εισόδων - Εξόδων

B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Στη συνέχεια θεωρούμε κάθε ψηφίο του κώδικα εξόδου (κώδικας Gray) **Gi**, σαν μια συνάρτηση των μεταβλητών B3,B2, B1,B0 του δυαδικού κώδικα και γράφουμε τη λογική συνάρτηση της αντίστοιχης εξόδου.

Για να απλοποιήσουμε τις συναρτήσεις των εξόδων θα χρησιμοποιήσουμε την μέθοδο του Χάρτη Καρνώ.

Γράφουμε ένα αρχικό Καρνώ που τον ονομάζουμε **χάρτη κλειδί**, τον οποίο αριθμούμε με βάση τον κώδικα που μετατρέπουμε (εδώ τον δυαδικό κώδικα) και σημειώνουμε τους αδιάφορους όρους (αν υπάρχουν). Αντιγράφουμε το **χάρτη κλειδί** 4 φορές (1 για κάθε μια έξοδο), από τους οποίους έχουμε τις απλοποιημένες εκφράσεις για τις λογικές συναρτήσεις των εξόδων Gi του κωδικοποιητή. Εδώ δεν έχουμε αδιάφορους όρους οπότε οι χάρτες απλοποίησης των λογικών συναρτήσεων φαίνονται παρακάτω.



Από τους Καρνώ, με την γνωστή διαδικασία, προκύπτουν οι τελικές απλοποιημένες λογικές συναρτήσεις (Α.Λ.Σ) του κυκλώματος του μετατροπέα κωδικοποίησης που είναι:

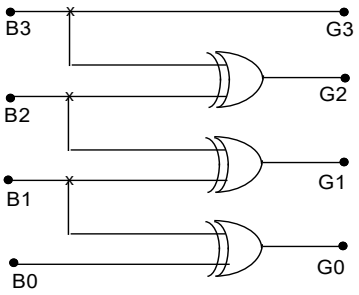
1η G3=B3 2η G2=$\bar{B}3B2+B3\bar{B}2$=B3⊕B2 πύλη XOR

3η G1=$\bar{B}2B1+B2\bar{B}1$=B2⊕B1 πύλη XOR 4η G0=$\bar{B}1B0+B1\bar{B}0$=B1⊕B0 πύλη XOR

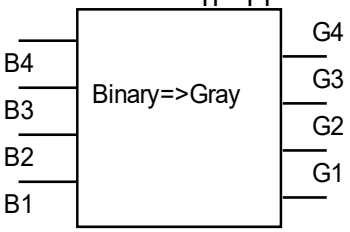
Γενικά μεταξύ των μεταβλητών **Gi και Bi** ισχύει η σχέση **Gn=Bn⊕Bn+1**

Το κύκλωμα του κωδικοποιητή σχεδιάζεται με πύλες XOR όπως φαίνεται στο σχήμα.

Αναλυτικό κύκλωμα



Block διάγραμμα



Αν τώρα θέλουμε το αντίστροφο κύκλωμα μετατροπής, του κώδικα Gray στον δυαδικό κώδικα, αλλάζουμε την θέση των κωδίκων στον πίνακα, ακολουθούμε την ίδια διαδικασία και οι τελικές Α.Λ.Σ των εξόδων θα είναι: 1η B3=G3 2η B3=$\bar{G}3G2+G3\bar{G}2$=G3⊕G2 πύλη XOR

3η B1=$\bar{G}2G1+G2\bar{G}1$=G2⊕G1 πύλη XOR 4η B0=$\bar{G}1G0+G1\bar{G}0$ =G1⊕G0 πύλη XOR

από τις οποίες σχεδιάζεται το κύκλωμα του κωδικοποιητή με πύλες XOR.

7.3.2 Κύκλωμα Μετατροπής του κώδικα Χ-3 στον κώδικα BCD-2421

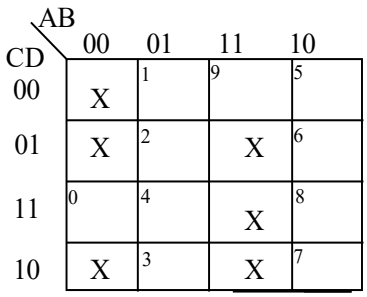
Το δεύτερο παράδειγμα αφορά την μετατροπή του κώδικα υπερβολής κατά 3 (του BCD Χ-3) στον κώδικα BCD-2421.

♦Γράφουμε τους κώδικες στον πίνακα εισόδων-εξόδων. Στο τμήμα εισόδων (αριστερά) τον κώδικα BCD Χ-3 και στο τμήμα εξόδων (δεξιά) τον κώδικα BCD-2421 (αριστερά ο κώδικας που θα μετατρέψουμε και δεξιά ο κώδικας που θα πάρουμε).

Πίνακας Εισόδων - Εξόδων

	A	B	C	D	B3	B2	B1	B0
0	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	1	0
3	0	1	1	0	0	0	1	1
4	0	1	1	1	0	1	0	0
5	1	0	0	0	1	0	1	1
6	1	0	0	1	1	1	0	0
7	1	0	1	0	1	1	0	1
8	1	0	1	1	1	1	1	0
9	1	1	0	0	1	1	1	1

♦Θεωρούμε κάθε ψηφίο του κώδικα εξόδου (BCD-2421) **Bi**, σαν μια συνάρτηση των μεταβλητών A,B,C,D του κώδικα υπερβολής κατά 3 και σημειώνουμε τον **χάρτη κλειδί** τον οποίο αριθμούμε με βάση αυτόν τον κώδικα, **σημειώνουμε** δε σαν αδιάφορους όρους όλους τους συνδυασμούς που δεν χρησιμοποιούμε

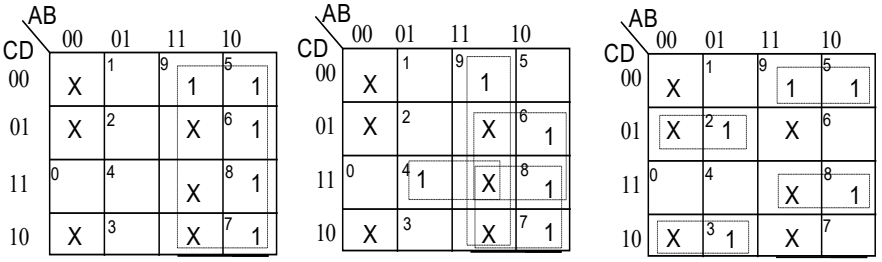


Χάρτης Κλειδί

(κενά τετράγωνα χωρίς αρίθμηση).

Αντιγράφουμε τον χάρτη κλειδί 4 φορές (1 για κάθε έξοδο), από τους οποίους έχουμε τις απλοποιημένες εκφράσεις για τις λογικές συναρτήσεις των εξόδων **Bi** του κωδικοποιητή.

Οι χάρτες απλοποίησης των λογικών συναρτήσεων φαίνονται στην επόμενη σελίδα.



♦ Από τους Καρνώ προκύπτουν οι τελικές απλοποιημένες λογικές συναρτήσεις των εξόδων του κυκλώματος κωδικοποίησης που είναι:

$1^{η}$ B3=A $2^{η}$ B2=AB+AD+AC+BCD $3^{η}$ B1=A⊕(C⊕D) $4^{η}$ B0=$\overline{D}$

♦Το κύκλωμα του κωδικοποιητή σχεδιάζεται με την βοήθεια λογικής NAND και την χρήση πυλών XOR (όπου απαιτούνται). Αν τώρα θέλουμε το αντίστροφο κύκλωμα μετατροπής, του κώδικα BCD-2421 στον κώδικα υπερβολής κατά 3, αλλάζουμε την θέση των κωδίκων στον πίνακα και ακολουθούμε την παραπάνω διαδικασία.

7.3.3 Κύκλωμα Μετατροπής του κώδικα BCD-2421 στον κώδικα υπερβολής Χ-3

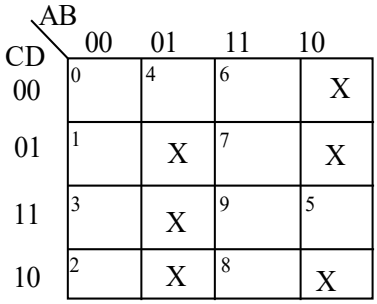
Το τρίτο παράδειγμα αφορά την μετατροπή του κώδικα BCD-2421 στον κώδικα υπερβολής κατά 3 τον BCD Χ-3.

α) Γράφουμε τους κώδικες στον πίνακα εισόδων-εξόδων. Στο αριστερό τμήμα τον κώδικα BCD-2421 και στο δεξιό τμήμα τον κώδικα BCD Χ-3.

Πίνακας Εισόδων - Εξόδων

	A	B	C	D	B3	B2	B1	B0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	1	0	1	1	1	0	0	0
6	1	1	0	0	1	0	0	1
7	1	1	0	1	1	0	1	0
8	1	1	1	0	1	0	1	1
9	1	1	1	1	1	1	0	0

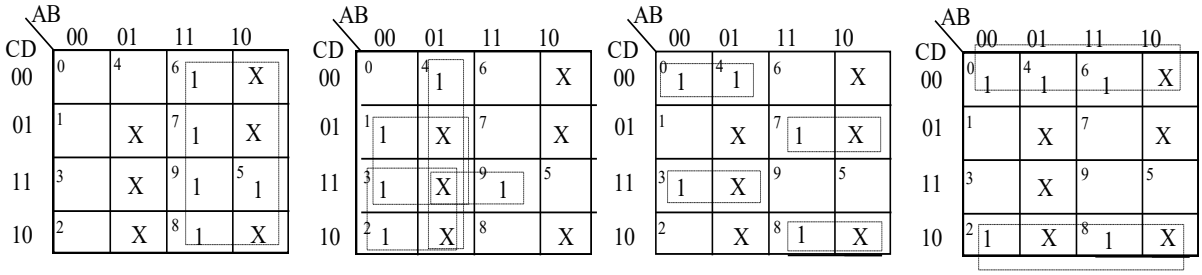
β) Θεωρούμε κάθε ψηφίο του κώδικα εξόδου (BCD Χ-3) **Bi**, σαν μια συνάρτηση των μεταβλητών A,B,C,D του κώδικα BCD-2421 και γράφουμε τον **χάρτη κλειδί,**



Χάρτης Κλειδί

τον οποίο αριθμούμε σύμφωνα με τον κώδικα που θα μετατρέψουμε, BCD-2421 και σημειώνουμε σαν αδιάφορους όρους τους συνδυασμούς που δεν χρησιμοποιούμε.

β1) Αντιγράφουμε τον χάρτη κλειδί 4 φορές, 1 για κάθε έξοδο, από τους οποίους προκύπτουν οι τελικές Α.Λ.Σ. των εξόδων Bi του κωδικοποιητή.



γ) Από τους επιμέρους χάρτες με την γνωστή διαδικασία, προκύπτουν οι τελικές απλοποιημένες λογικές συναρτήσεις του κυκλώματος κωδικοποίησης που είναι:

1η B3=A **2η** B2= $\overline{A}B+\overline{A}D+\overline{A}C+BCD$ **3η** B1= $\overline{A}.\overline{C}.\overline{D}+\overline{A}CD+A\overline{C}D+AC\overline{D}$ **4η** B0= $\overline{D}$

δ) Το κύκλωμα του κωδικοποιητή σχεδιάζεται με λογική NAND (και την χρήση πυλών XOR αν προκύπτουν αντίστοιχες συναρτήσεις άμεσα ή μετά από τροποποίηση).

7.3.4 Κύκλωμα Μετατροπής του κώδικα Gray στον κώδικα BCD-5211

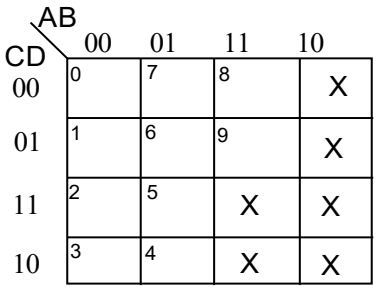
Το τέταρτο παράδειγμα αφορά την μετατροπή των ψηφίων του δεκαδικού συστήματος με τον κώδικα Gray στον κώδικα BCD-5211.

α) Γράφουμε τους κώδικες στον πίνακα εισόδων-εξόδων. Στο αριστερό τμήμα τον κώδικα Gray και στο δεξιό τον κώδικα BCD-5211.

Πίνακας Εισόδων - Εξόδων

	A	B	C	D	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	1	0	0
3	0	0	1	0	0	1	0	1
4	0	1	1	0	0	1	1	1
5	0	1	1	1	1	0	0	0
6	0	1	0	1	1	0	1	0
7	0	1	0	0	1	0	1	1
8	1	1	0	0	1	1	1	0
9	1	1	0	1	1	1	1	1

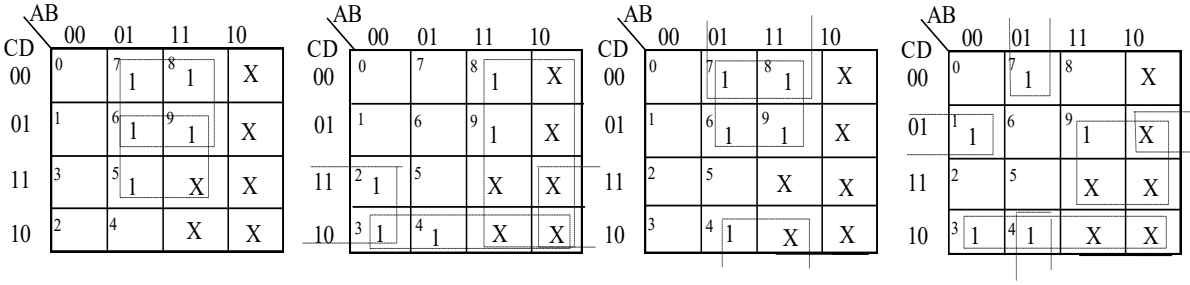
β) Θεωρούμε κάθε ψηφίο του κώδικα εξόδου (BCD-5211) **Bi**, σαν μια συνάρτηση των μεταβλητών A,B,C,D του κώδικα Gray και σημειώνουμε τον **χάρτη κλειδί**



Χάρτης Κλειδί

τον οποίο αριθμούμε σύμφωνα με αυτό τον κώδικα και σημειώνουμε τους αδιάφορους όρους (κενά τετράγωνα χωρίς αρίθμηση).

β1) Αντιγράφουμε 4 φορές τον χάρτη κλειδί, 1 για κάθε έξοδο, όπως αυτοί φαίνονται παρακάτω, από τους οποίους θα πάρουμε τις τελικές εκφράσεις των συναρτήσεων των εξόδων Bi του κωδικοποιητή.



γ) Από τους χάρτες Καρνώ προκύπτουν οι τελικές απλοποιημένες λογικές συναρτήσεις Α.Λ.Σ. του κυκλώματος κωδικοποίησης,

οι οποίες είναι: $B3 = B\overline{C} + BD$ $\qquad B2 = A + C\overline{D} + \overline{B}C$

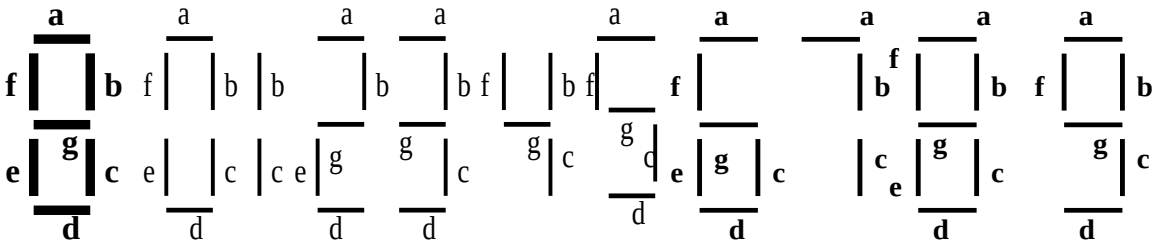
$B1 = B.\overline{C} + B\overline{D}$ $\qquad B0 = C\overline{D} + AD + \overline{A}B\overline{D} + \overline{B}.\overline{C}D$

δ) Το κύκλωμα του κωδικοποιητή σχεδιάζεται με τη λογική σχεδίασης NAND (και χρήση πυλών XOR αν προκύπτουν αντίστοιχες συναρτήσεις άμεσα ή μετά από τροποποίηση).

7.3.5 Μετατροπή του κώδικα BCD-8421 σε κώδικα απεικόνισης 7 τμημάτων

Στην απεικόνιση αριθμών στις αριθμομηχανές (Calculators) απαιτείται ένας κώδικας 7-τμημάτων (Seven Segment Display Code), ο οποίος εμφανίζει τον δυαδικό αριθμό με το αντίστοιχο δεκαδικό ψηφίο πάνω στην οθόνη. Το κάθε δεκαδικό ψηφίο μπορεί να σχηματιστεί με κατάλληλο συνδυασμό μερικών από τα επτά τμήματα μιας ειδικής λυχνίας. Τα τμήματα αντιστοιχούν σε φωτοδιόδους LED (Liguid Crystal

Display-LCD). Η διάταξη των επτά τμημάτων a, b, c, d, e, f, g για τον σχηματισμό του ψηφίου φαίνεται στο σχήμα.



Τα δέκα ψηφία σχηματίζονται με τους κατάλληλους συνδυασμούς των LED's.

Το ζητούμενο κύκλωμα θα έχει τέσσερις εισόδους και επτά εξόδους όπου κάθε μια έξοδος θα αντιστοιχεί σε ένα από τα Led τις λυχνίας a,b,c,d,e,f,g .

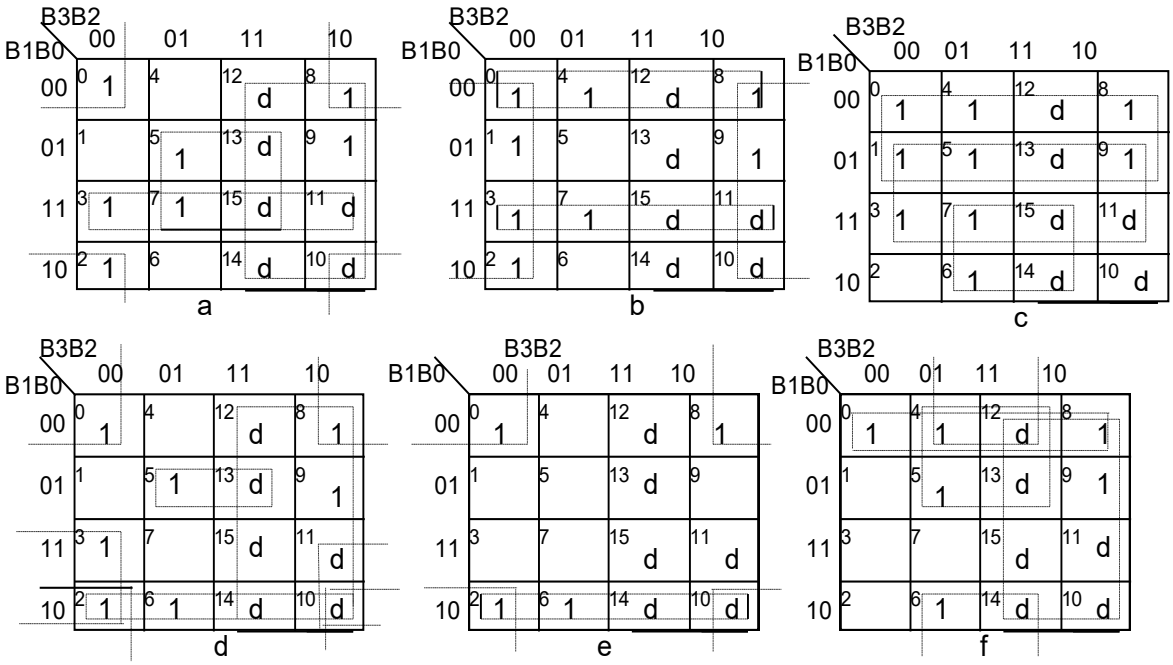
Γράφουμε τους κώδικες στον πίνακα εισόδων-εξόδων. Στο αριστερό τμήμα τον κώδικα BCD-8421 και στο δεξιό τμήμα τα επτά τμήματα της λυχνίας.

Θεωρούμε κάθε τμήμα της λυχνίας των επτά τμημάτων a,b,c,d,e,f,g σαν μια συνάρτηση των μεταβλητών A,B,C,D του κώδικα BCD-8421 και σημειώνουμε τον **χάρτη κλειδί,** τον οποίο αριθμούμε σύμφωνα με αυτό τον κώδικα και σημειώνουμε τους αδιάφορους όρους (τις καταστάσεις από 10 έως 15).

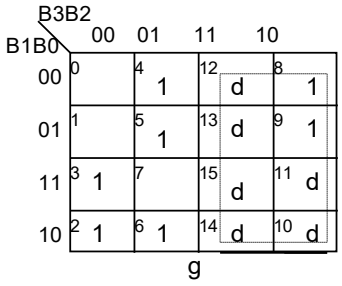
Πίνακας Εισόδων -- Εξόδων

Δεκαδικό	Κώδικας BCD				Είσοδοι Λυχνίας 7 τμημάτων						
Ψηφίο	**8**	**4**	**2**	**1**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Αντιγράφουμε τον χάρτη κλειδί 7 φορές, ένα για κάθε έξοδο, όπως παρακάτω:



Από τους χάρτες καρνώ παίρνουμε τις τελικές εκφράσεις των Α.Λ.Σ. των εξόδων, τα επτά τμήματα της λυχνίας, οι οποίες είναι :



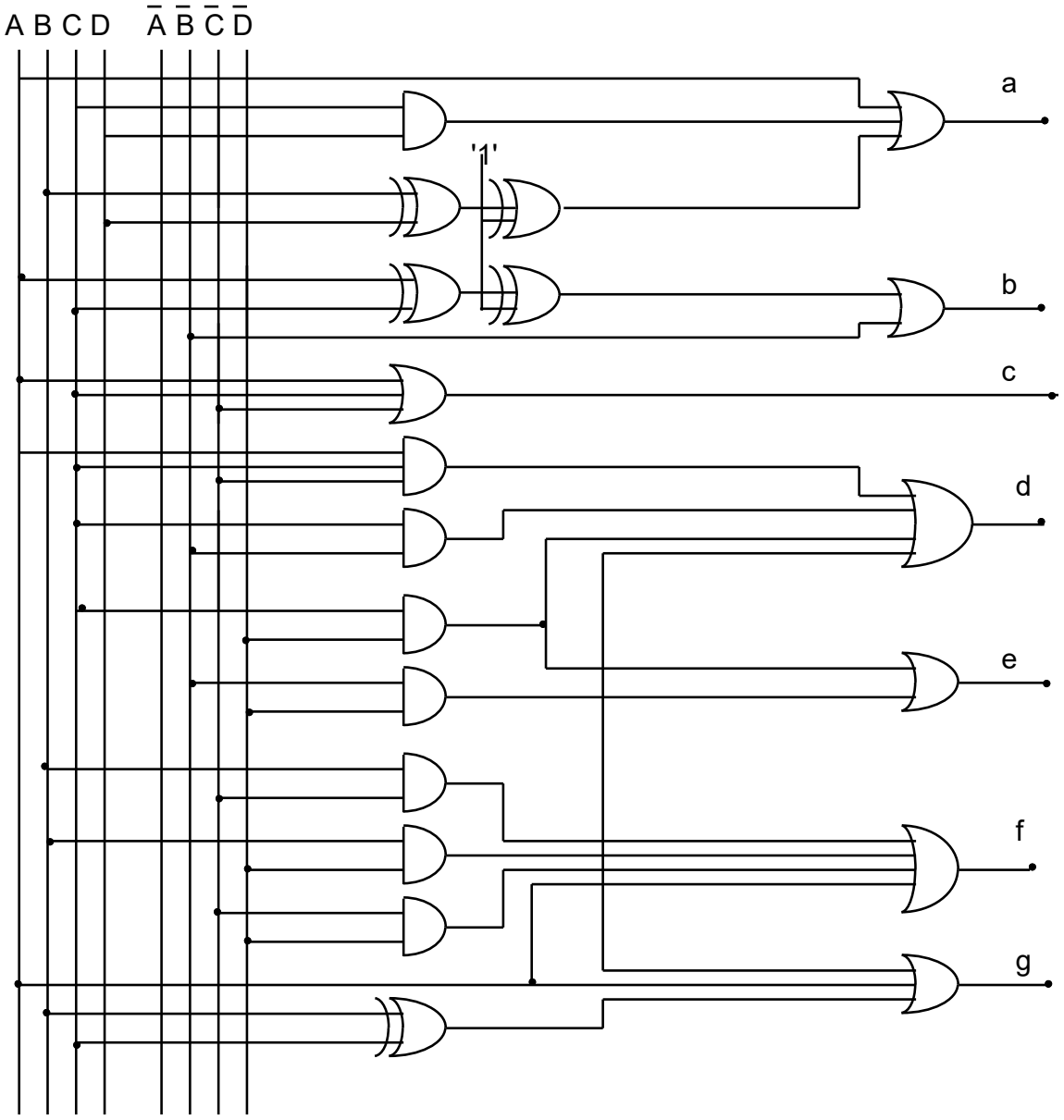
a= $A+CD+\overline{B}.\overline{D}+BD=A+CD+B\otimes D=A+CD+(\overline{B\oplus D})$

b= $\overline{B}+\overline{C}.\overline{D}+CD=\overline{D}+C\otimes D=\overline{D}+(\overline{C\oplus D})$

c= $B+\overline{C}+D$ **d**= $C\overline{D}+\overline{B}C+B\overline{C}D+\overline{B}.\overline{D}$

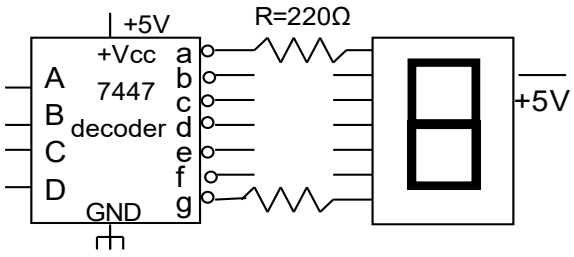
e= $C\overline{D}+\overline{B}.\overline{D}$ **f**= $\overline{C}.\overline{D}+A+B\overline{D}+B\overline{C}$ **g**= $C\overline{D}+\overline{B}C+B\overline{C}+A=A+C\overline{D}+(B\oplus C)$

Από τις τελικές εκφράσεις σχεδιάζεται το κύκλωμα μετατροπής των ψηφίων του δεκαδικού συστήματος στην λυχνία των 7 τμημάτων με πύλες AND, OR, NOT, XOR, όπως αυτό φαίνεται στο σχήμα.



Οι αποκωδικοποιητές BCD σε κώδικα απεικόνισης επτά γραμμών κατασκευάζονται σε ολοκληρωμένα κυκλώματα MSI π.χ 7446A και 7447A της οικογένειας TTL, τα οποία, σε οθόνες πολλών ψηφίων, συνδέοντα μέσω ακροδεκτών RBI-Ripple Blanking Input, έχουν την δυνατότητα να αφαιρούν τα μη σημαντικά μηδενικά μπροστά από τον αριθμό π.χ. αντί 0023 φαίνεται το 23.

Το ίδιο κύκλωμα με δυο IC φαίνεται στο σχήμα.



7.4 Σχεδίαση Κυκλωμάτων Αποκωδικοποιητών (Decoders)

Ο αποκωδικοποιητής είναι συνδυαστικό κύκλωμα το οποίο εκτελεί την αντίστροφη διαδικασία της κωδικοποίησης και μετατροπής από τον **X**-κώδικα στον αρχικό. Είναι ένα κύκλωμα με **n** το πλήθος εισόδων και μέχρι **m** το πλήθος εξόδων όπου $\mathbf{m} \le 2^{\mathbf{n}}$, οι οποίες είναι αμοιβαία αποκλειόμενες αφού από τις οποίες εξόδους **μόνο μία** είναι **ένα-"1"** (οι υπόλοιπες είναι μηδέν), όπως φαίνεται στον πίνακα**.**

Είσοδοι			Έξοδοι							
D2	**D1**	**D0**	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

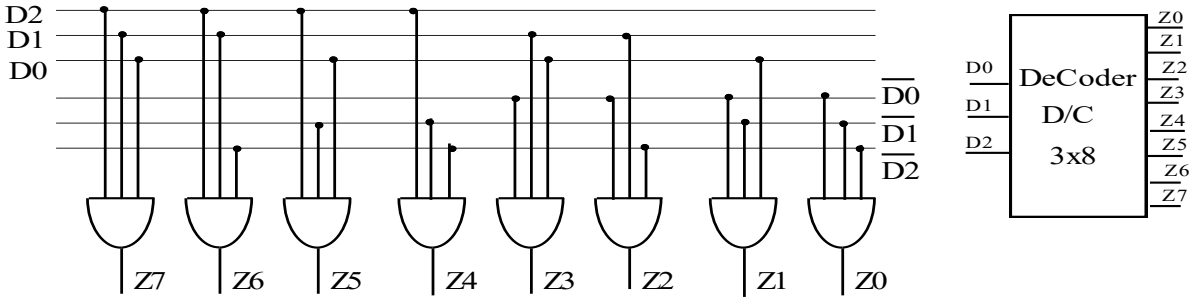
Αν θέλουμε να αποκωδικοποιήσουμε το περιεχόμενο ενός καταχωρητή με 3-FF, ο αποκωδικοποιητής αποτελείται από **n=3** εισόδους και **m=8** $\le 2^{\mathbf{3}}$ (**n** σε **m** ή **3** σε **8**) εξόδους, από τις οποίες **μια μόνο** έξοδος είναι ένα-'**1**' και οι υπόλοιπες είναι μηδέν-'0'. Από τον πίνακα αληθείας έχουμε τις συναρτήσεις εξόδου.

$$Z_0 = \overline{D_2}.\overline{D_1}.\overline{D_0} \quad Z_1 = \overline{D_2}.\overline{D_1}.D_0 \quad Z_2 = \overline{D_2}.D_1.\overline{D_0} \quad Z_3 = \overline{D_2}.D_1D_0$$

$$Z_4 = D_2.\overline{D_1}.\overline{D_0} \quad Z_5 = D_2.\overline{D_1}.D_0 \quad Z_6 = D_2.\overline{D_1}.D_0 \quad Z_7 = D_2.D_1D_0$$

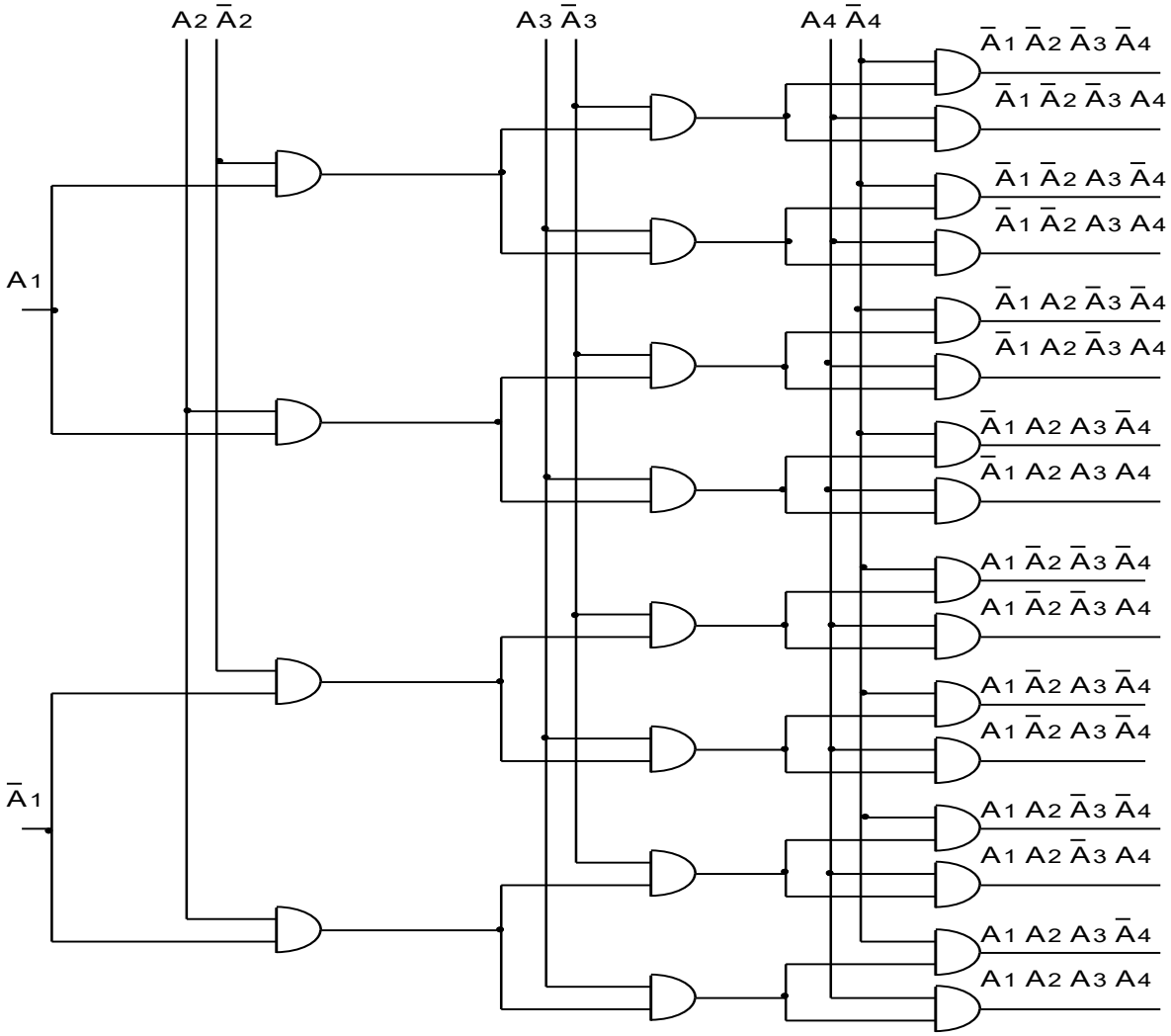
Το συνδυαστικό κύκλωμα προκύπτει από οκτώ πύλες AND 3 εισόδων.

Με την μορφή αυτή το κύκλωμα είναι γνωστό σαν "παράλληλος" αποκωδικοποιητής. Είναι ταχύτατος στην μεταφορά δεδομένων, όμως μειονεκτεί στην χρήση πολλών πυλών στην είσοδο.



Υπάρχουν και άλλα κυκλώματα αποκωδικοποίησης ένα από αυτά είναι ο ισοσταθμισμένος αποκωδικοποιητής ή ο λεγόμενος αποκωδικοποιητής δένδρου (Tree Decoder), ο οποίος χρησιμοποιεί πάντοτε πύλες δύο μόνο εισόδων.

Το κύκλωμα ενός τέτοιου αποκωδικοποιητή με 4 εισόδους και 16 εξόδους 2^4=16 φαίνεται στο σχήμα.



Σε κατασκευές αποκωδικοποιητών σε IC κυκλώματα, τα συμπληρώματα των εισόδων σχηματίζονται συνήθως μέσα στο ολοκληρωμένο κύκλωμα π.χ TTL 7442.

7.5 Χρήση Αποκωδικοποιητή στην σχεδίαση κυκλωμάτων

7.5.1 Εισαγωγή

Μια συνάρτηση **n** μεταβλητών εκφράζεται με 2^n ελαχιστόρους σε μορφή Α.Γ. Ένας αποκωδικοποιητής (**n**x**m**) δίδει στην έξοδό του 2^n=m συνδυασμούς των **n** γραμμών εισόδου. Κατά συνέπεια μπορούμε να χρησιμοποιήσουμε ένα αποκωδικοποιητή 2^n γραμμών με **m** πύλες OR για να πραγματοποιήσουμε μια λογική συνάρτηση σε άθροισμα ελάχιστων όρων.

7.5.2 Πραγματοποίηση συνδυαστικών κυκλωμάτων με αποκωδικοποιητή

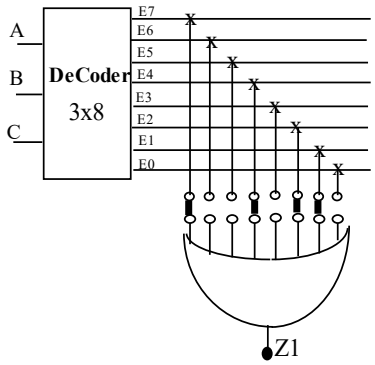
Για να **πραγματοποιήσουμε** μια συνάρτηση με την βοήθεια αποκωδοκοποιητή:

α) Εκφράζουμε τη συνάρτηση σε Α.Γ (Άθροισμα Γινομένων) (ή Α.Ε.Ο Ελάχιστων Όρων) &

β) Ενώνουμε τις αντίστοιχες συνδέσεις

Π.χ. Να υλοποιηθεί με D/C η λογική συνάρτηση

$$Z = \overline{A}.\overline{B}.C + A.B.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C}$$



α) Εκφράζουμε τη Λ.Σ. στη μορφή Α.Γ. και έχουμε

$Z=\Sigma(1,2,4,7)$

β) Αφήνουμε τους συνδέσμους E_7,E_4,E_2,E_1 οπότε η έξοδος της πύλης OR θα είναι η ζητούμενη έκφραση Z.

7.6 Χρήση μνήμης ROM στην σχεδίαση κυκλωμάτων

7.6.1 Εισαγωγή

Η ROM (Read Only Memory) είναι μια διάταξη που περιλαμβάνει ένα αποκωδικοποιητή και πύλες OR, ορίζεται δε από το πλήθος των γραμμών **εισόδου** και **εξόδου** από τη σχέση $\mathbf{2^n xm}$ όπου **n**=αριθμός εισόδων και **m**=αριθμός εξόδων.

Ο **προγραμματισμός** της ROM είναι στην ουσία ο **τρόπος καθορισμού** των **συνδέσμων** των **πυλών OR των εξόδων** με τις **εξόδους** του αποκωδικοποιητή.

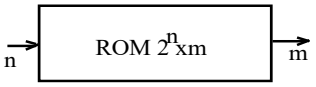
Μια ROM 16x4=$\mathbf{2^4x4}$ σημαίνει ότι έχει **n**=4 εισόδους και **m**=4 εξόδους.

Κάθε ένας συνδυασμός των 4 ψηφίων της εισόδου λέγεται διεύθυνση (address) και κάθε συνδυασμός ψηφίων στις 4 γραμμές εξόδου λέγεται λέξη (word).

Οπότε μια διεύθυνση είναι στην ουσία ένας δυαδικός αριθμός που αναλογεί σε ένα από τους ελάχιστους $\mathbf{2^n}$ όρους που μπορούν να σχηματιστούν από το πλήθος **n** των μεταβλητών, έτσι για **n** μεταβλητές μπορούμε να έχουμε $\mathbf{2^n}$ διαφορετικές διευθύνσεις. Κάθε λέξη στην έξοδο τώρα επιλέγεται από μια και μοναδική διεύθυνση, οπότε έχουμε συνολικά $\mathbf{2^n}$ διαφορετικές λέξεις αποθηκευμένες στην μνήμη.

Κάθε μια λέξη στην έξοδο επιλέγεται από μια και μοναδική διεύθυνση, οπότε έχουμε συνολικά $\mathbf{2^n}$ διαφορετικές λέξεις αποθηκευμένες στην μνήμη μας.

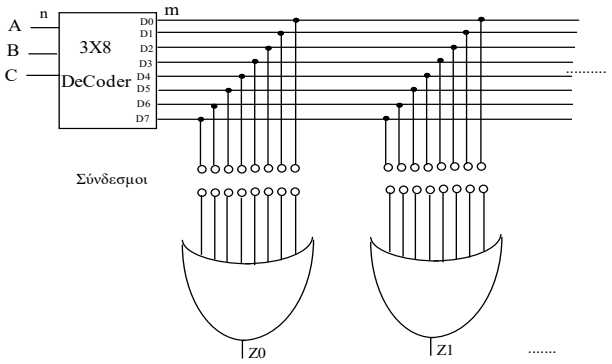
Π.χ. Μια ROM $\mathbf{2^3x4=}$8x4 αποτελείται από 8 λέξεις των 4 ψηφίων.



Έχουμε δηλαδή **3** γραμμές εισόδου και **8** διαφορετικές λέξεις αποθηκευμένες στη μνήμη.

Στη διεύθυνση **000** επιλέγουμε τη λέξη **0** να εμφανίζεται στην έξοδο και στη διεύθυνση **111** επιλέγουμε τη λέξη **7**.

Η διάταξη της ROM φαίνεται στο σχήμα.



7.6.2 Πραγματοποίηση Λογικών Συναρτήσεων με μνήμη ROM.

Η ROM πραγματοποιεί μια λογική συνάρτηση σε δυο επίπεδα σχεδίασης.

Π.χ Να πραγματοποιηθούν με ROM οι λογικές συναρτήσεις $Z_0=\Sigma(1,2,4,7)$ και $Z_1=\Sigma(3,5,6,7)$.

α) Γράφουμε τον πίνακα καταστάσεων και αριθμούμε όλους τους δυνατούς συνδυασμούς των τριών μεταβλητών.

α/α	A	B	C	Z_0	Z_1
0	0	0	0	**0**	**0**
1	0	0	1	**1**	**0**
2	0	1	0	**1**	**0**
3	0	1	1	**0**	**1**
4	1	0	0	**1**	**0**
5	1	0	1	**0**	**1**
6	1	1	0	**0**	**1**
7	1	1	1	**1**	**1**

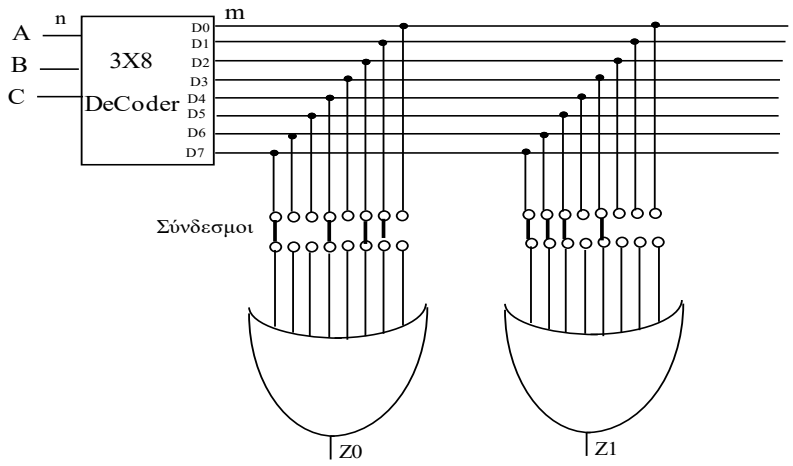
β) Από την εσωτερική οργάνωση της ROM προκύπτει ότι κάθε έξοδος είναι το άθροισμα των ελάχιστων όρων (Α.Γ) των **n**-μεταβλητών της εισόδου.

γ) **Καταστρέφοντας** τους συνδέσμους που δεν χρειάζονται έχουμε τις εξόδους Z_0,Z_1.

Στο παράδειγμά μας απαιτείται μια **ελάχιστη ROM** 2^3x2=8x2 με **n=3** εισόδους και **m=2** εξόδους τις Z_0,Z_1.

Από τον πίνακα καταστάσεων προκύπτει ότι πρέπει να **αφήσουμε** τους συνδέσμους με «βάρος» 1,2,4,7 για την Z_0 και τους συνδέσμους με «βάρος» 3,5,6,7 για την Z_1.

Οι λογικές συναρτήσεις που πραγματοποιήσαμε με την ROM είναι στην πραγματικότητα οι έξοδοι ενός πλήρους αθροιστή όπου **A** είναι ο πρώτος αριθμός, **B** είναι ο δεύτερος αριθμός που θα προσθέσουμε και **C** είναι το κρατούμενο από προηγούμενη πράξη. Το κύκλωμα με τις απαιτούμενες συνδέσεις φαίνεται στην επόμενη σελίδα.



7.6.3 Τύποι μνήμης ROM

Οι συνδέσεις μιας ROM προγραμματίζονται με ειδική μέθοδο (mask programming) και για μικρό αριθμό μνημών δεν συμφέρει από την άποψη της κατασκευής. Υπάρχουν άλλα είδη μνήμης ROM που είναι περισσότερο κατάλληλα για μικροεφαρμογές. Αυτά είναι:

Α) **PROM**. Προγραμματιζόμενη ROM.

Με ειδικό μηχάνημα, εφαρμόζοντας ισχυρούς παλμούς ρεύματος, **καταστρέφουμε** τους μη απαραίτητους συνδέσμους οπότε προκύπτει το ζητούμενο κύκλωμα.

Β) **EPROM**. Σβηνόμενη (Erasable) PROM.

Σε αυτήν έχουμε τη δυνατότητα να σβήσουμε το περιεχόμενο με μια συγκεκριμένη διαδικασία και να την επαναπρογραμματίζουμε.

Στην EPROM. μπορούμε να επέμβουμε και με ηλεκτρικό τρόπο ώστε να σβήσουμε ο περιεχόμενο οπότε έχουμε την EEPROM Electrical erasable PROM.

7.7 Σχεδίαση κυκλωμάτων σύγκρισης (Συγκριτές - Comparators)

Ο συγκριτής είναι ένα συνδυαστικό κύκλωμα απαραίτητο στοιχείο σε άλλα περισσότερο σύνθετα κυκλώματα. Συγκρίνει δύο μονοψήφιους δυαδικούς αριθμούς, οι οποίοι φθάνουν στην είσοδό του από ένα καταχωρητή **ή** ένα άλλο κύκλωμα **ή** άμεσα από εξωτερικά αισθητήρια. Οι έξοδοί του είναι τρείς-3 και αντιστοιχούν στις τρεις δυνατές περιπτώσεις σύγκρισης των δύο δυαδικών ψηφίων A,B δηλαδή 1) A<B 2) A=B και 3) A>B. Τα A,B είναι οι δυαδικοί αριθμοί **A** ο πρώτος και **B** ο δεύτερος.

Ας δούμε αναλυτικά όλη τη διαδικασία για την σχεδίαση ενός συγκριτή δύο μονοψήφιων δυαδικών αριθμών A,B.

Γράφουμε τον πίνακα καταστάσεων από τον οποίο έχουμε τις λογικές συναρτήσεις.

Είσοδοι		**Έξοδοι**				
A	**B**	Z_L	Z_E	Z_G	Z_{LE}	Z_{GE}
0	0	0	1	0	1	1
0	1	1	0	0	1	0
1	0	0	0	1	0	1
1	1	0	1	0	1	1

Οπότε οι λογικές συναρτήσεις είναι:

α) Το A να είναι μικρότερο από το B (A<B) $Z_L = \overline{A}B$ (1)

β) Το A να είναι ίσο με το B (A=B) $Z_E = \overline{A}.\overline{B} + AB = \overline{(A \oplus B)} = A \otimes B$ (2)

γ) Το A να είναι μεγαλύτερο από το B (A>B) $Z_G = A\overline{B}$ (3)

Από τον ίδιο πίνακα αληθείας μπορούμε να πάρουμε άλλες δύο καταστάσεις σύγκρισης για τις περιπτώσεις όπου:

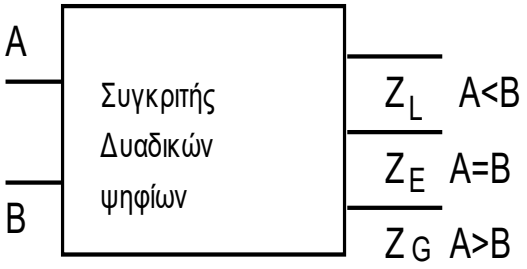
δ) Το A να είναι μικρότερο ή ίσο με το B (A≤B) $Z_{LE} = \overline{A}B + \overline{A}.\overline{B} + AB)$ (4)

ε) Το A να είναι μεγαλύτερο ή ίσο με το B (A≥B) $Z_{GE} = A\overline{B} + \overline{A}.\overline{B} + AB$ (5)

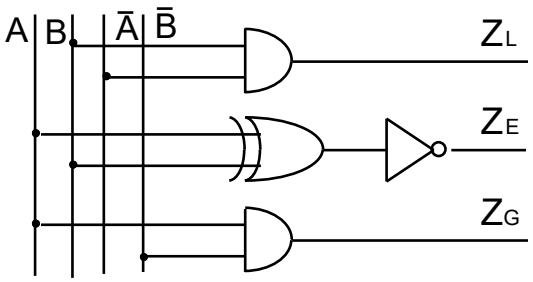
οι οποίες προκύπτουν από την επαλληλία των απλών περιπτώσεων

Το Block διάγραμμα και το αναλυτικό κύκλωμα του συγκριτή φαίνονται στα σχήματα.

α) Block διάγραμμα



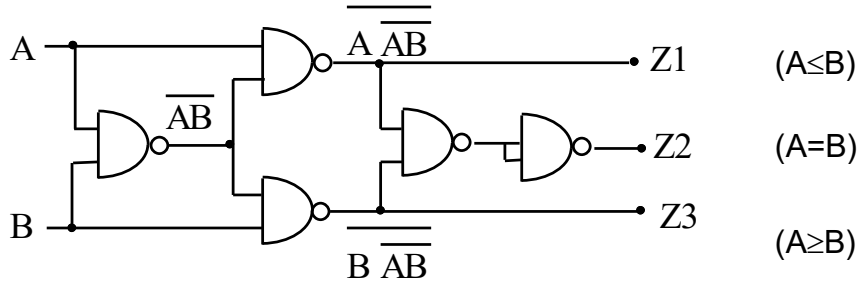
β) Αναλυτικό κύκλωμα



Το κύκλωμα ενός τέτοιου συγκριτή σχεδιάζεται σύμφωνα με τις λογικές συναρτήσεις του όμως είναι αρκετά πολύπλοκο.

Για τον λόγο αυτό προχωρούμε στην ανάλυση των συναρτήσεων, στην παραπάνω τελική μορφή, ώστε με το ισοδύναμο κύκλωμα μιας πύλης XOR, με πύλες NAND, να έχουμε και τις τρεις περιπτώσεις σύγκρισης μικρότερο ή ίσο, ίσο, μεγαλύτερο ή ίσο ($A \leq B$, $A=B$, $A \geq B$).

Το νέο κύκλωμα καθώς και οι αντίστοιχες έξοδοι για τις 3 περιπτώσεις σύγκρισης φαίνονται στο σχήμα.



Η απόδειξη σε κάθε περίπτωση προκύπτει αν κάνουμε τις πράξεις στις περιπτώσεις 2 & 4 & 5 και τις συγκρίνουμε με τις εξόδου του κυκλώματος.

Από την 4 έχουμε : $Z_{LE} = \overline{A}B + \overline{A}.\overline{B} + AB = \overline{A}(B + \overline{B}) + AB = \overline{A} + AB = A(\overline{AB})$ = **Z1**

Από την 2 έχουμε : $Z_E = \overline{A(A\overline{B})}.\overline{B(\overline{AB})} = (\overline{A} + AB)(.\overline{B} + AB) = \overline{A}.\overline{B} + AB = \overline{A \oplus B}$ = **Z2**

Από την 5 έχουμε : $Z_{GE} = A\overline{B} + \overline{A}.\overline{B} + AB = \overline{B}(A + \overline{A}) + AB = \overline{B} + AB = B(\overline{AB})$ = **Z3**

Έτσι με πέντε πύλες NAND έχουμε τον συγκριτή των δύο δυαδικών ψηφίων.

Αν τώρα ότι οι αριθμοί A & B που πρέπει να συγκρίνουμε είναι πολυψήφιοι, δηλαδή αριθμοί με περισσότερα, του ενός, ψηφία π.χ. οι αριθμοί Ai & Bi με $Ai=A_2A_1A_0$ και $Bi=B_2B_1B_0$, τότε στον πίνακα αληθείας θα πρέπει να χρησιμοποιήσουμε συνδυασμούς κάτι το οποίο είναι ασύμφορο. Αν όμως δούμε τα πράγματα με μια λογική τότε για την περίπτωση π.χ. $A_i<B_i$ έχουμε ότι:

Για να είναι $\mathbf{A_i<B_i}$ πρέπει να έχουμε : α. $A_2<B_2$ **ή**

β. $A_2=B_2$ **και** $A_1<B_1$ **ή**

γ. $A_2=B_2$ **και** $A_1=B_1$ **και** $A_0<B_0$, επομένως η έκφραση της συνάρτησης Z για την σύγκριση των αριθμών, δηλαδή η Z_L σχηματίζεται από την **μετατροπή των παραπάνω προτάσεων** με λογική AND-OR και είναι :

$Z_L=(A_2<B_2)+(A_2=B_2).(A_1<B_1)+(A_2=B_2).(A_1=B_1).(A_0<B_0)$

Αντικαθιστούμε τις εκφράσεις με τις σχέσεις που αποδείξαμε στον πίνακα σύγκρισης των δυο μονοψήφιων δυαδικών αριθμών και έχουμε :

$$Z_L = \overline{A}_2B_2 + (\overline{A}_2\overline{B}_2 + A_2B_2)\overline{A}_1B_1 + (\overline{A}_2\overline{B}_2 + A_2B_2)(\overline{A}_1\overline{B}_1 + A_1B_1)\overline{A}_0B_0$$

Με παρόμοια διαδικασία έχουμε :

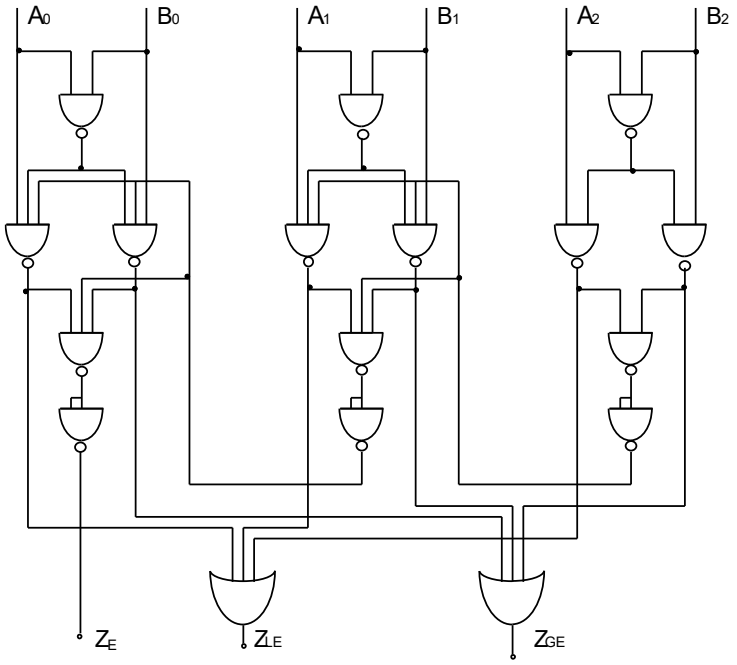
Για την ισότητα των αριθμών **Ai = Bi**

$$Z_E = (\overline{A}_2\overline{B}_2 + A_2B_2)(\overline{A}_1\overline{B}_1 + A_1B_1)(\overline{A}_0\overline{B}_0 + A_0B_0)$$ και

Για την περίπτωση **Ai > Bi**

$$Z_G = A_2\overline{B}_2 + (\overline{A}_2\overline{B}_2 + A_2B_2)A_1\overline{B}_1 + (\overline{A}_2\overline{B}_2 + A_2B_2)(\overline{A}_1\overline{B}_1 + A_1B_1)A_0\overline{B}_0$$

Το κύκλωμα του συγκριτή των δυο τριψήφιων αριθμών θα προκύψει με τη χρήση τριών απλών κυκλωμάτων σύγκρισης τα οποία όμως συνδυαζόμενα μπορούν να εμφανιστούν σε ένα κύκλωμα όπως φαίνεται στο σχήμα.



Οι έξοδοι των βαθμίδων συνδέονται σε πύλες OR για τον σχηματισμό των εκφράσεων A<B, A>B. Τις πύλες OR μπορούμε να τις αντικαταστήσουμε με το ισοδύναμό τους κύκλωμα σχεδιασμένο με πύλες NAND.

7.8 Πραγματοποίηση Συνδυαστικών κυκλωμάτων με IC

7.8.1 Εισαγωγή

Είναι γνωστό ότι υπάρχουν τέσσερα επίπεδα πολυπλοκότητας στην κατασκευή των ολοκληρωμένων κυκλωμάτων **I**ntegrated **C**ircuits α) το επίπεδο μικρής σκάλας ολοκλήρωσης – SSI έως 10 πύλες β) το μεσαίο επίπεδο ολοκλήρωσης -MSI έως 100 πύλες γ) το επίπεδο μεγάλης σκάλας ολοκλήρωσης-VSI έως 10.00 πύλες και τέλος δ) το επίπεδο πολύ μεγάλης σκάλας ολοκλήρωσης -VLSI έως 100.000 πύλες.

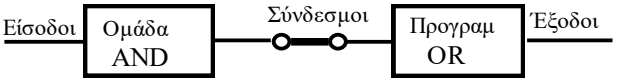
Επομένως η χρήση IC σε εφαρμογές πραγματοποίησης συνδυαστικών κυκλωμάτων διευκολύνει την κατασκευή των κυκλωμάτων. Τα IC MSI χρησιμοποιούνται πάρα πολύ στη σχεδίαση λογικών και ψηφιακών κυκλωμάτων επειδή έχουμε μια σημαντική μείωση στο κόστος κατασκευής σε σχέση με την χρήση IC SSI.

Η πραγματοποίηση συγκεκριμένων κυκλωμάτων με χρήση IC LSI γίνεται με ολοκληρωμένα τα οποία μπορούν να προγραμματιστούν ώστε να δώσουν την απαιτούμενη λογική.

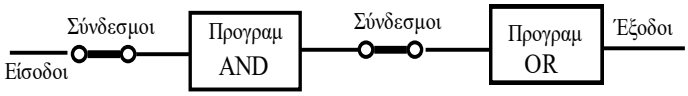
Οι διατάξεις προγραμματιζόμενης λογικής PLD (προγραμματιζόμενες λογικές διατάξεις) είναι ολοκληρωμένα κυκλώματα που περιέχουν πύλες συνδεόμενες με ηλεκτρονικούς συνδέσμους.

Ο λεγόμενος **προγραμματισμός** μιας τέτοιας συσκευής είναι στην πράξη η «**καταστροφή**» των συνδέσμων που δεν χρειάζονται ώστε να πραγματοποιηθεί μια συγκεκριμένη λογική συνάρτηση. Οι πύλες σε μια PLD διατάσσονται σε μια παράταξη – AND και σε μια παράταξη – OR οι οποίες συνδέονται μεταξύ τους και δίδουν μορφή αθροίσματος γινομένων. Ανάλογα με την τοποθέτηση των συνδέσμων AND, OR έχουμε τρεις κατηγορίες διατάξεων PLD:

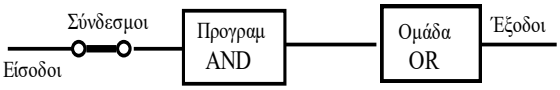
α. **PROM** –Programmable Read Only Memory (Διάταξη προγραμματιζόμενης μνήμης ανάγνωσης μόνο) με σταθερή παράταξη AND και προγραμματιζόμενους συνδέσμους OR.



β. **PLA** -Programmable logic Array (Προγραμματιζόμενη λογική διάταξη) με προγραμματιζόμενους συνδέσμους AND & OR που είναι και **η πιο εύχρηστη** PLD και



γ. PAL Programmable Array logic (Προγραμματιζόμενη διάταξη λογικής) με προγραμματιζόμενους συνδέσμους AND και σταθερή παράταξη OR.



7.8.2 Πραγματοποίηση Συνδυαστικών κυκλωμάτων με ROM

Στην 7.6 είδαμε αναλυτικά την χρήση ROM στην πραγματοποίηση λογικής συνάρτησης. Όμως ένα συνδυαστικό κύκλωμα είναι δυνατόν να έχει και όρους που δεν χρησιμοποιούνται, οι γνωστοί αδιάφοροι όροι.

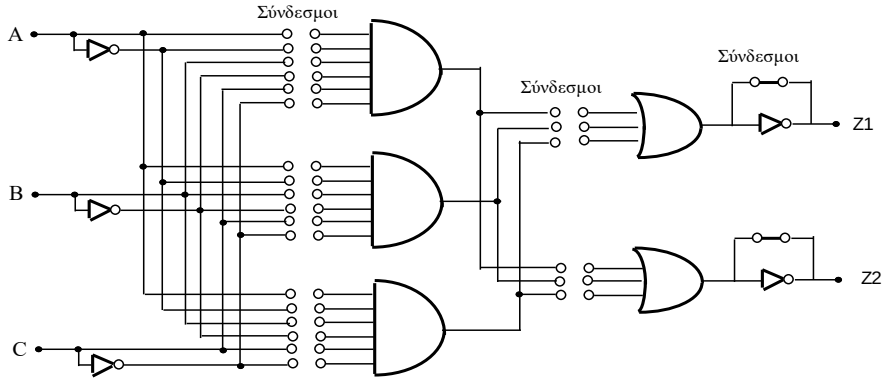
Με τη χρήση ROM οι αδιάφοροι όροι **μεταφράζονται** σε διευθύνσεις εισόδου που **δεν θα συμβούν ποτέ**, δεν χρειάζεται να προγραμματιστούν **τις αφήνουμε στην αρχική** κατάσταση με 0 ή 1, με αποτέλεσμα να μην χρησιμοποιούμε όλες τις θέσεις αποθήκευσης της ROM κάτι που μπορεί να θεωρηθεί σαν κατάχρηση.

Π.χ. διαθέτουμε μια ROM 4Kx4 για την πραγματοποίηση συνάρτησης που απαιτεί 62 ελαχιστόρους. Τότε $2^2x2^{10}=2^{12}$ δηλαδή απαιτούνται 12 είσοδοι και 4 πύλες OR για τις εξόδους. Από τις 4096 διαθέσιμες θέσεις χρησιμοποιούνται μόνο οι 62, οι υπόλοιπες 4034 είναι στην ουσία χαμένες θέσεις. Στις περιπτώσεις αυτές (που οι αδιάφοροι όροι είναι πάρα πολλοί) είναι περισσότερο οικονομικό να χρησιμοποιούμε το chip LSI της προγραμματιζόμενης λογικής διάταξης PLA.

7.8.3 Πραγματοποίηση Συνδυαστικών κυκλωμάτων με PLA

Η PLA.είναι όμοια με την ROM όμως δεν κάνει πλήρη αποκωδικοποίηση και δεν παράγει όλους τους ελαχιστόρους όπως η ROM, αφού ο αποκωδικοποιητής έχει αντικατασταθεί από μια ομάδα πυλών AND **κάθε μια** από τις οποίες **προγραμματίζεται** ώστε να παράγει **ένα μόνο** όρο του γινομένου των μεταβλητών εισόδου.

Το διάγραμμα προγραμματισμού μιας PLA φαίνεται στο σχήμα.



Ο πίνακας προγραμματισμού μιας PLA αποτελείται από 4 στήλες, όπου στην :

1η στήλη σημειώνονται οι διαφορετικοί όροι επομένως το πλήθος των εισόδων της

2η στήλη αριθμούνται όλοι οι όροι του γινομένου.

3η στήλη καθορίζονται οι συνδέσεις ανάμεσα στις εισόδους και τις πύλες AND. Σε κάθε όρο του γινομένου σημειώνεται '**1**' αν εμφανίζεται η κανονική τιμή μιας μεταβλητής, '**0**' αν εμφανίζεται η συμπληρωματική τιμή της & '**-**' αν δεν εμφανίζεται καθόλου μια μεταβλητή.

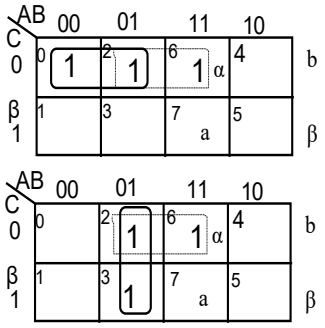
4η στήλη καθορίζονται οι συνδέσεις ανάμεσα στις πύλες AND & OR οπότε κάτω από κάθε έξοδο σημειώνεται το γράμμα **T** (Truth) αν πρέπει να παρακάμψουμε την πύλη NOT στην έξοδο ή το γράμμα **C** (Complementary) αν πρέπει να την χρησιμοποιήσουμε.

Ο **προγραμματισμός** της PLA είναι η **διαδικασία** για τον **καθορισμό** των απαιτούμενων συνδέσεων. Το παρακάτω παράδειγμα είναι απλώς ενδεικτικό για την κατανόηση του προγραμματισμού αφού η χρήση της PLA ενδείκνυται για πάρα πολλές εισόδους και εξόδους.

Π.χ. Να σχεδιαστούν με χρήση PLA οι συναρτήσεις $Z_1=\Sigma(0,2,6)$ & $Z_2=\Sigma(2,3,6)$

α. Πίνακας Καταστάσεων Σημειώνονται οι είσοδοι **n** (A,B,C) και οι έξοδοι Zi

α/α	A	B	C	Z_1	Z_2
0	0	0	0	**1**	**0**
1	0	0	1	**0**	**0**
2	0	1	0	**1**	**1**
3	0	1	1	**0**	**1**
4	1	0	0	**0**	**0**
5	1	0	1	**0**	**0**
6	1	1	0	**1**	**1**
7	1	1	1	**0**	**0**



β. Χάρτες Καρνώ Σημειώνονται οι χάρτες Καρνώ για κάθε μια έξοδο

γ. Α.Λ.Σ Από τους χάρτες εξάγονται οι απλοποιημένες συναρτήσεις των εξόδων Z_1,Z_2. Οι έξοδοι που προκύπτουν είναι $Z1=\overline{A}.\overline{C}+B.\overline{C}$ και $Z2=\overline{A}B+B.\overline{C}$

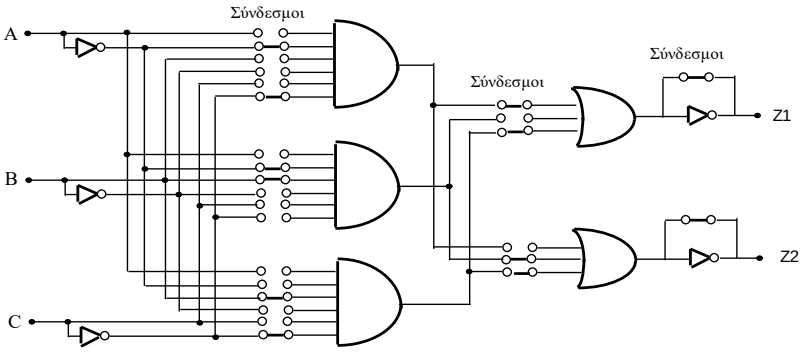
Αν τις συγκρίνουμε προκύπτουν τρεις διαφορετικοί όροι, ο ένας είναι κοινός.

δ. Πίνακες Προγραμματισμού PLA

Σημειώνουμε τους 3 διαφορετικούς όρους των γινομένων με **1,0,-** στις αντίστοιχες μεταβλητές, **1,-** στις εξόδους Z_1,Z_2 και **T** ή **C** αν απαιτείται ή όχι παράλειψη τις πύλης NOT της εξόδου.

α/α	**Όρος γινομένου**	A	B	C	Z_1	Z_2
1	$\overline{A}.\overline{C}$	0	-	0	**1**	-
2	$\overline{A}.B$	0	1	-	-	**1**
3	$B.\overline{C}$	-	1	0	**1**	**1**
					T	**T**

Το κύκλωμα που πραγματοποιεί τη δοθείσα συνάρτηση σχεδιάζεται με βάση τον παραπάνω πίνακα.



7.9 Ασκήσεις

1) Να σχεδιαστεί το συνδυαστικό κύκλωμα 4 γραμμών εισόδου και 4 γραμμών εξόδου (παράσταση δυαδικού σε BCD) που παριστάνουν το συμπλήρωμα προς 9 του ψηφίου εισόδου.

2) Να σχεδιαστεί το συνδυαστικό κύκλωμα το οποίο δέχεται στην είσοδό του ένα αριθμό τριών ψηφίων και δίδει στην έξοδο τον δυαδικό ο οποίος είναι ίσος με το τετράγωνο του αριθμού εισόδου.

3) Να σχεδιαστεί με κωδικοποιητή η συνάρτηση $Z = \overline{A}.\overline{B}.\overline{C} + A.B.C + A.B.\overline{C} + A.\overline{B}.\overline{C}$

4) Να πραγματοποιηθούν με ROM οι λογικές συναρτήσεις $Z_1=\Sigma(0,2,4,7)$, $Z_2=\Sigma(1,3,6,7)$ και

5) Να σχεδιαστεί το κύκλωμα σύγκρισης δυο τετραψήφιων δυαδικών αριθμών Κ,Λ

6) Να σχεδιαστούν με χρήση PLA οι συναρτήσεις $Z_1=\Sigma(0,3,4,7)$ & $Z_2=\Sigma(0,4,5)$