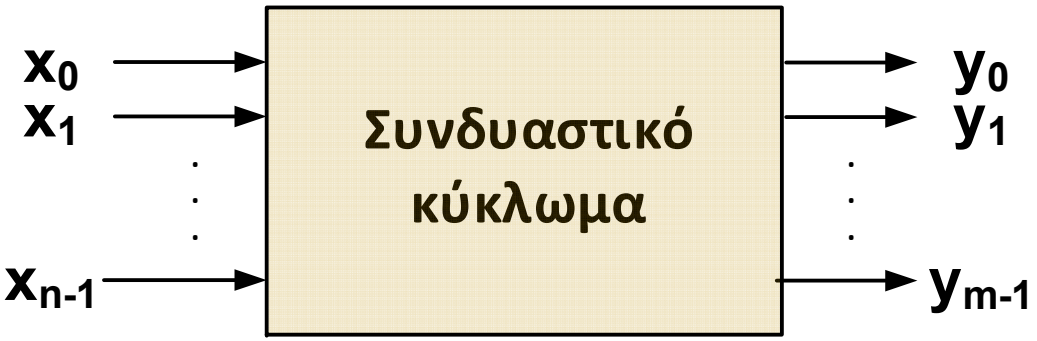
# 5. Σύνθεση & ανάλυση συνδυαστικών κυκλωμάτων και τυποποιημένα συνδυαστικά κυκλώματα


Τμήμα Ηλεκτρολόγων Μηχανικών
& Μηχανικών Υπολογιστών
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ

# Συνδυαστικά κυκλώματα

Τα λογικά κυκλώματα που έχουμε μελετήσει, αναφέρονται ως συνδυαστικά κυκλώματα (combinational circuits).

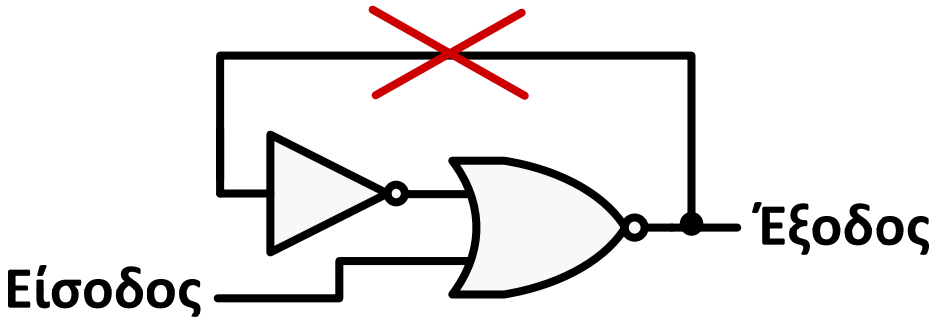
Ο χαρακτηρισμός αυτός αφορά λογικά κυκλώματα των οποίων η λογική τιμή της εξόδου ή των εξόδων, κάθε χρονική στιγμή, εξαρτάται μόνο από τη λογική τιμή των εισόδων που εφαρμόζεται σε αυτά την ίδια χρονική στιγμή.



# Συνδυαστικά κυκλώματα

Οι λογικές πύλες που συνθέτουν ένα συνδυαστικό κύκλωμα διασυνδέονται με τέτοιο τρόπο, ώστε να μη δημιουργείται ανατροφοδότηση.

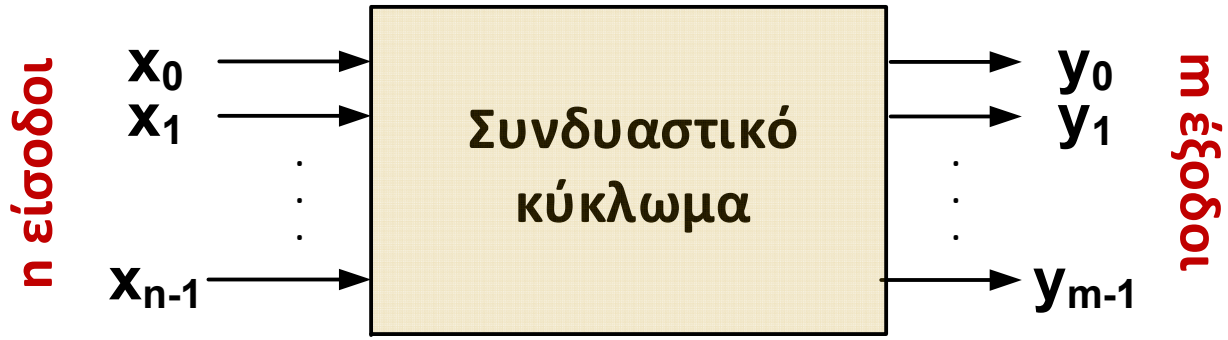
Με την ανατροφοδότηση, δημιουργείται εξάρτηση της εξόδου από λογικές τιμές που λαμβάνει η ανατροφοδοτούμενη είσοδος σε προηγούμενες χρονικές στιγμές, δηλαδή προσδίδεται μνήμη στο κύκλωμα, κάτι το οποίο είναι χαρακτηριστικό των ακολουθιακών κυκλωμάτων (sequential circuits).



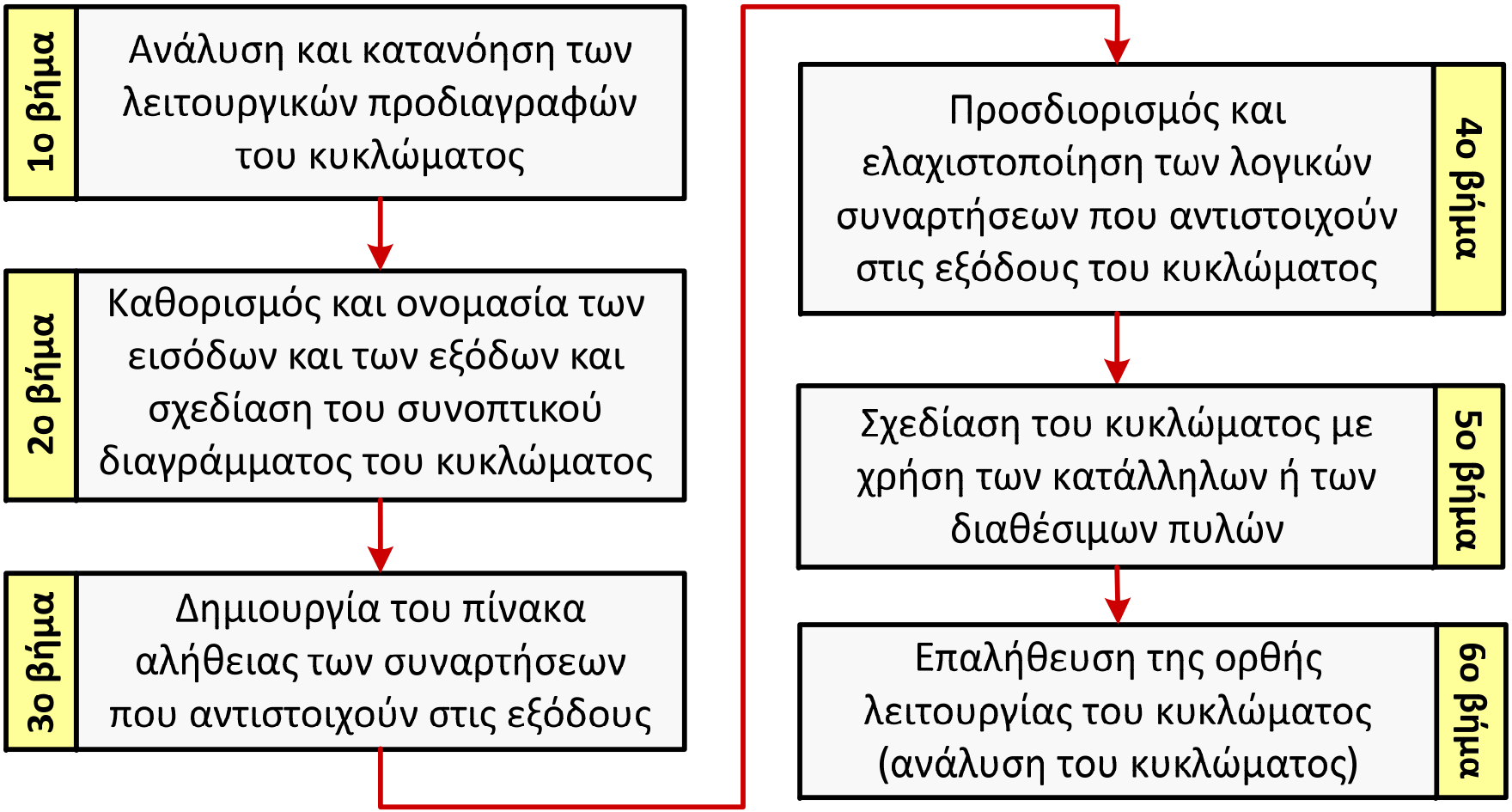
# Συνδυαστικά κυκλώματα

Η λειτουργία ενός συνδυαστικού κυκλώματος μπορεί να περιγραφεί με μοναδικό τρόπο από έναν πίνακα αλήθειας με $2^n$ γραμμές (που αντιστοιχούν στους δυνατούς συνδυασμούς λογικών τιμών των μεταβλητών εισόδου) και n + m στήλες (που αντιστοιχούν στο πλήθος των μεταβλητών εισόδου και εξόδου).

Επίσης, η λειτουργία του μπορεί να περιγραφεί από m λογικές συναρτήσεις, μία για κάθε μεταβλητή εξόδου.



# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων



# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

**Παράδειγμα 1**: Σύνθεση συνδυαστικού κυκλώματος που υλοποιεί τη συνάρτηση πλειοψηφίας 3 εισόδων α) με πύλες NOT, AND, OR και β) μόνο με πύλες NAND.

Ως συνάρτηση πλειοψηφίας 3 εισόδων ορίζεται η λογική συνάρτηση που αναγνωρίζει πότε μεταξύ 3 εισόδων πλειοψηφούν οι μονάδες έναντι των μηδενικών.

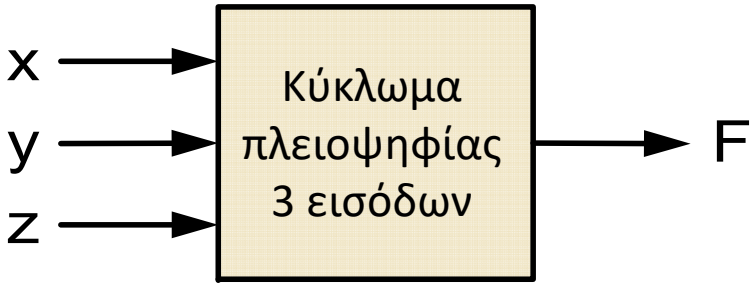
Από την περιγραφή της λειτουργίας του κυκλώματος είναι προφανές ότι το ζητούμενο συνδυαστικό κύκλωμα διαθέτει 3 εισόδους (x, y, z).

Επιπλέον, διαθέτει μία έξοδο (F), αφού το πλήθος των εισόδων είναι περιττός αριθμός (3) και επομένως θα έχουμε περισσότερες μονάδες ή περισσότερα μηδενικά (αποκλείεται να έχουμε ίδιο 1 και 0).

Η έξοδος λαμβάνει τιμή 1, όταν έχουμε περισσότερες μονάδες από μηδενικά, διαφορετικά λαμβάνει τιμή 0.

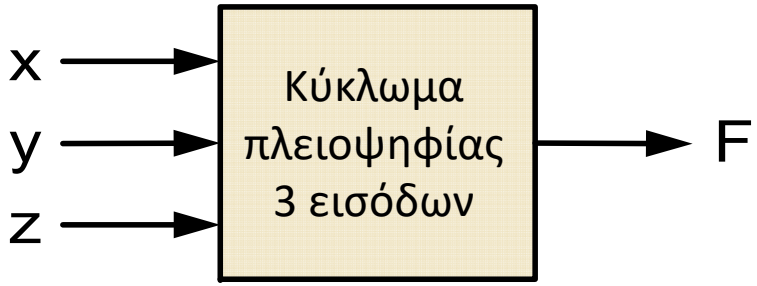
# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:



# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:

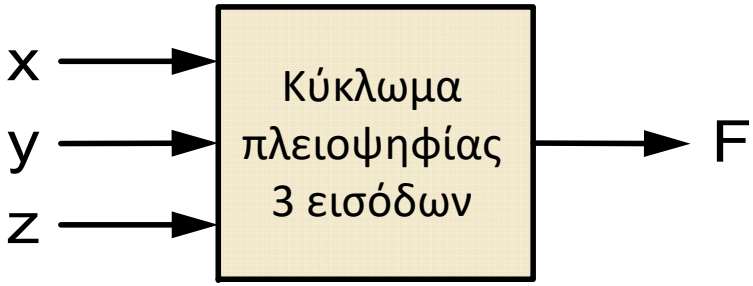



Πίνακας αλήθειας :

| x | y | z | F(x, y, z) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

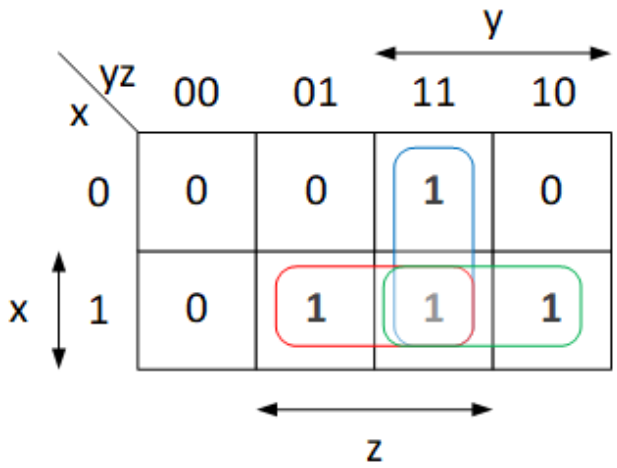
# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:




Πίνακας αλήθειας και χάρτης Karnaugh:

| x | y | z | F(x, y, z) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

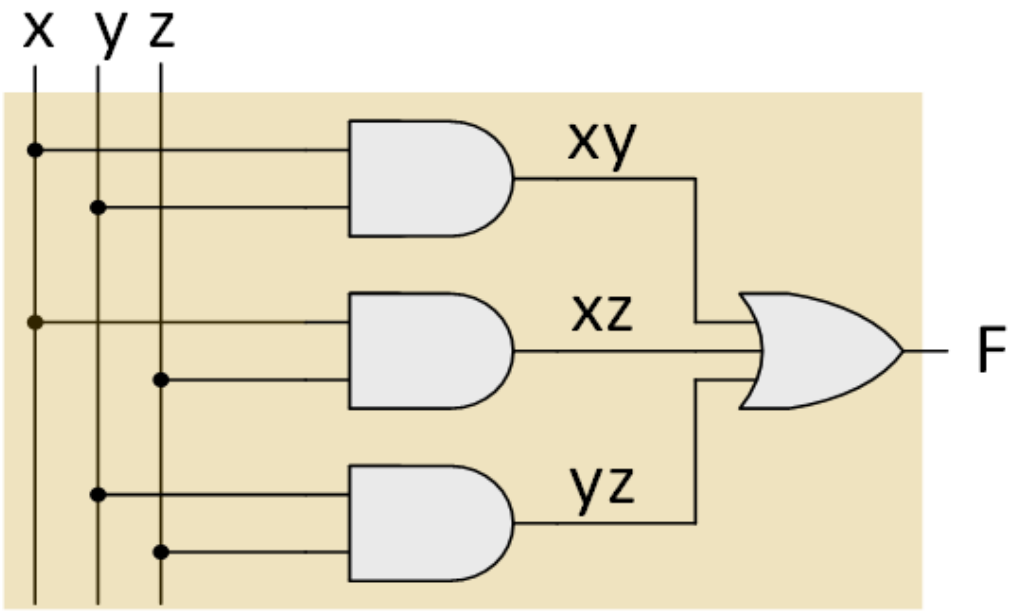



$F = \Sigma(3,5,6,7) = xy + xz + yz$

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

$F = \Sigma(3,5,6,7) = xy + xz + yz$

Σχεδίαση λογικού διαγράμματος του κυκλώματος με πύλες NOT, AND, OR:

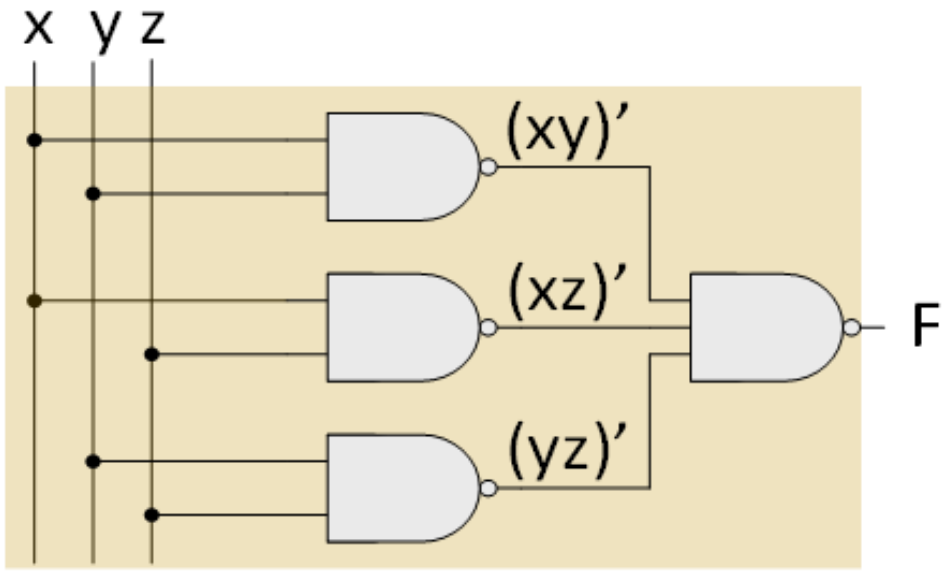


# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση λογικού διαγράμματος του κυκλώματος μόνο με πύλες NAND:

Εφαρμόζουμε κατά σειρά τα θεωρήματα διπλής άρνησης και De Morgan στην ελαχιστοποιημένη μορφή αθροίσματος γινομένων της συνάρτησης εξόδου:

$$F = [(xy + xz + yz)']' = [(xy)'(xz)'(yz)']'$$



# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

**Παράδειγμα 2**: Σύνθεση συνδυαστικού κυκλώματος που να ανιχνεύει την εσφαλμένη λειτουργία ενός φωτεινού σηματοδότη κυκλοφορίας, με πύλες NOT, AND και OR.

Σε κανονική λειτουργία, μόνο μια από τις 3 λάμπες του σηματοδότη (πράσινη, πορτοκαλί, κόκκινη) είναι αναμμένη και οποιοσδήποτε άλλος συνδυασμός οδηγεί σε εσφαλμένη λειτουργία.

Από την παραπάνω ανάλυση της λειτουργίας του σηματοδότη προκύπτει ότι το συνδυαστικό κύκλωμα διαθέτει 3 εισόδους (A, B, C), μία από κάθε λάμπα (πράσινη, πορτοκαλί, κόκκινη) και μία έξοδο (F), η οποία όταν ανιχνεύεται εσφαλμένη λειτουργία λαμβάνει λογική τιμή 1, ενώ όταν η λειτουργία του σηματοδότη είναι η προβλεπόμενη, λαμβάνει λογική τιμή 0.

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:


Πίνακας αλήθειας :

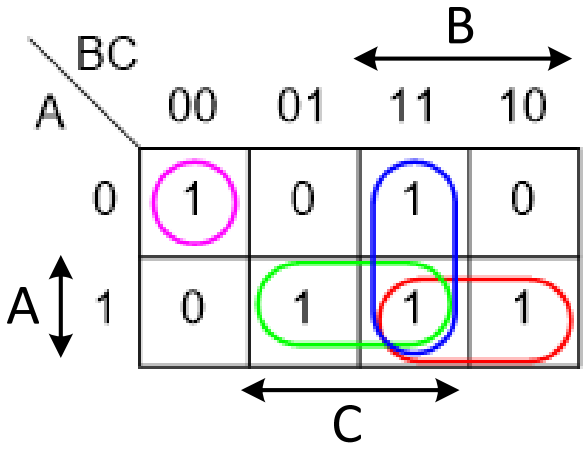
| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | **1** |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | **1** |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | **1** |
| 1 | 1 | 0 | **1** |
| 1 | 1 | 1 | **1** |

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:


Πίνακας αλήθειας και χάρτης Karnaugh:

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | **1** |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | **1** |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | **1** |
| 1 | 1 | 0 | **1** |
| 1 | 1 | 1 | **1** |

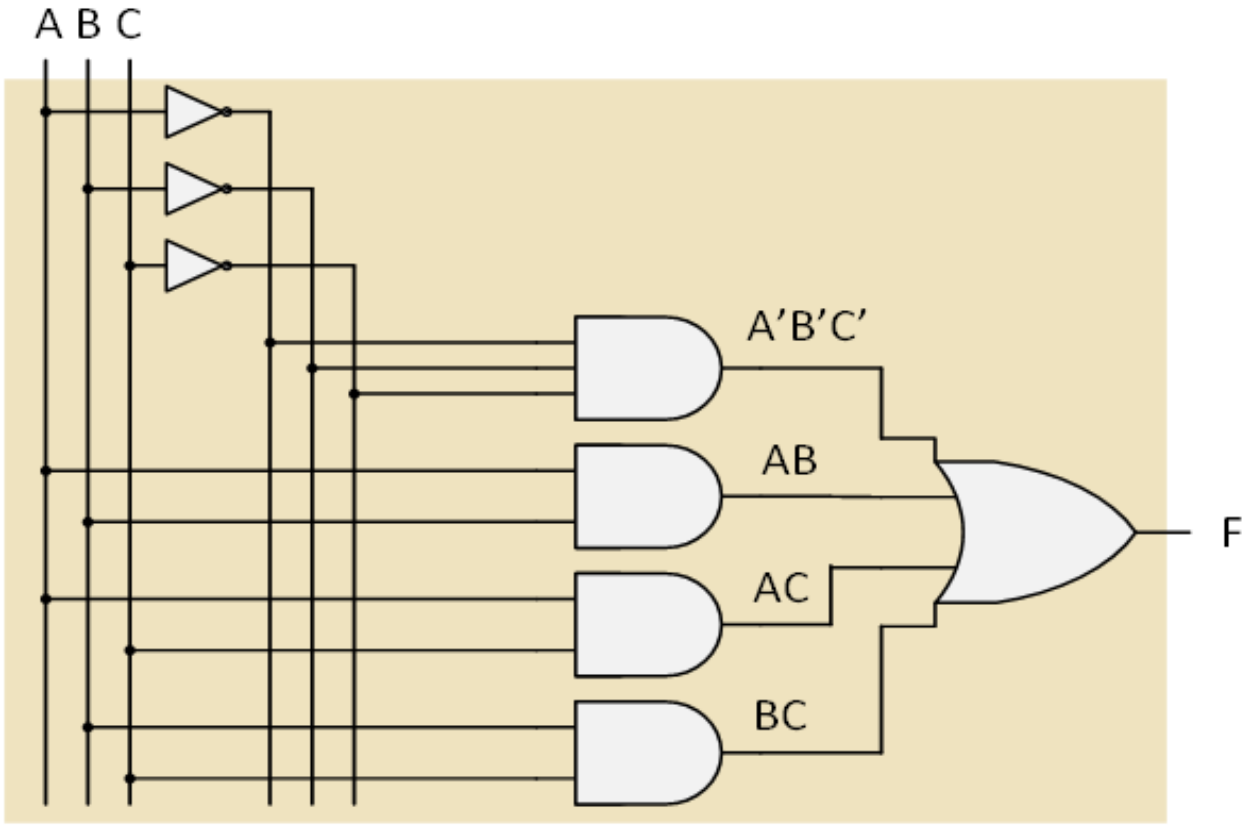



$F = \Sigma(1,3,5,6,7) = A'B'C' + AB + AC + BC$

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

$F = \Sigma(1,3,5,6,7) = A'B'C' + AB + AC + BC$

Σχεδίαση λογικού διαγράμματος του κυκλώματος με πύλες NOT, AND, OR:



# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

**Παράδειγμα 3**: Σύνθεση συνδυαστικού κυκλώματος για τη μετατροπή δυαδικά κωδικοποιημένων δεκαδικών ψηφίων (BCD) σε δυαδικά κωδικοποιημένα ψηφία σύμφωνα με τον κώδικα Gray, με χρήση του ελάχιστου δυνατού πλήθους λογικών πυλών.

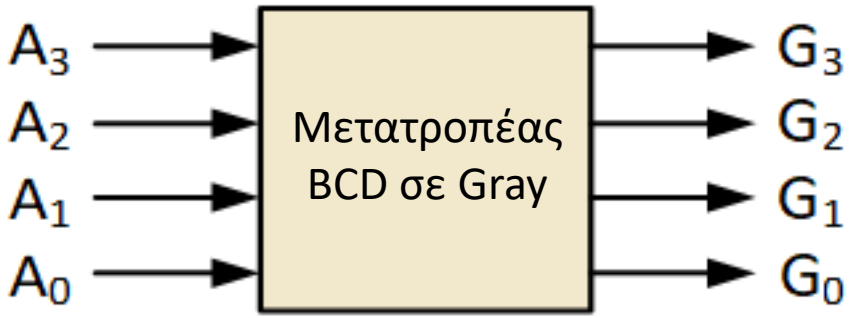
Ο κώδικας BCD κωδικοποιεί με 4 δυαδικά ψηφία καθένα από τα 10 δεκαδικά ψηφία. Συνεπώς, το κύκλωμα διαθέτει 4 εισόδους ($A_3$, $A_2$, $A_1$, $A_0$, όπου $A_3A_2A_1A_0$ είναι το δυαδικά κωδικοποιημένο δεκαδικό ψηφίο).

Ο κώδικας Gray είναι κώδικας παράστασης αριθμών με δυαδικά ψηφία κατά τέτοιο τρόπο ώστε, κατά τη μετάβαση μεταξύ δύο διαδοχικών αριθμών, να αλλάζει μόνο ένα ψηφίο.

Το αντίστοιχο ψηφίο ενός δεκαδικού ψηφίου BCD σε κώδικα Gray συνίσταται από 4 ψηφία, επομένως το κύκλωμα διαθέτει 4 εξόδους ($G_3$, $G_2$, $G_1$, $G_0$, όπου $G_3G_2G_1G_0$ είναι το δεκαδικό ψηφίο κωδικοποιημένο σύμφωνα με τον κώδικα Gray).

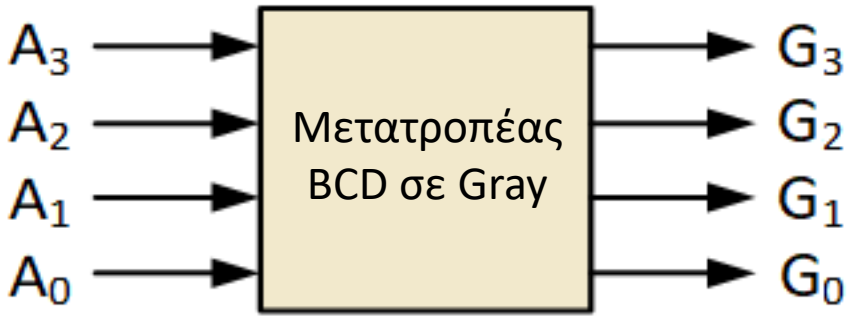
# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:



# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση συνοπτικού διαγράμματος:




Παρατηρούμε ότι οι τελευταίοι 6 συνδυασμοί των εισόδων δεν είναι επιτρεπτοί (αφού δεν αποτελούν δυαδικά κωδικοποιημένα δεκαδικά ψηφία BCD) και αποτελούν αδιάφορες λογικές συνθήκες. Οι ελαχιστόροι που αντιστοιχούν σε αυτούς αποτελούν αδιάφορους όρους και οι 4 συναρτήσεις εξόδου είναι μερικώς καθορισμένες.

Πίνακας αλήθειας:

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $G_3$ | $G_2$ | $G_1$ | $G_0$ |
|---|---|---|---|---|---|---|---|
| 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 | 0 | 1 | 1 | × | × | × | × |
| 1 | 1 | 0 | 0 | × | × | × | × |
| 1 | 1 | 0 | 1 | × | × | × | × |
| 1 | 1 | 1 | 0 | × | × | × | × |
| 1 | 1 | 1 | 1 | × | × | × | × |

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Λογικές συναρτήσεις των 4 εξόδων σε μορφή αθροίσματος ελαχιστόρων:

$$G_3 = \Sigma(8,9) + d(10,11,12,13,14,15)$$

$$G_2 = \Sigma(4,5,6,7,8,9) + d(10,11,12,13,14,15)$$

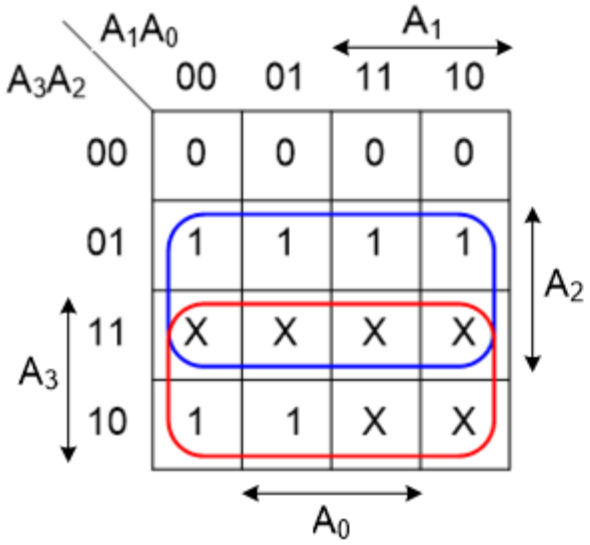
$$G_1 = \Sigma(2,3,4,5) + d(10,11,12,13,14,15)$$

$$G_0 = \Sigma(1,2,5,6,9) + d(10,11,12,13,14,15)$$

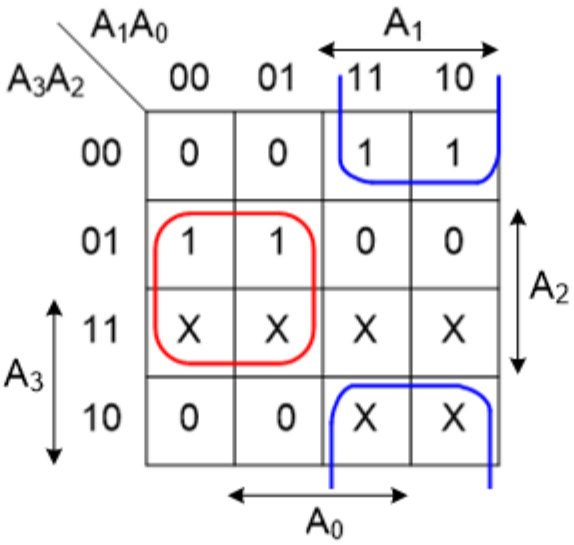
Από τον πίνακα αλήθειας μπορούμε εύκολα να συμπεράνουμε ότι $G_3 = A_3$, επομένως για τη συνάρτηση της εξόδου $G_3$ δεν απαιτείται να διενεργήσουμε ελαχιστοποίηση, ούτε να χρησιμοποιήσουμε λογικές πύλες για την υλοποίησή της.

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

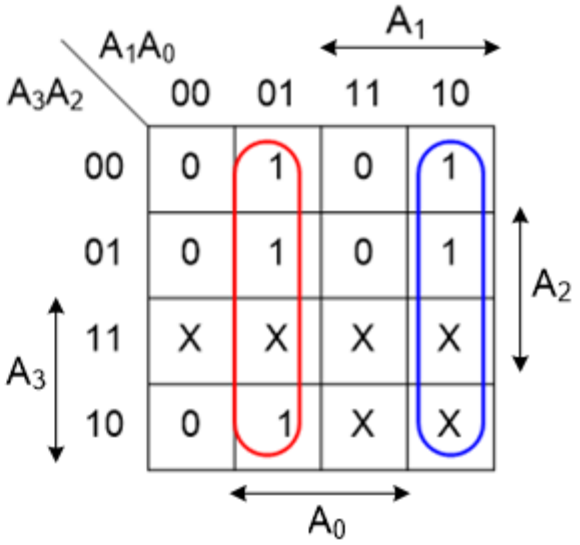
Ελαχιστοποίηση των λογικών συναρτήσεων των εξόδων $G_2$, $G_1$ και $G_0$ με χάρτες Karnaugh:




$G_2 = A_2 + A_3$



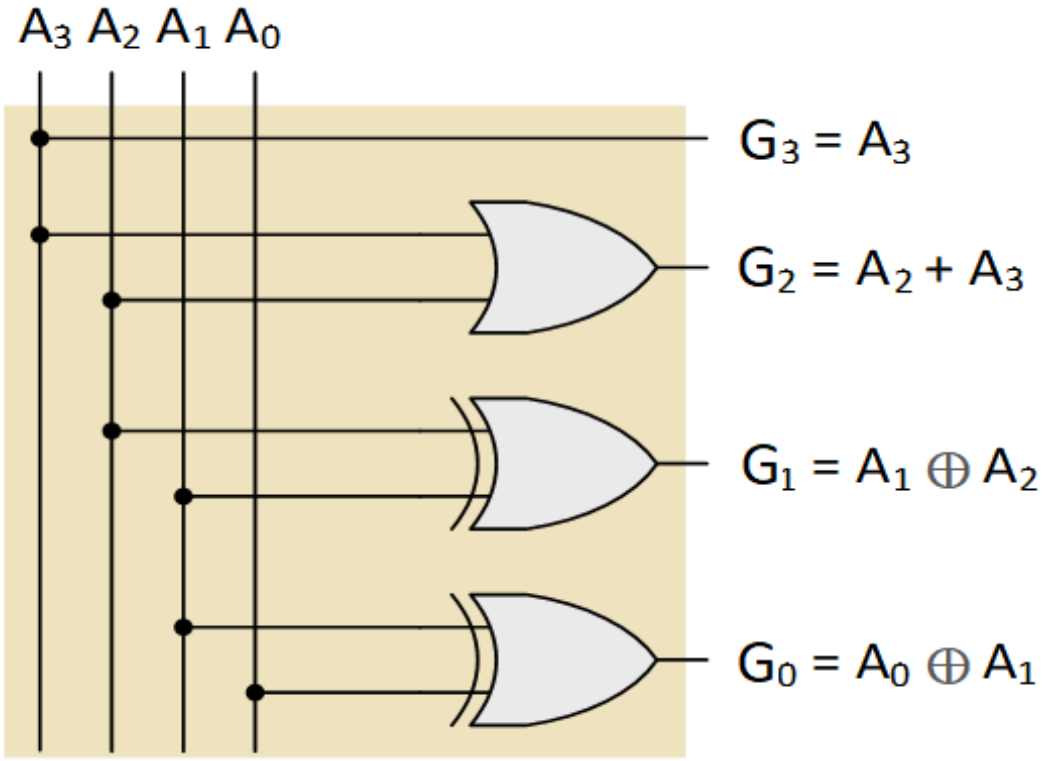

$G_1 = A'_1A_2 + A_1A'_2$
$= A_1 \oplus A_2$




$G_0 = A'_0A_1 + A_0A'_1$
$= A_0 \oplus A_1$

# Σύνθεση (σχεδίαση) συνδυαστικών κυκλωμάτων

Σχεδίαση λογικού διαγράμματος με το ελάχιστο πλήθος λογικών πυλών:



# Ανάλυση συνδυαστικών κυκλωμάτων

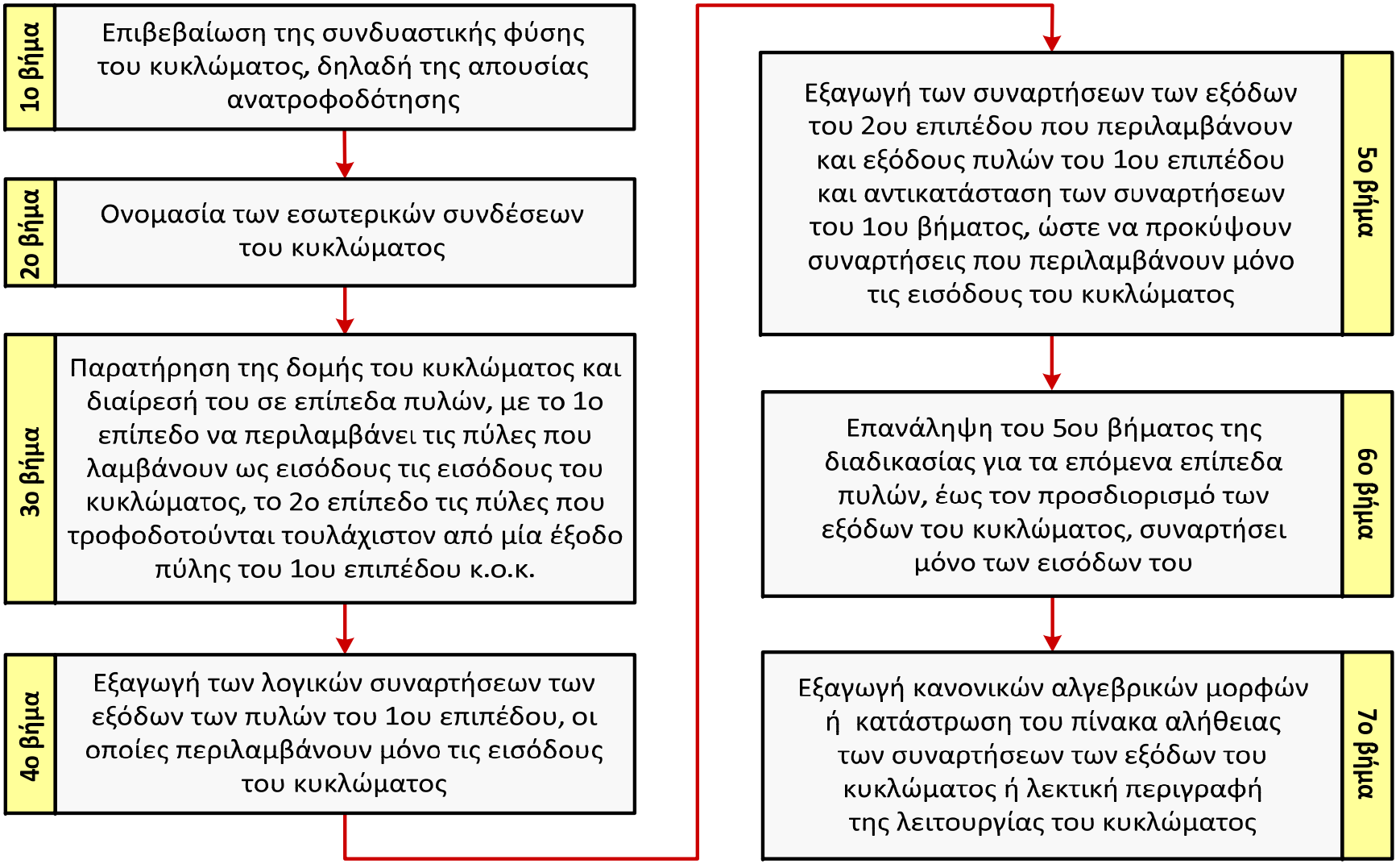
Για να επαληθευτεί η ορθή λειτουργία ενός συνδυαστικού κυκλώματος, δηλαδή για να επιβεβαιωθεί ότι λειτουργεί σύμφωνα με τις προδιαγραφές που σχεδιάστηκε, ακολουθείται μια διαδικασία που αναφέρεται ως ανάλυση συνδυαστικού κυκλώματος.

Επίσης, προκειμένου να διαπιστωθεί η λειτουργική ισοδυναμία δύο ή περισσότερων συνδυαστικών κυκλωμάτων, θα πρέπει αυτά να αναλυθούν.

Λειτουργικά ισοδύναμα είναι δύο συνδυαστικά κυκλώματα που έχουν το ίδιο πλήθος εισόδων και εξόδων και αντιστοιχούν στον ίδιο πίνακα αλήθειας ή στις ίδιες κανονικές αλγεβρικές μορφές.

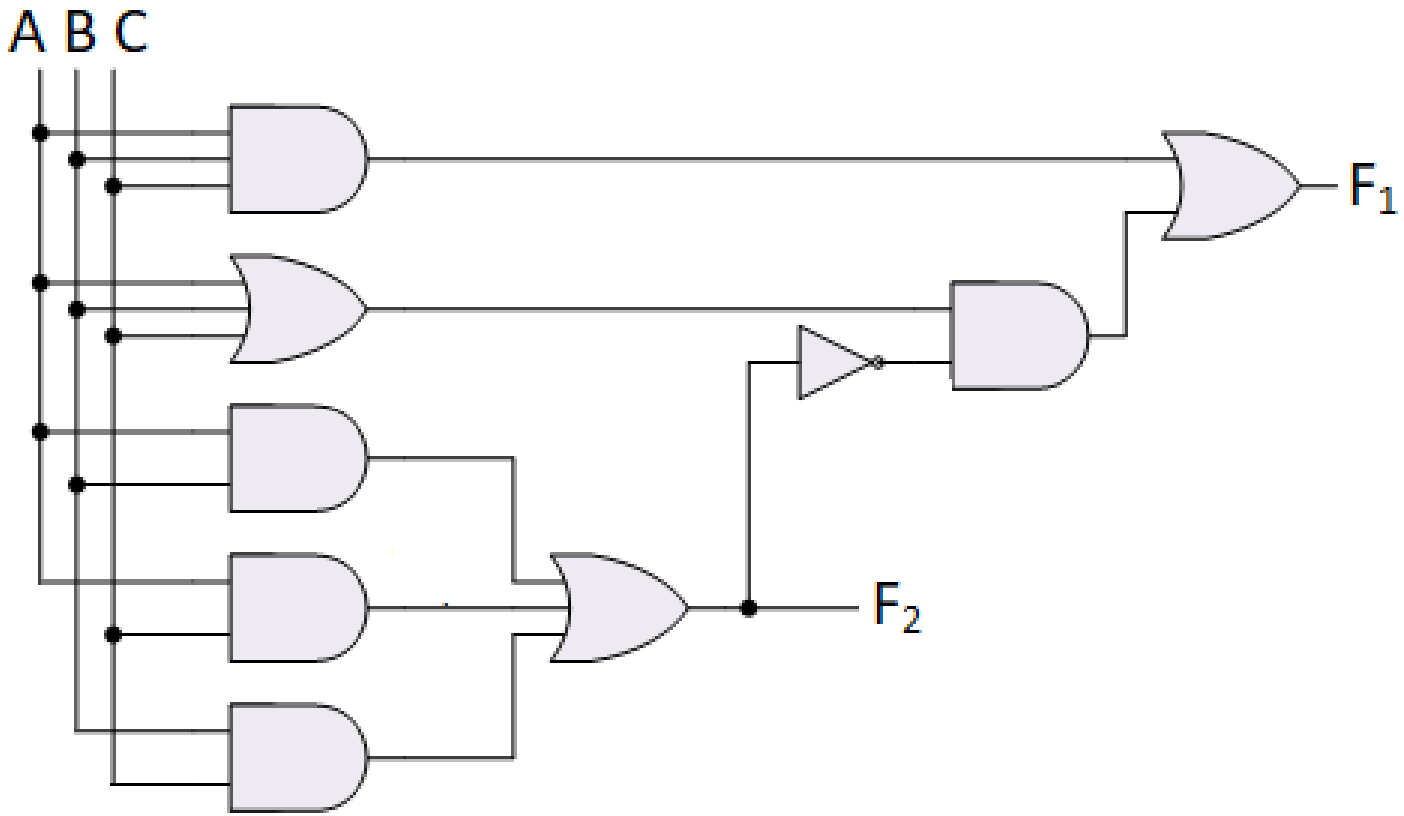
Ανάλυση συνδυαστικού κυκλώματος είναι λοιπόν ο προσδιορισμός της λειτουργίας ή των λειτουργιών που εκτελεί και συνίσταται στην εξαγωγή των λογικών συναρτήσεων ή του πίνακα αλήθειας ή ακόμη και μιας λεκτικής περιγραφής της λειτουργίας ή των λειτουργιών που εκτελούνται.

# Ανάλυση συνδυαστικών κυκλωμάτων

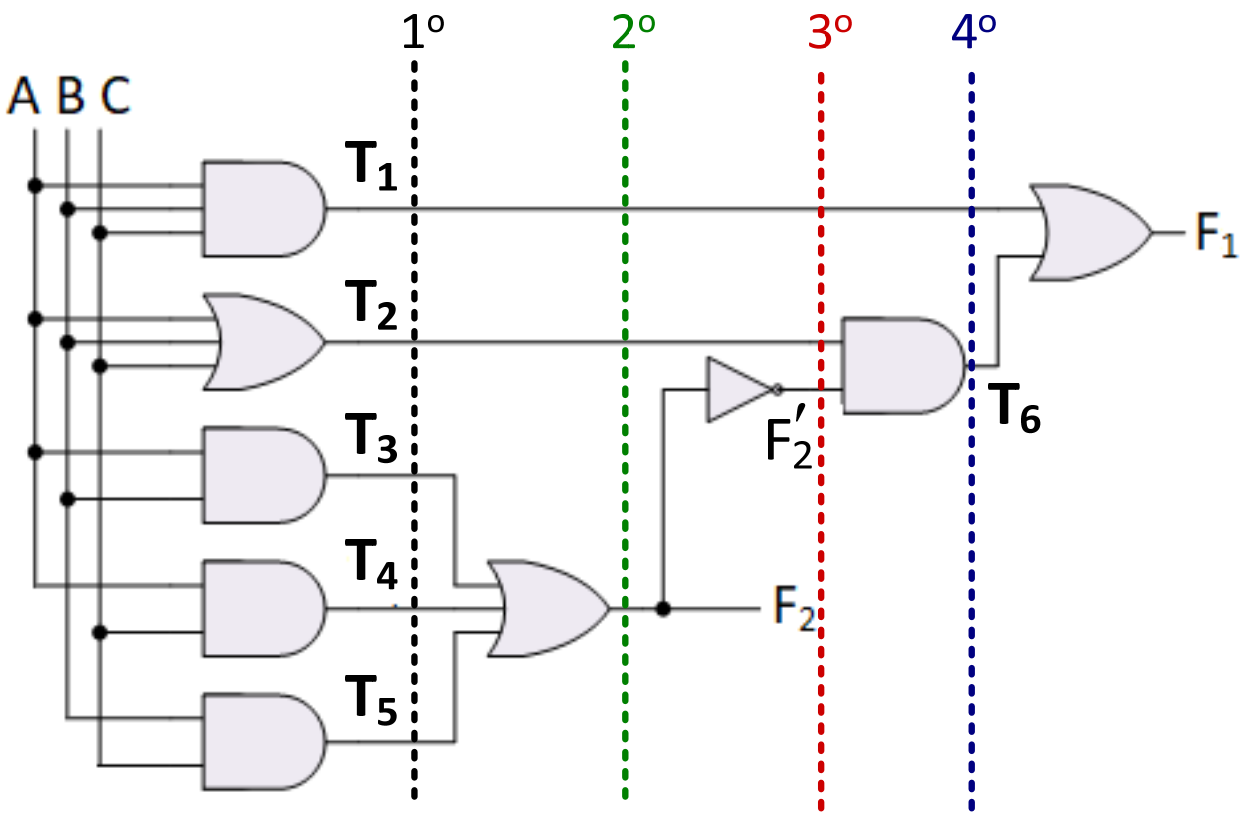


# Ανάλυση συνδυαστικών κυκλωμάτων

**Παράδειγμα:** Ανάλυση συνδυαστικού κυκλώματος ξεκινώντας από το λογικό διάγραμμά του:

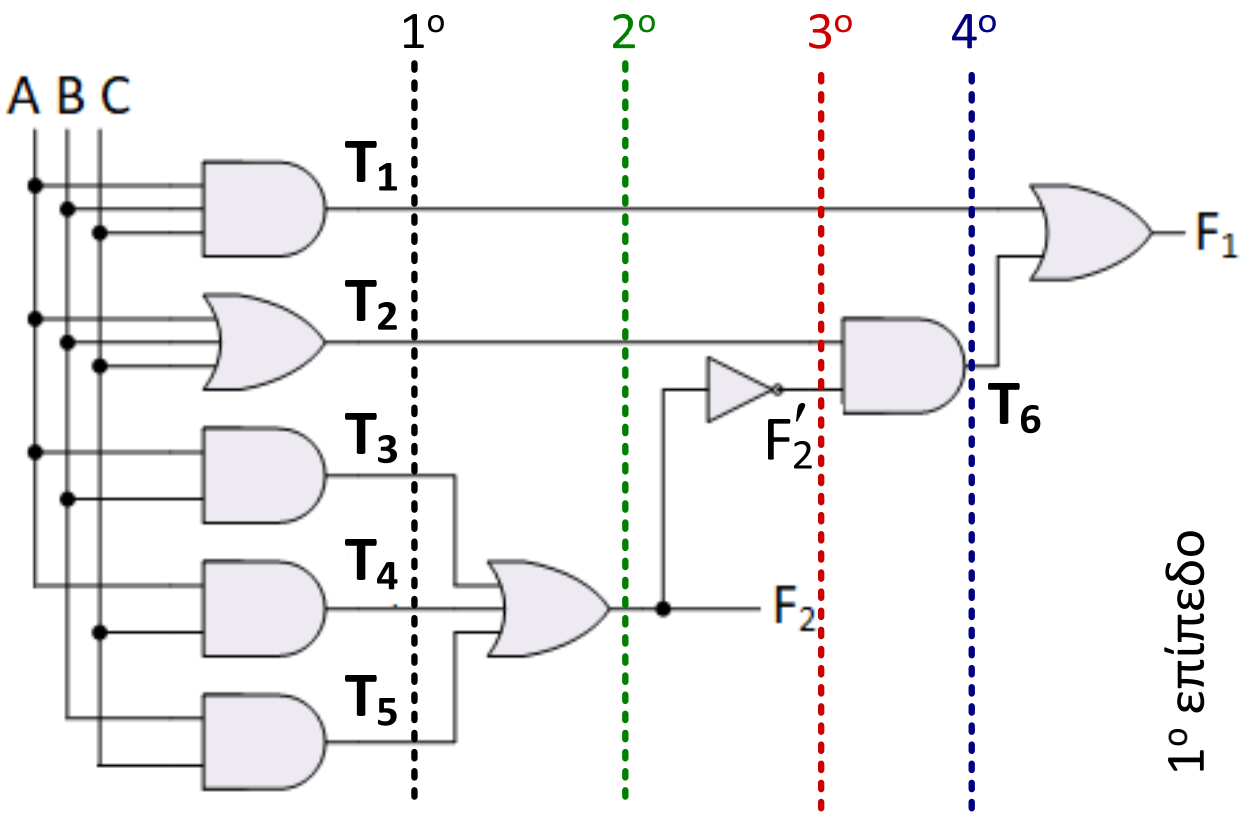


# Ανάλυση συνδυαστικών κυκλωμάτων




Διαίρεση του κυκλώματος σε επίπεδα πυλών

# Ανάλυση συνδυαστικών κυκλωμάτων

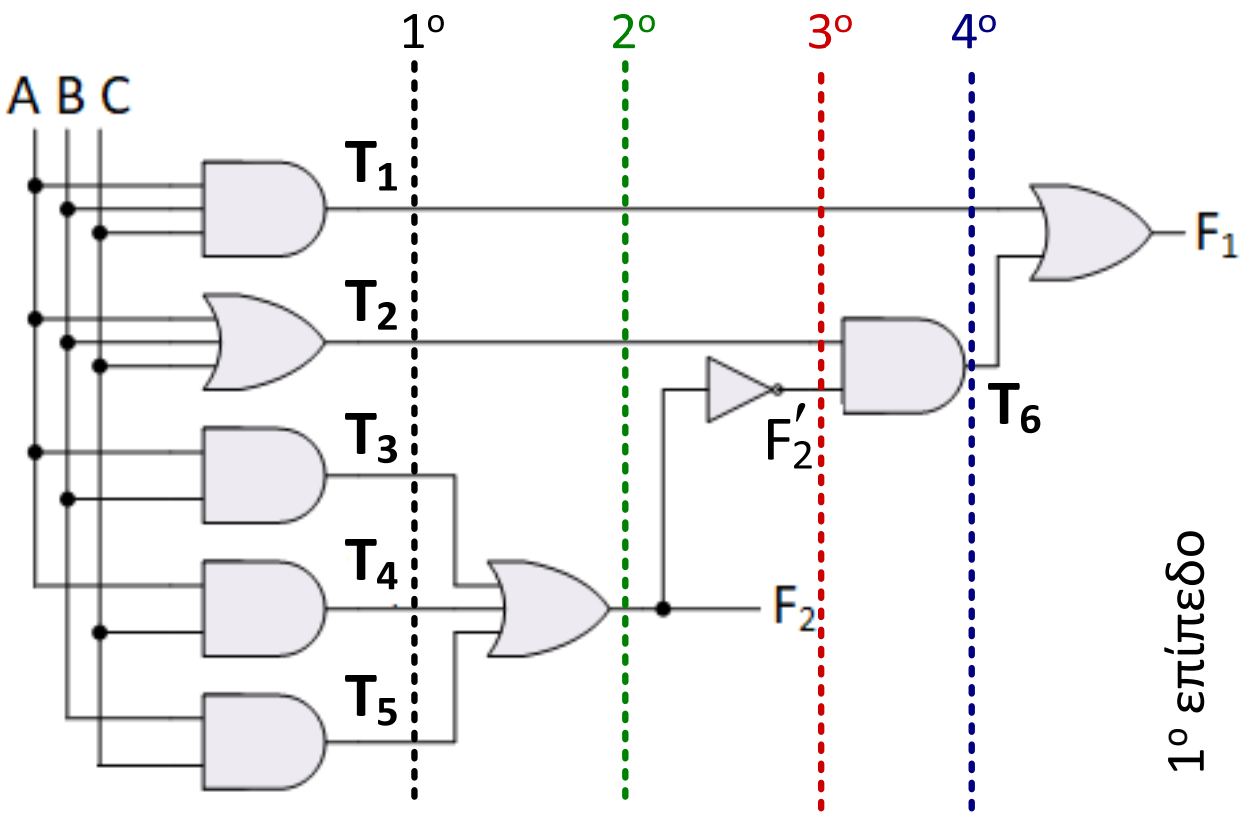



Διαίρεση του κυκλώματος σε επίπεδα πυλών

1ο επίπεδο

$T_1 = ABC$
$T_2 = A + B + C$
$T_3 = AB$
$T_4 = AC$
$T_5 = BC$

# Ανάλυση συνδυαστικών κυκλωμάτων



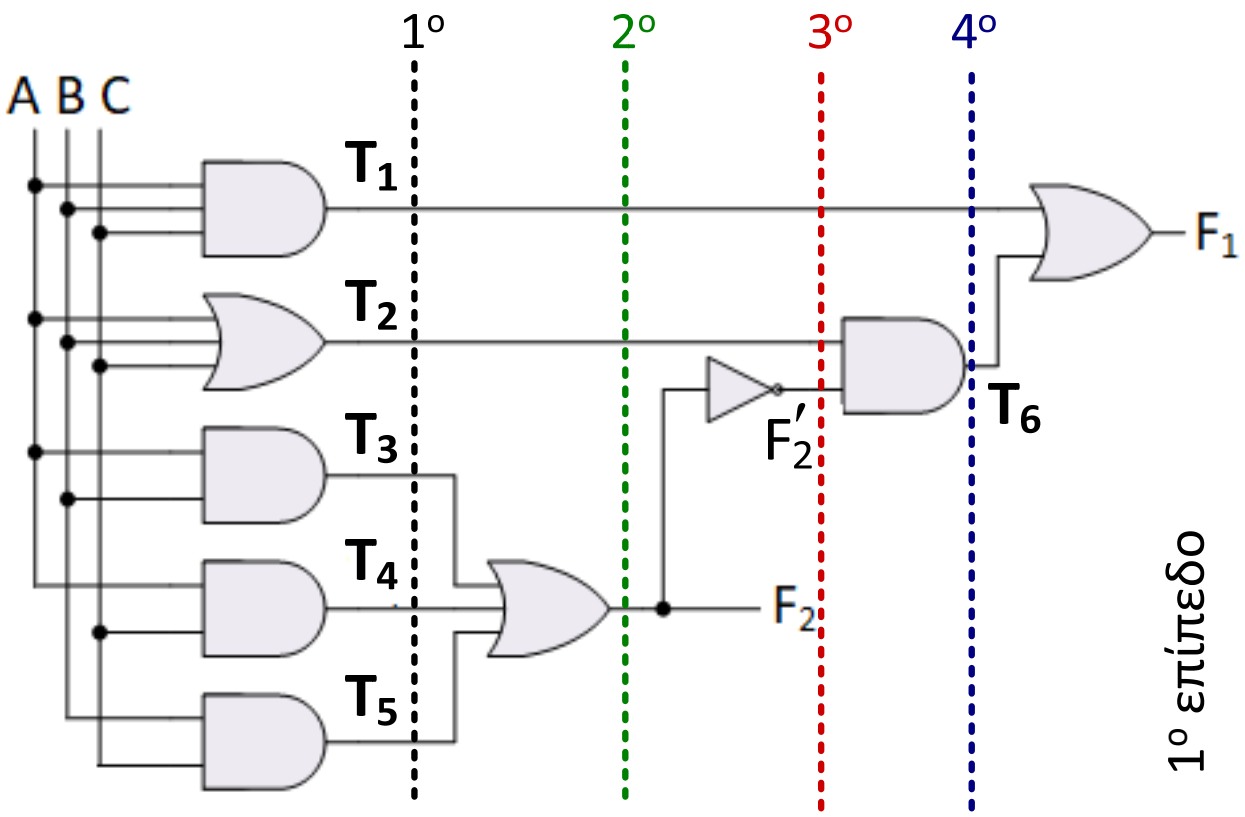

Διαίρεση του κυκλώματος σε επίπεδα πυλών

1ο επίπεδο

$T_1 = ABC$
$T_2 = A + B + C$
$T_3 = AB$
$T_4 = AC$
$T_5 = BC$

2ο επίπεδο: $F_2 = T_3 + T_4 + T_5 = AB + AC + BC$

# Ανάλυση συνδυαστικών κυκλωμάτων




Διαίρεση του κυκλώματος σε επίπεδα πυλών

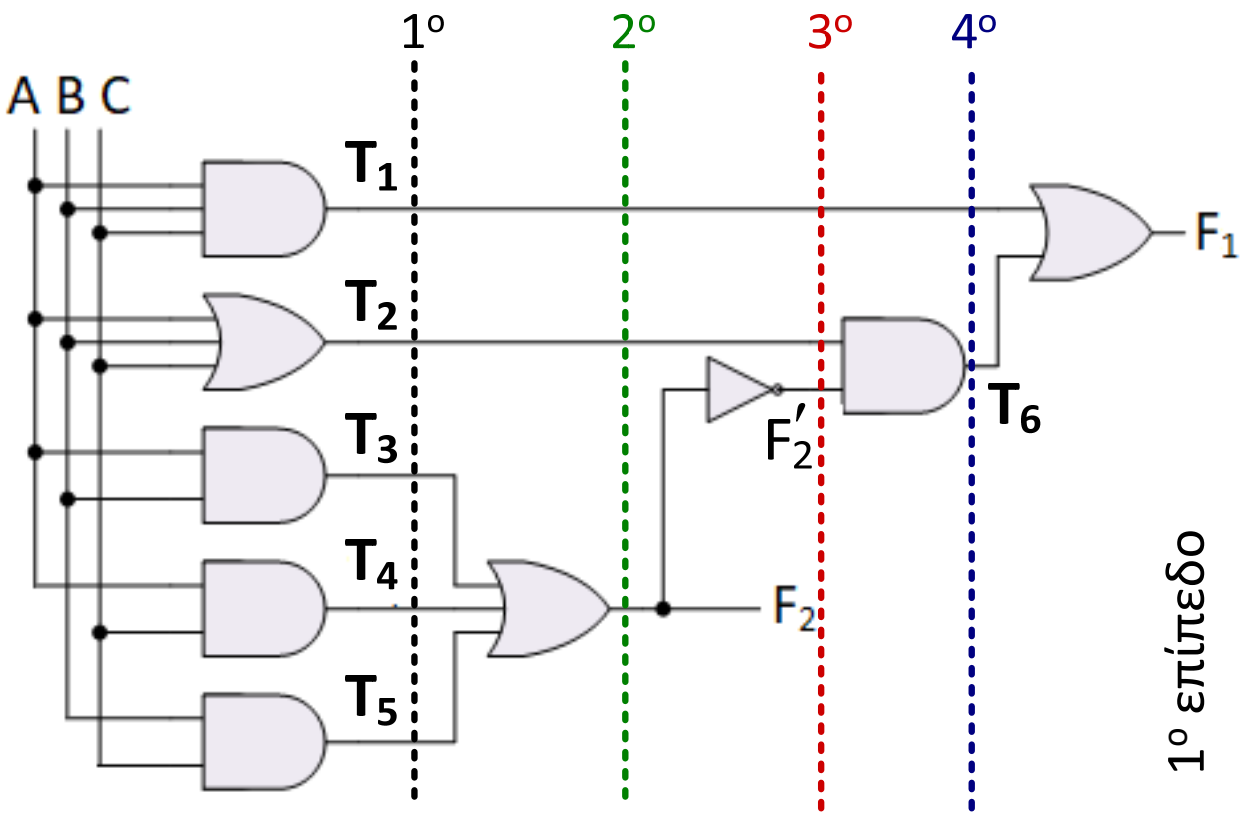
1ο επίπεδο

$T_1 = ABC$
$T_2 = A + B + C$
$T_3 = AB$
$T_4 = AC$
$T_5 = BC$

2ο επίπεδο: $F_2 = T_3 + T_4 + T_5 = AB + AC + BC$

3ο & 4ο επίπεδο: $T_6 = T_2 F'_2 = (A + B + C)(AB + AC + BC)'$

# Ανάλυση συνδυαστικών κυκλωμάτων




Διαίρεση του κυκλώματος σε επίπεδα πυλών

1ο επίπεδο

$T_1 = ABC$
$T_2 = A + B + C$
$T_3 = AB$
$T_4 = AC$
$T_5 = BC$

2ο επίπεδο: $F_2 = T_3 + T_4 + T_5 = AB + AC + BC$

3ο & 4ο επίπεδο: $T_6 = T_2 F'_2 = (A + B + C)(AB + AC + BC)'$

$F_1 = T_1 + T_6 = ABC + (A + B + C)(AB + AC + BC)'$

# Ανάλυση συνδυαστικών κυκλωμάτων

Εξαγωγή κανονικής μορφής (άθροίσμα ελαχιστόρων) των 2 εξόδων:

$F_2 = AB + AC + BC = AB(C + C') + A(B + B')C + (A + A')BC$
$= ABC + ABC' + ABC + AB'C + ABC + A'BC$
$= ABC + ABC' + AB'C + A'BC = \Sigma(7,6,5,3) = \mathbf{\Sigma(3,5,6,7)}$

$F_1 = ABC + (A + B + C)(AB + AC + BC)'$
$= ABC + (A + B + C)(AB)'(AC)'(BC)'$
$= ABC + (A + B + C )(A' + B')(A' + C')(B' + C')$
$= ABC + (AA' + AB' +A'B + BB' + A'C + B'C)(A'B' +A'C' + B'C' + C'C')$
$= ABC + (AB' +A'B + A'C + B'C)(A'B' + A'C' + B'C' + C')$
$= ABC + (AB' +A'B + A'C + B'C)[A'B' + C'(A' + B' + 1)]$
$= ABC + (AB' +A'B + A'C + B'C)(A'B' + C')$
$= ABC + (AA'B'B' + AB'C' + A'A'BB' + A'BC' + A'A'B'C + A'CC' + A'B'B'C + B'CC')$
$= ABC + AB'C' + A'BC' + A'B'C = \Sigma(7,4,2,1) = \mathbf{\Sigma(1,2,4,7)}$

# Ανάλυση συνδυαστικών κυκλωμάτων

Κατάστρωση πινάκων αλήθειας των 2 εξόδων:

$F_2 = \Sigma(3,5,6,7)$

$F_1 = \Sigma(1,2,4,7)$

| A | B | C | $F_2$ | $F_1$ |
|---|---|---|---|---|
| 0 | 0 | 0 | **0** | **0** |
| 0 | 0 | 1 | **0** | **1** |
| 0 | 1 | 0 | **0** | **1** |
| 0 | 1 | 1 | **1** | **0** |
| 1 | 0 | 0 | **0** | **1** |
| 1 | 0 | 1 | **1** | **0** |
| 1 | 1 | 0 | **1** | **0** |
| 1 | 1 | 1 | **1** | **1** |

Λεκτική περιγραφή λειτουργίας:

Αν και από τις κανονικές αλγεβρικές μορφές δεν μπορούμε να εξάγουμε μια λεκτική περιγραφή της λειτουργίας του κυκλώματος, από τον πίνακα αλήθειας προκύπτει ότι οι 2 έξοδοι του κυκλώματος σχηματίζουν διψήφιο δυαδικό αριθμό που ισούται με το πλήθος των μονάδων που περιλαμβάνονται στις 3 εισόδους.

# Ανάλυση συνδυαστικών κυκλωμάτων

Εναλλακτικά, μπορούμε να καταστρώσουμε τον πίνακα αλήθειας των εξόδων, χωρίς προηγουμένως να εξάγουμε τις αλγεβρικές τους εκφράσεις.

Η διαδικασία συνίσταται στη σταδιακή εξαγωγή των στηλών του πίνακα αλήθειας που αντιστοιχούν στις εσωτερικές συνδέσεις του κυκλώματος, ώστε από αυτές να προκύψουν οι στήλες του πίνακα που αντιστοιχούν στις εξόδους του κυκλώματος, με χρήση μόνο των ορισμών των βασικών λογικών πράξεων.

**ABC|A+B+C|AB|AC|BC**

| A | B | C | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $F_2=T_3+T_4+T_5$ | $F'_2$ | $T_6=T_2F'_2$ | $F_1=T_1+T_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | **1** | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | **1** | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | **1** | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **1** | 0 | 0 | **1** |

# Τυποποιημένα συνδυαστικά κυκλώματα

Τα ψηφιακά συστήματα σχεδιάζονται και υλοποιούνται κυρίως για να μετασχηματίζουν δεδομένα.

Κάθε μετασχηματισμός μπορεί να παρασταθεί από μία ή περισσότερες λογικές συναρτήσεις.

Ορισμένες κατηγορίες μετασχηματισμών εμφανίζονται αρκετά συχνά σε πρακτικά προβλήματα, όπως για παράδειγμα οι αριθμητικές πράξεις, η κωδικοποίηση και η αποκωδικοποίηση δεδομένων, η επιλογή και μεταφορά δεδομένων.

Είναι λοιπόν χρήσιμο να υπάρχουν τα κυκλώματα αυτά προσχεδιασμένα και τυποποιημένα, έτσι ώστε να είναι εύκολη η χρησιμοποίησή τους στα ψηφιακά συστήματα.

# Αριθμητικά συνδυαστικά κυκλώματα

Στα ψηφιακά συστήματα χρησιμοποιούνται ευρέως συνδυαστικά κυκλώματα που επιτελούν βασικές αριθμητικές λειτουργίες, όπως πρόσθεση, αφαίρεση, πολλαπλασιασμό, σύγκριση δυαδικών αριθμών κ.ά.

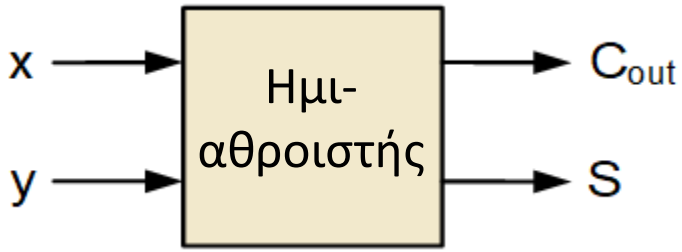
Η πρόσθεση δύο δυαδικών ψηφίων υλοποιείται από ένα συνδυαστικό κύκλωμα δύο εισόδων, δηλαδή των ψηφίων που προστίθενται, και δύο εξόδων, δηλαδή του αθροίσματος και του κρατουμένου.

Ωστόσο, κατά την πρόσθεση δύο αριθμών με περισσότερα ψηφία, για την υλοποίηση της πρόσθεσης των ψηφίων μιας θέσης απαιτείται συνδυαστικό κύκλωμα με τρεις εισόδους, έτσι ώστε να συμμετέχει στην πράξη και το κρατούμενο της πρόσθεσης των ψηφίων της προηγούμενης θέσης.

Το πρώτο κύκλωμα αναφέρεται ως ημιαθροιστής (half adder, HA), ενώ το δεύτερο ως πλήρης αθροιστής (full adder, FA).

# Ημιαθροιστής (half adder)

Ημιαθροιστής είναι το συνδυαστικό κύκλωμα 2 εισόδων (x, y), δηλαδή των ψηφίων που προστίθενται, και 2 εξόδων, δηλαδή του αθροίσματος (S) και του κρατουμένου ($C_{out}$).




| x | y | $C_{out}$ | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | **1** |
| 1 | 0 | 0 | **1** |
| 1 | 1 | **1** | 0 |

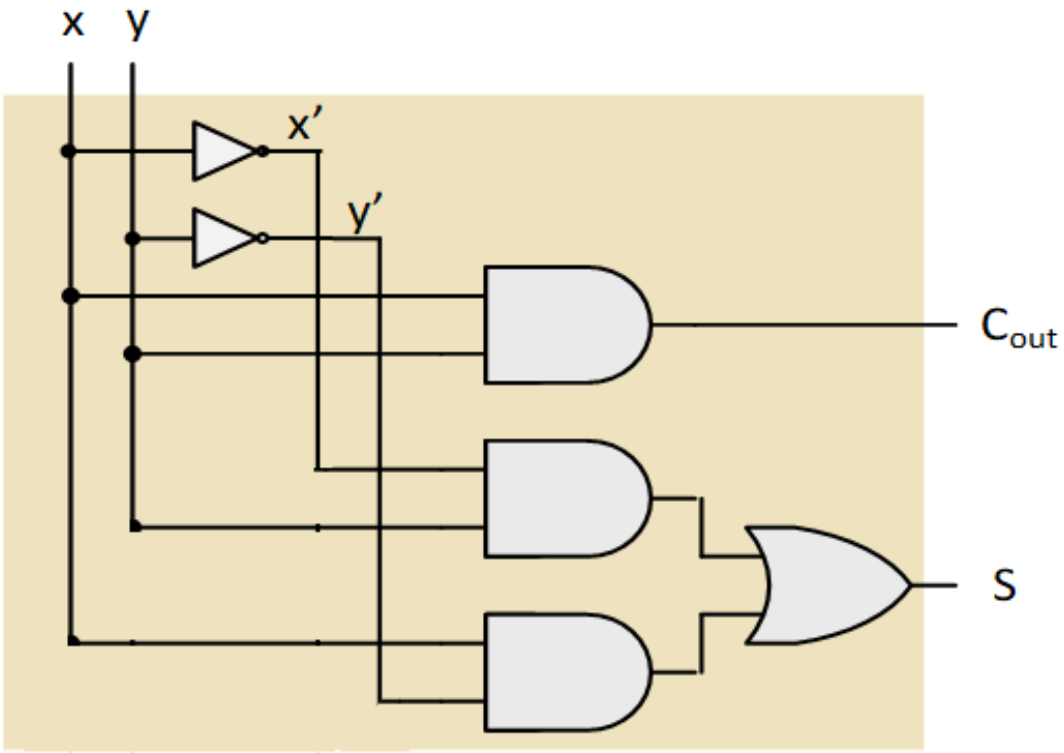
$$C_{out} = xy$$

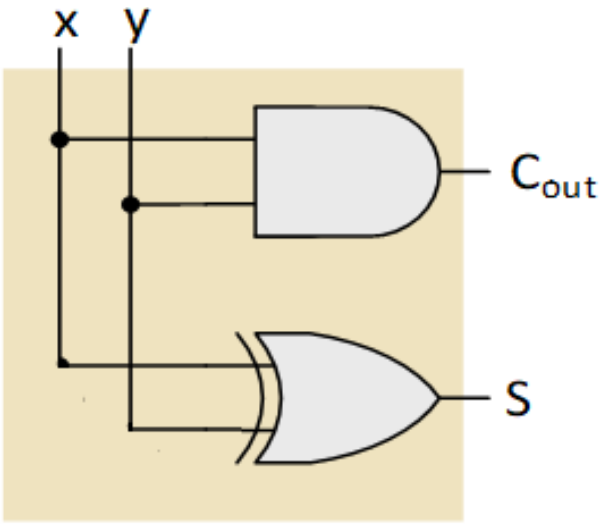
$$S = x'y + xy' = x \oplus y$$

# Ημιαθροιστής (half adder)

$C_{out} = xy$

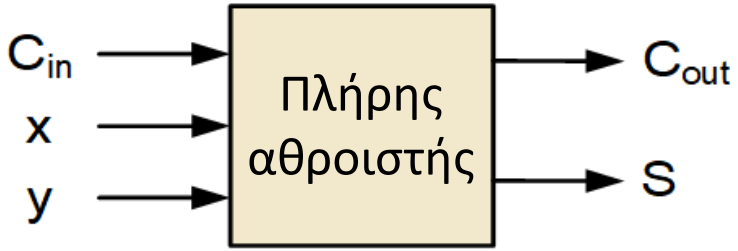
$S = x'y + xy' = x \oplus y$



# Πλήρης αθροιστής (full adder)

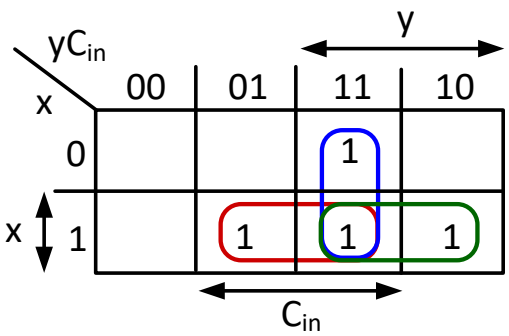
Πλήρης αθροιστής είναι το συνδυαστικό κύκλωμα 3 εισόδων που προσθέτει 2 δυαδικά ψηφία (x, y) και το τυχόν κρατούμενο ($C_{in}$) που έχει προκύψει από πρόσθεση σε προηγούμενη θέση και δίνει ως αποτέλεσμα το άθροισμα (S) και το κρατούμενο ($C_{out}$) της πρόσθεσης.




| x | y | $C_{in}$ | S | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Πλήρης αθροιστής (full adder)

Πλήρης αθροιστής είναι το συνδυαστικό κύκλωμα 3 εισόδων που προσθέτει 2 δυαδικά ψηφία (x, y) και το τυχόν κρατούμενο ($C_{in}$) που έχει προκύψει από πρόσθεση σε προηγούμενη θέση και δίνει ως αποτέλεσμα το άθροισμα (S) και το κρατούμενο ($C_{out}$) της πρόσθεσης.

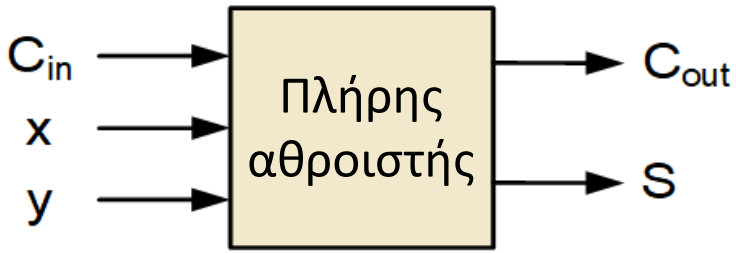



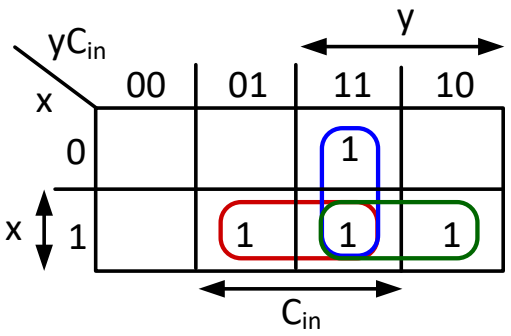

| x | y | $C_{in}$ | S | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

$$C_{out} = x'yC_{in} + xy'C_{in} + xyC'_{in} + xyC_{in}$$
$$= \mathbf{xy + yC_{in} + xC_{in}}$$

# Πλήρης αθροιστής (full adder)

Πλήρης αθροιστής είναι το συνδυαστικό κύκλωμα 3 εισόδων που προσθέτει 2 δυαδικά ψηφία (x, y) και το τυχόν κρατούμενο ($C_{in}$) που έχει προκύψει από πρόσθεση σε προηγούμενη θέση και δίνει ως αποτέλεσμα το άθροισμα (S) και το κρατούμενο ($C_{out}$) της πρόσθεσης.

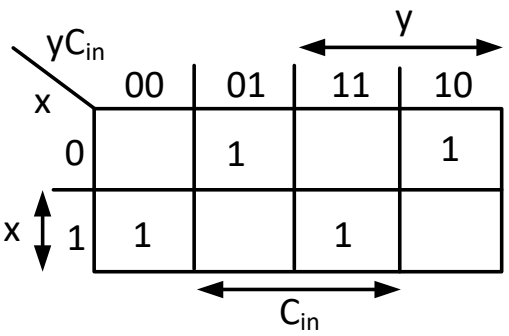



| x | y | $C_{in}$ | S | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

$C_{out} = x'yC_{in} + xy'C_{in} + xyC'_{in} + xyC_{in}$

$= \mathbf{xy + yC_{in} + xC_{in}}$



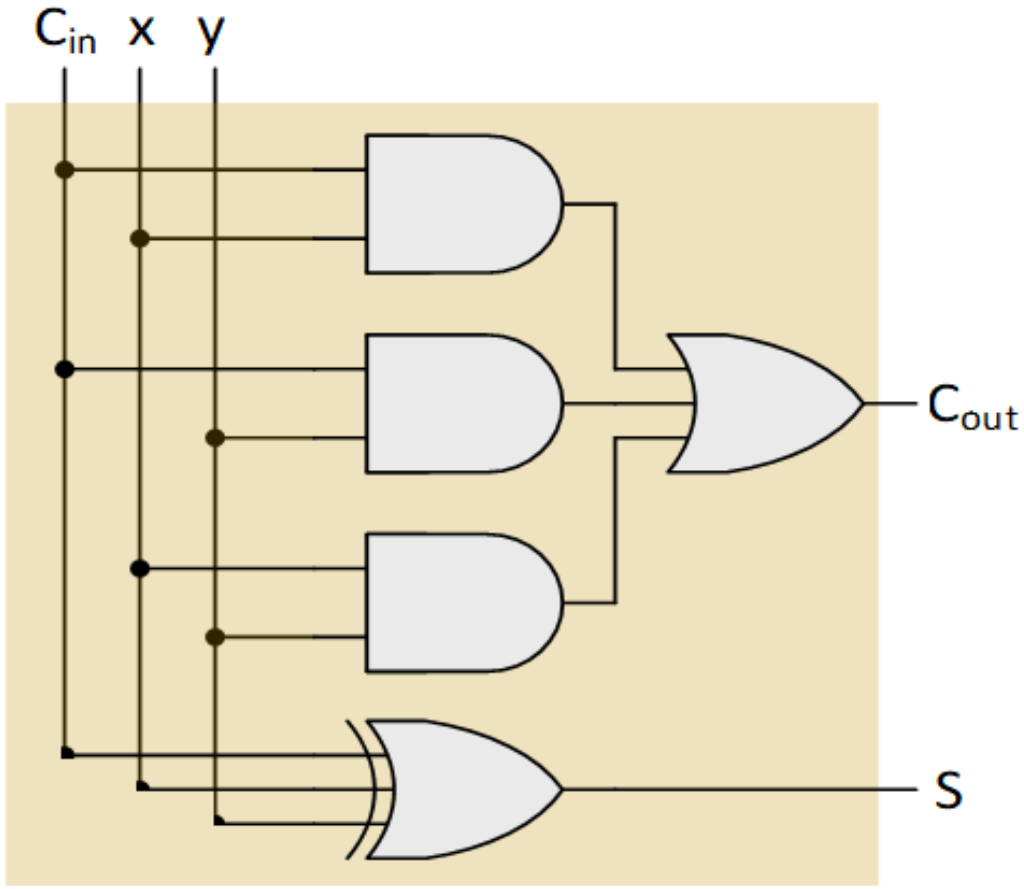

**S** $= x'y'C_{in} + x'yC'_{in} + xy'C'_{in} + xyC_{in}$

$= (xy' + x'y)C'_{in} + (xy + x'y')C_{in}$

$= (x \oplus y)C'_{in} + (x \oplus y)'C_{in} = \mathbf{x \oplus y \oplus C_{in}}$ (περιττή συνάρτηση)

Ο χάρτης Karnaugh του S δεν οδηγεί σε απλοποίηση.

# Πλήρης αθροιστής (full adder)




$$C_{out} = xy + yC_{in} + xC_{in}$$

$$S = x \oplus y \oplus C_{in}$$

# Πλήρης αθροιστής (full adder)

**Παράδειγμα:** Υλοποίηση ενός πλήρους αθροιστή με 2 ημιαθροιστές και μία πύλη OR 2 εισόδων.

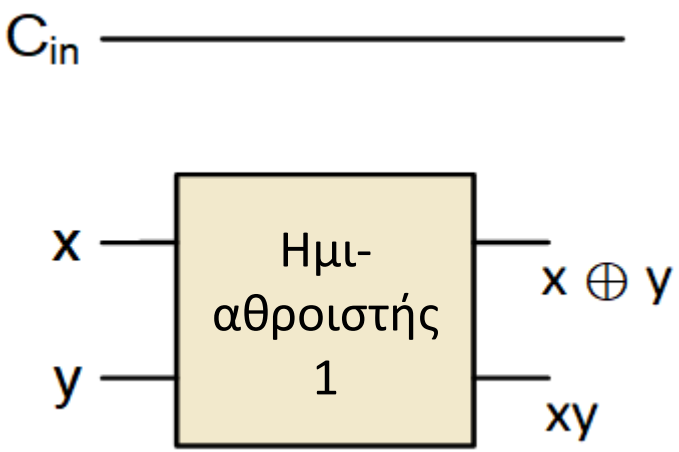
Η λογική συνάρτηση του αθροίσματος του πλήρους αθροιστή μπορεί να εκφραστεί ως εξής:

$$S = (x \oplus y) \oplus C_{in}.$$

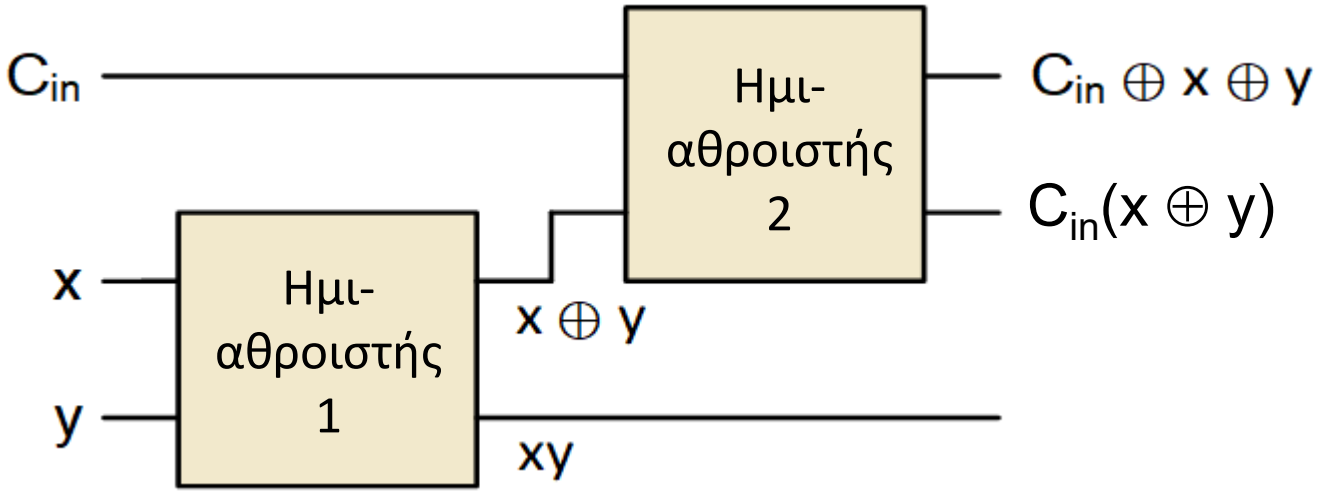
Η λογική συνάρτηση του κρατούμενου εξόδου του πλήρους αθροιστή μπορεί να εκφραστεί ως εξής:

$$\begin{aligned} C_{out} &= x'yC_{in} + xy'C_{in} + xyC'_{in} + xyC_{in} \\ &= C'_{in}xy + C_{in}x'y + C_{in}xy' + C_{in}xy \\ &= (C'_{in} + C_{in})xy + C_{in}(x'y + xy') \\ &= xy + (x \oplus y)C_{in} \end{aligned}$$

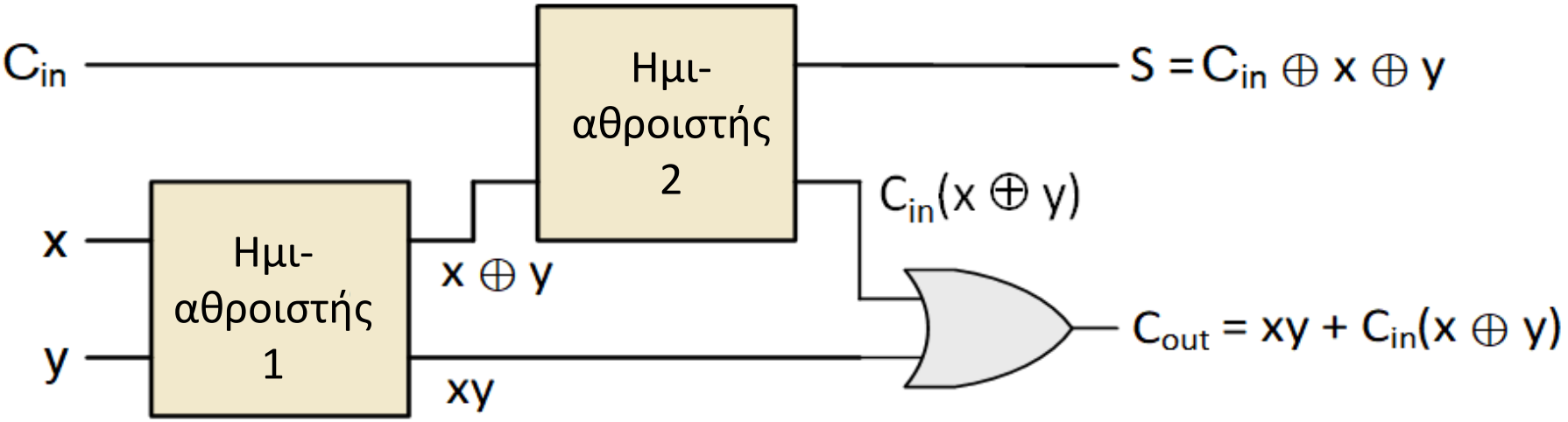
# Πλήρης αθροιστής (full adder)



# Πλήρης αθροιστής (full adder)



# Πλήρης αθροιστής (full adder)



# Πλήρης αφαιρέτης (full subtractor)

Ο πλήρης αφαιρέτης διαθέτει 3 εισόδους, x (μειωτέος), y (αφαιρετέος) και $B_{in}$ (δανειζόμενο ψηφίο από προηγούμενη αφαίρεση) και 2 εξόδους D (διαφορά) και $B_{out}$ (δανειζόμενο ψηφίο που προκύπτει από την αφαίρεση).

Κατά την αφαίρεση 2 δυαδικών ψηφίων προσθέτουμε το δανειζόμενο ψηφίο από την προηγούμενη αφαίρεση στον αφαιρετέο και το άθροισμά τους αφαιρείται από τον μειωτέο: $D = x - (y + B_{in})$.

Πίνακας αλήθειας πλήρους αφαιρέτη

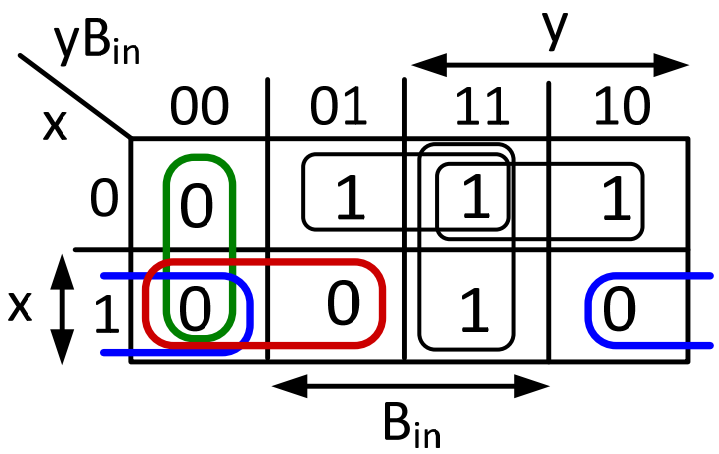
| x | y | $B_{in}$ | $B_{out}$ | D |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | **1** | **1** |
| 0 | 1 | 0 | **1** | **1** |
| 0 | 1 | 1 | **1** | 0 |
| 1 | 0 | 0 | 0 | **1** |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | **1** | **1** |

Παρατηρούμε ότι η διαφορά D είναι περιττή συνάρτηση των 3 μεταβλητών x, y, $B_{in}$, συνεπώς ισχύει ότι και για το άθροισμα S στον πλήρη αθροιστή:

$$D = x \oplus y \oplus B_{in}$$

Επίσης, προκύπτει ότι: $B_{out} = \Sigma(1,2,3,7)$

# Πλήρης αφαιρέτης (full subtractor)




$$B_{out} = x'y + yB_{in} + x'B_{in}$$

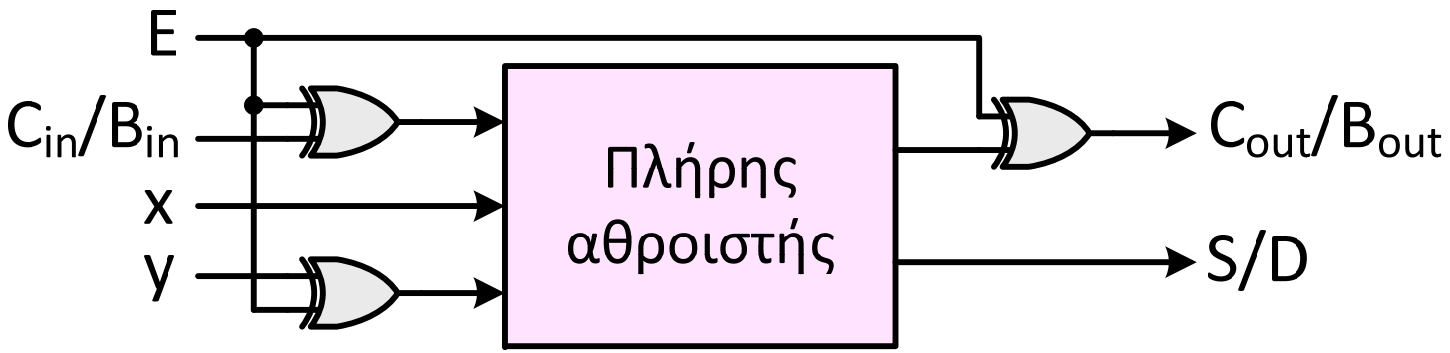
$$B'_{out} = xy' + y'B'_{in} + xB'_{in}$$

$$C_{out} = xy + yC_{in} + xC_{in}$$

Συγκρίνοντας τη συμπληρωματική συνάρτηση του $B_{out}$ του πλήρους αφαιρέτη με τη συνάρτηση του $C_{out}$ του πλήρους αθροιστή και λαμβάνοντας υπόψη ότι $S = D = x \oplus y \oplus B_{in} = x \oplus y' \oplus B'_{in}$, προκύπτει ότι ο πλήρης αφαιρέτης υλοποιείται με έναν πλήρη αθροιστή και 3 αντιστροφείς:

# Πλήρης αθροιστής / αφαιρέτης

Με βάση τα προαναφερθέντα, και αφού ισχύει ότι $A \oplus 1 = A'$ και $A \oplus 0 = A$, μπορούμε να υλοποιήσουμε ένα κύκλωμα πλήρους αθροιστή / αφαιρέτη με έναν πλήρη αθροιστή και 3 πύλες XOR 2 εισόδων.




Το παραπάνω κύκλωμα λειτουργεί ως πλήρης αθροιστής όταν E = 0 και ως πλήρης αφαιρέτης όταν E = 1.

# Παράλληλος αθροιστής

Χρησιμοποιώντας ως δομικό στοιχείο τον πλήρη αθροιστή, μπορούμε να συνθέσουμε συνδυαστικά κυκλώματα παράλληλων αθροιστών για αριθμούς με περισσότερα ψηφία.

Το αποτέλεσμα της πρόσθεσης δύο μη προσημασμένων δυαδικών αριθμών Ν ψηφίων αποτελείται από Ν + 1 ψηφία, συμπεριλαμβανομένου του τελικού κρατουμένου, συνεπώς το συνδυαστικό κύκλωμα του παράλληλου αθροιστή διαθέτει Ν + 1 εξόδους.

Η πρόσθεση 2 δυαδικών αριθμών εκτελείται ανά ζεύγη ψηφίων της ίδιας θέσης, ξεκινώντας από το ζεύγος των λιγότερο σημαντικών ψηφίων των αριθμών και αν προκύπτει κρατούμενο ψηφίο μετά την πρόσθεση σε κάποια θέση, τότε αυτό προστίθεται στα ψηφία τής αμέσως πιο σημαντικής θέσης.

Έτσι, ο παράλληλος αθροιστής δύο δυαδικών αριθμών Ν ψηφίων προκύπτει εύκολα, εάν συνδέσουμε σειριακά Ν πλήρεις αθροιστές.

# Παράλληλος αθροιστής

Κάθε πλήρης αθροιστής δέχεται ένα ζεύγος ψηφίων των αριθμών εισόδου και παράγει ένα ψηφίο αθροίσματος και ένα ψηφίο κρατουμένου, το οποίο διαδίδεται στον πλήρη αθροιστή της επόμενης (πιο σημαντικής) θέσης.

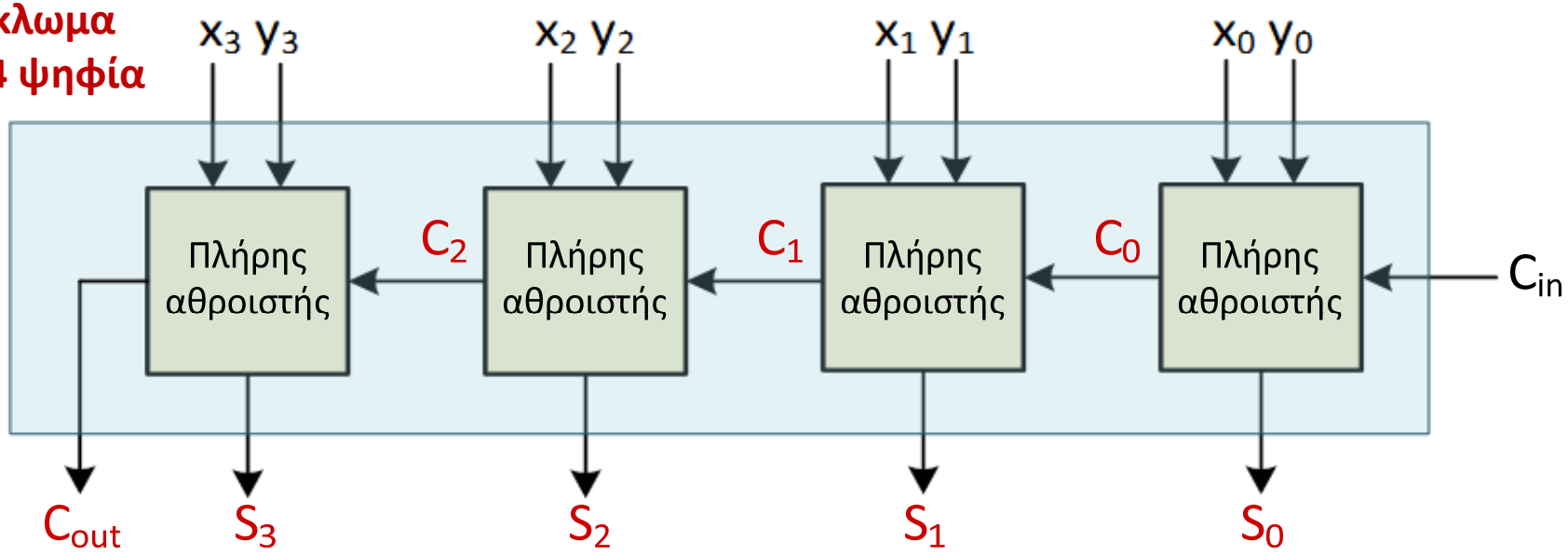
Το τελικό κρατούμενο της πρόσθεσης των δύο αριθμών λαμβάνεται στην έξοδο κρατουμένου του πλήρους αθροιστή της πιο σημαντικής θέσης.

Στην λιγότερο σημαντική θέση δεν υφίσταται κρατούμενο από πρόσθεση σε προηγούμενη θέση ($C_{in} = 0$).

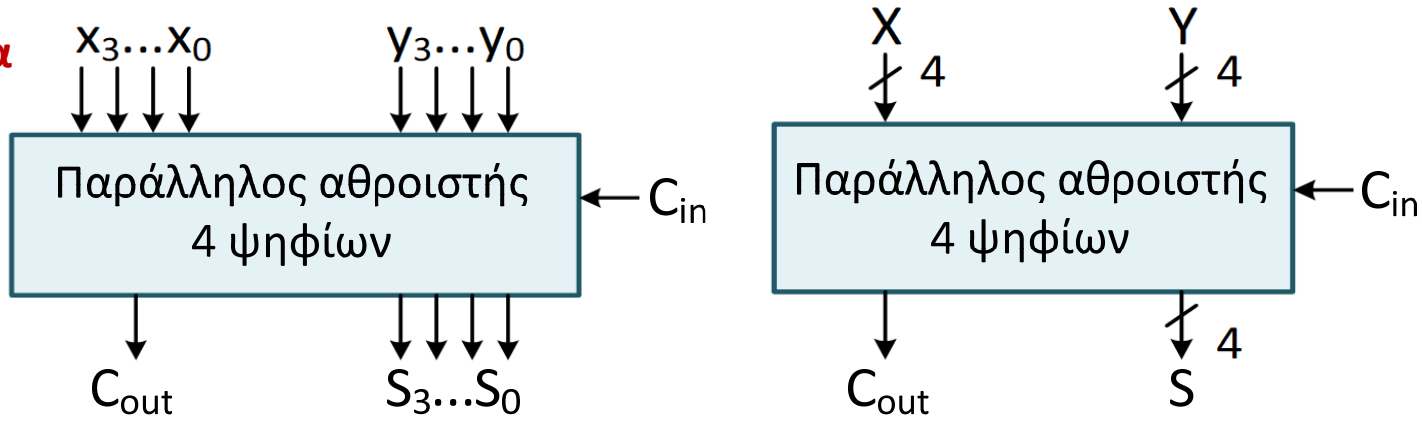
Οι παράλληλοι αθροιστές αυτής της δομής αναφέρονται ως αθροιστές διάδοσης κρατουμένου ή κυματικοί αθροιστές (ripple carry adders).

# Παράλληλος αθροιστής

**Λογικό κύκλωμα για 4 ψηφία**



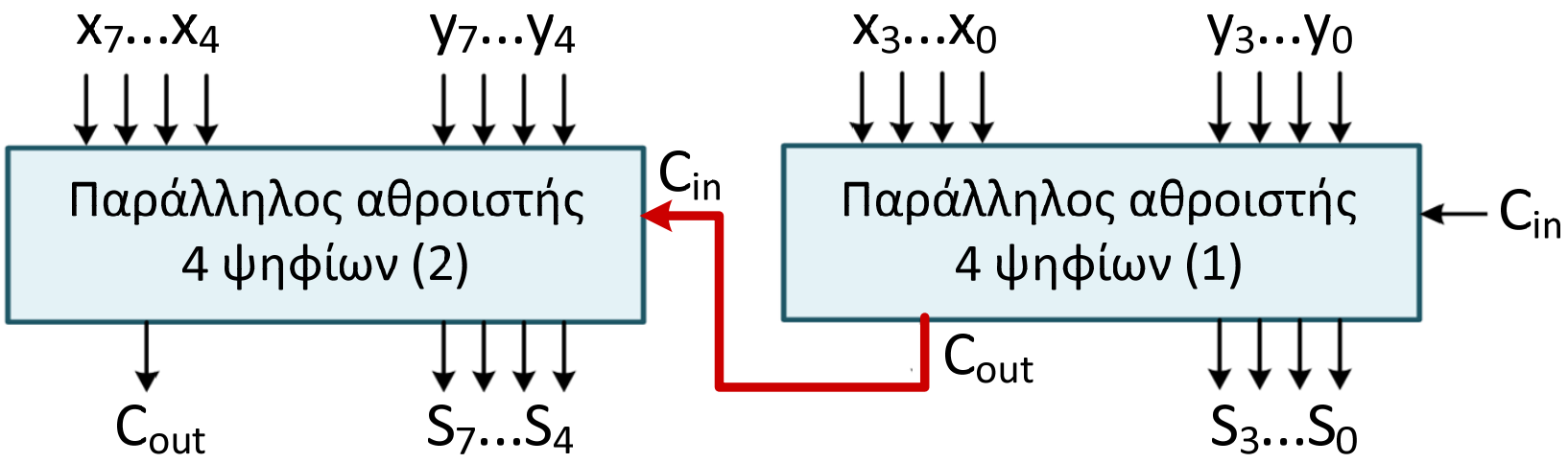

**Συνοπτικά διαγράμματα για 4 ψηφία**



# Παράλληλος αθροιστής

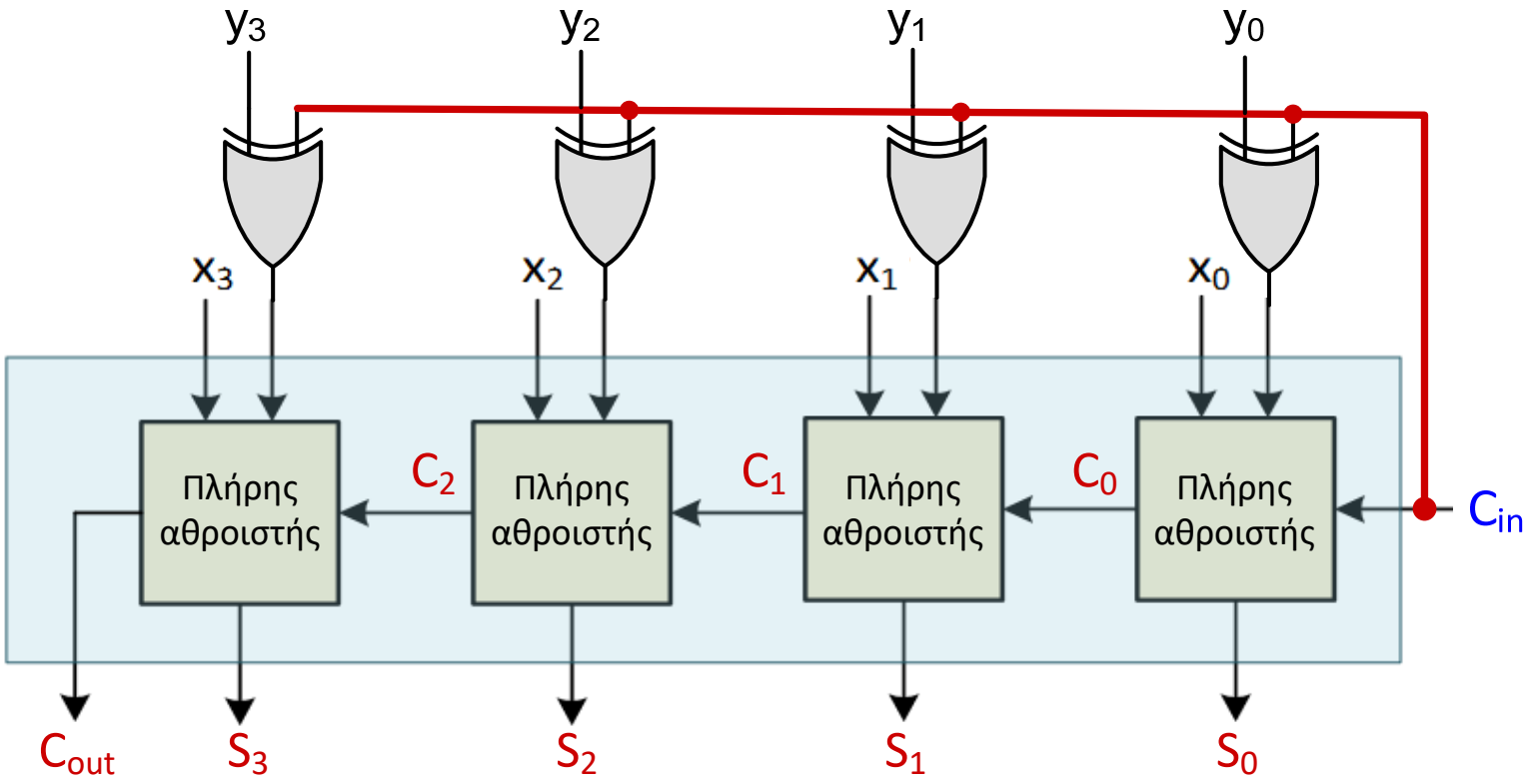
**Παράδειγμα:** Υλοποίηση παράλληλου αθροιστή 8 ψηφίων με χρήση δύο παράλληλων αθροιστών 4 ψηφίων.

Η ζητούμενη υλοποίηση επιτυγχάνεται με την τροφοδότηση του κρατουμένου εισόδου του δεύτερου παράλληλου αθροιστή 4 ψηφίων με το κρατούμενο εξόδου του πρώτου παράλληλου αθροιστή 4 ψηφίων.

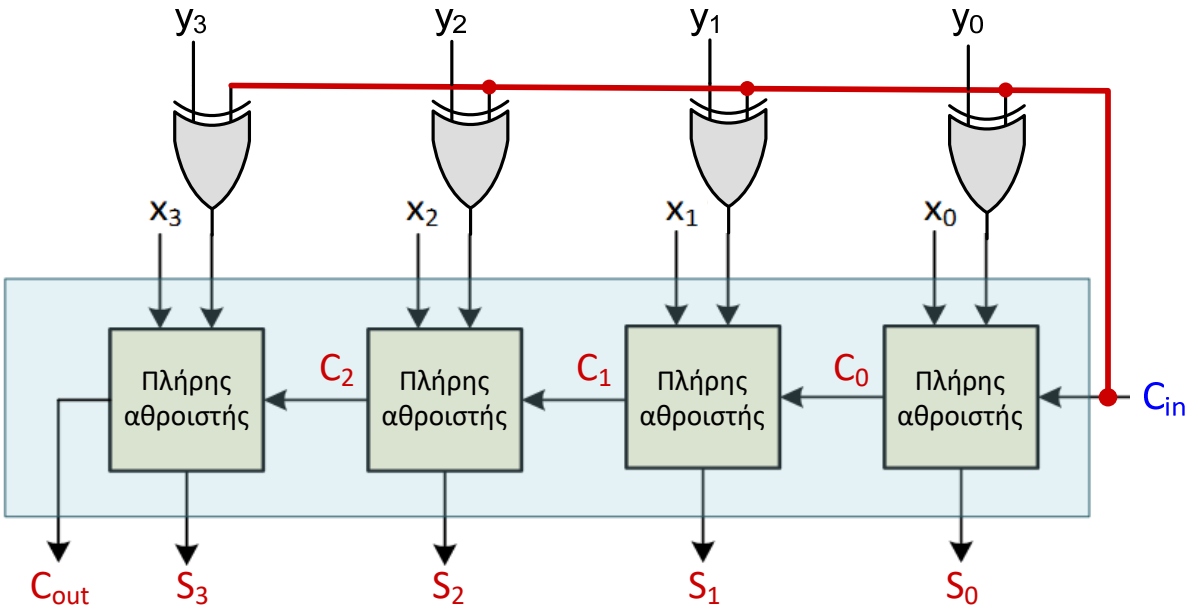


# Παράλληλος αθροιστής / αφαιρέτης

Ένας παράλληλος αθροιστής / αφαιρέτης μπορεί να σχεδιαστεί με προσθήκη σε έναν παράλληλο αθροιστή ισάριθμων πυλών XOR με το πλήθος των ψηφίων των αριθμών που προστίθενται ή αφαιρούνται.



# Παράλληλος αθροιστής / αφαιρέτης




$X = x_3x_2x_1x_0$

$Y = y_3y_2y_1y_0$

$C_{in} = 0 \Rightarrow S = X + Y$

$C_{in} = 1 \Rightarrow S = X + Y' + 1$

$S = S_3S_2S_1S_0$

Τροφοδοτούμε τη μία είσοδο κάθε πύλης XOR με το αντίστοιχο ψηφίο του δεύτερου αριθμού ($y_i$), ενώ την άλλη είσοδό της την τροφοδοτούμε με το κρατούμενο εισόδου ($C_{in}$) της λιγότερο σημαντικής θέσης.

Όταν $C_{in} = 0$, η έξοδος κάθε πύλης XOR είναι $y_i$ ($S = X + Y$), ενώ όταν $C_{in} = 1$, η έξοδος κάθε πύλης XOR είναι $y'_i$ και το αποτέλεσμα της πράξης είναι το άθροισμα του X με το συμπλήρωμα ως προς 2 του Y ($S = X + Y' + 1 = X + \Sigma_2 Y$).

# Παράλληλος αθροιστής / αφαιρέτης

Όταν $\mathbf{C_{in} = 0}$, το κύκλωμα λειτουργεί ως παράλληλος αθροιστής και εκτελεί την πρόσθεση $X + Y = x_3x_2x_1x_0 + y_3y_2y_1y_0$.

Για μη προσημασμένους αριθμούς (X, Y), το κύκλωμα έχει ως αποτέλεσμα $S = S_3S_2S_1S_0 = X + Y$ και κρατούμενο $C_{out}$.

Για προσημασμένους αριθμούς, το κύκλωμα έχει ως αποτέλεσμα $S = S_3S_2S_1S_0 = X + Y$ εφόσον δε συμβαίνει υπερχείλιση (overflow).

Όταν $\mathbf{C_{in} = 1}$, το κύκλωμα λειτουργεί ως παράλληλος αφαιρέτης, αφού εκτελεί την πρόσθεση $X + Y' + 1 = x_3x_2x_1x_0 + y'_3y'_2y'_1y'_0 + 1$, στην οποία ανάγεται η πράξη της αφαίρεσης.

Για μη προσημασμένους αριθμούς το κύκλωμα έχει ως αποτέλεσμα $S = S_3S_2S_1S_0 = X - Y$ όταν $X \geq Y$ και $S = S_3S_2S_1S_0 = \Sigma_2(Y - X)$ όταν $X < Y$.

Για προσημασμένους αριθμούς, το κύκλωμα έχει ως αποτέλεσμα $S = S_3S_2S_1S_0 = X - Y$ εφόσον δε συμβαίνει υπερχείλιση.

# Παράλληλος αθροιστής / αφαιρέτης

Υπερχείλιση κατά την πρόσθεση ή την αφαίρεση δύο προσημασμένων αριθμών, σημαίνει ότι οι αριθμοί αποτελούνται από n ψηφία και για την παράσταση του αποτελέσματος της πράξης απαιτούνται n + 1 ψηφία.

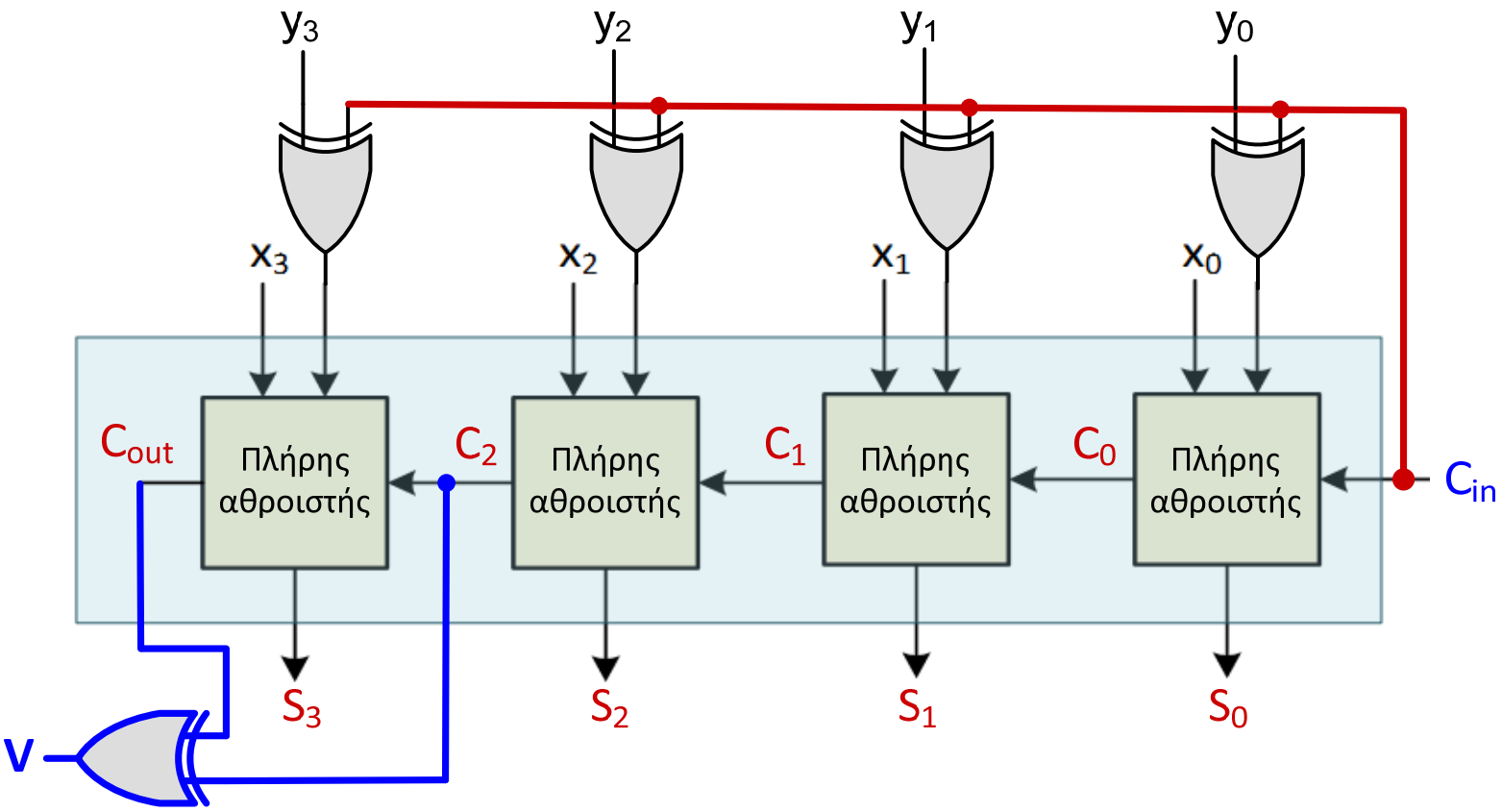
Στην περίπτωση αυτή μετατοπίζεται το ψηφίο-πρόσημο από την αναμενόμενη θέση.

Υπερχείλιση κατά την πρόσθεση (X + Y) δύο προσημασμένων αριθμών συμβαίνει όταν οι X και Y είναι ομόσημοι και το αποτέλεσμα της πράξης έχει διαφορετικό πρόσημο από τους προσθετέους X και Y.
Παράδειγμα: (+7) 0111 + (+5) 0101 = (−4) 1100 αντί για (+12) 01100.

Υπερχείλιση κατά την αφαίρεση (X – Y) δύο προσημασμένων αριθμών συμβαίνει όταν ο X είναι θετικός, ο Y είναι αρνητικός και το αποτέλεσμα της πράξης είναι αρνητικό, καθώς και όταν ο X είναι αρνητικός, ο Y είναι θετικός και το αποτέλεσμα της πράξης είναι θετικό.
Παράδειγμα: (−7) 1001 − (+5) 0101 = (−7) 1001 + (−5) 1011 = (+4) 0100 αντί για (−12) 10100.

# Παράλληλος αθροιστής / αφαιρέτης

Η υπερχείλιση ανιχνεύεται με την προσθήκη μιας πύλης XOR δύο εισόδων στο κύκλωμα του παράλληλου αθροιστή / αφαιρέτη.



# Παράλληλος αθροιστής / αφαιρέτης

Η ανίχνευση της υπερχείλισης γίνεται μέσω της διαπίστωσης ότι τα κρατούμενα ψηφία που προκύπτουν κατά την πράξη στις δύο πιο σημαντικές θέσεις λαμβάνουν διαφορετική τιμή ($V = C_2 \oplus C_{out} = 1$).

Αυτό προκύπτει από το αποτέλεσμα της πρόσθεσης των ψηφίων-προσήμων των δύο προσθετέων που εκτελείται στον πλήρη αθροιστή της πιο σημαντικής θέσης, όταν οι δύο προσθετέοι είναι ομόσημοι και το άθροισμα έχει διαφορετικό πρόσημο από τους προσθετέους:

$$0\ (x_3) + 0\ (y_3) + 1\ (C_2) = 1\ (S_3) + 0\ (C_{out})$$
$$1\ (x_3) + 1\ (y_3) + 0\ (C_2) = 0\ (S_3) + 1\ (C_{out})$$

Εάν $V = 0$ το αποτέλεσμα $S_3S_2S_1S_0$ είναι σωστό, ενώ εάν $V = 1$ συμβαίνει υπερχείλιση και το αποτέλεσμα της πράξης χρειάζεται 5 ψηφία για να παρασταθεί.

Στη δεύτερη περίπτωση ($V = 1$), η τιμή του $C_{out}$ εκφράζει το πρόσημο του αποτελέσματος, το οποίο έχει μετατοπιστεί.

# Δυαδικός πολλαπλασιαστής

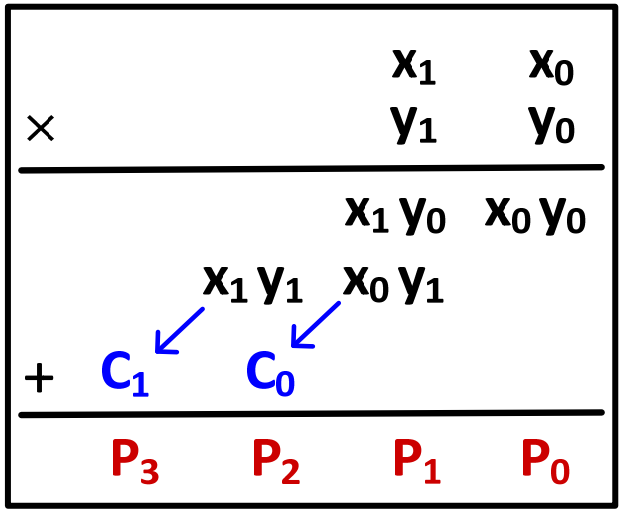
Το αριθμητικό γινόμενο δύο δυαδικών ψηφίων ταυτίζεται με το λογικό γινόμενό τους, επομένως υλοποιείται εύκολα με μία πύλη AND 2 εισόδων.

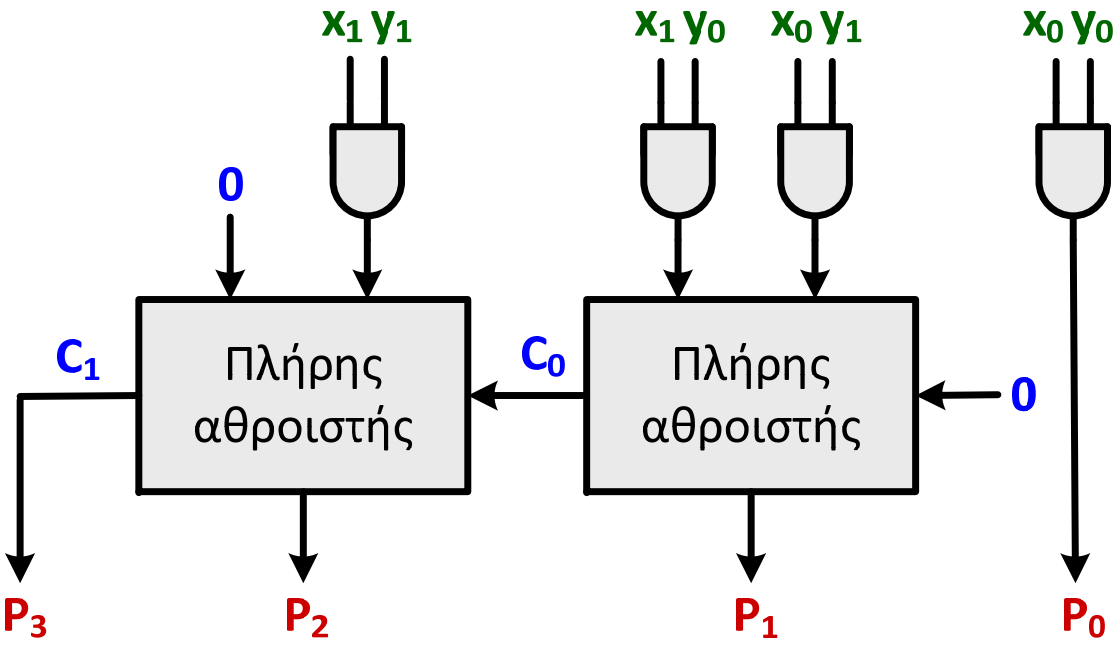
Κατά τον πολλαπλασιασμό αριθμών με περισσότερα δυαδικά ψηφία, απαιτείται η πρόσθεση των μερικών γινομένων 2 ψηφίων.

Για τους παραπάνω λόγους, η σύνθεση ενός πολλαπλασιαστή δυαδικών αριθμών, επιτυγχάνεται με χρήση πυλών AND 2 εισόδων και παράλληλων αθροιστών.

Για την υλοποίηση ενός πολλαπλασιαστή δύο μη προσημασμένων δυαδικών αριθμών 2 ψηφίων, απαιτούνται 4 πύλες AND 2 εισόδων για την εξαγωγή των ισάριθμων μερικών γινομένων και ένας παράλληλος αθροιστής 2 ψηφίων για την πρόσθεσή τους και την εξαγωγή του τελικού γινομένου.

# Δυαδικός πολλαπλασιαστής




Οι πλήρεις αθροιστές του παράλληλου αθροιστή που χρησιμοποιήθηκε, στον συγκεκριμένο πολλαπλασιαστή μπορούν να αντικατασταθούν με ημιαθροιστές, αφού εκτελούν πρόσθεση μεταξύ 2 ψηφίων.

# Συγκριτής μεγέθους αριθμών

Η σύγκριση δύο αριθμών είναι η πράξη που προσδιορίζει εάν ένας αριθμός είναι μεγαλύτερος από έναν άλλο αριθμό, μικρότερος από αυτόν ή ίσος με αυτόν.

Ο συγκριτής μεγέθους (magnitude comparator) αριθμών είναι συνδυαστικό κύκλωμα, το οποίο συγκρίνει δύο αριθμούς A και B και προσδιορίζει τη σχέση των μεγεθών τους.

Το αποτέλεσμα της σύγκρισης προκύπτει από την τιμή των τριών εξόδων του κυκλώματος G (A > B), E (A = B) και L (A < B), με τα ονόματα των τριών εξόδων να προκύπτουν από τα αρχικά γράμματα των λέξεων Greater, Equal και Lesser, αντίστοιχα.

Ο πίνακας αλήθειας του κυκλώματος σύγκρισης δύο αριθμών με n ψηφία ο καθένας, περιλαμβάνει $2^{2 \cdot n}$ γραμμές, με αποτέλεσμα να είναι δύσκολο να χρησιμοποιηθεί για τη σύνθεση του κυκλώματος σύγκρισης (παράδειγμα: για n = 3 προκύπτουν $2^6 = 64$ γραμμές).

# Συγκριτής μεγέθους αριθμών

Για το λόγο αυτό, ακολουθούμε μια πιο αποδοτική μέθοδο για τη σύνθεση του κυκλώματος σύγκρισης.

Έστω δύο αριθμοί Α και Β με πλήθος ψηφίων n = 4:

$$A = A_3\, A_2\, A_1\, A_0$$
$$B = B_3\, B_2\, B_1\, B_0$$

Οι δύο αριθμοί είναι ίσοι όταν:

$$A_3 = B_3, A_2 = B_2, A_1 = B_1, A_0 = B_0$$

Η σχέση ισότητας δύο ψηφίων μπορεί να εκφραστεί λογικά με τη συνάρτηση ισοδυναμίας (XNOR): $x_i = A_i B_i + A_i' B_i'$

$x_i$ = 1 όταν τα ψηφία αντίστοιχων θέσεων των δύο αριθμών είναι ίσα.

Για να είναι ίσοι οι αριθμοί Α και Β (δηλαδή για να ισχύει Ε = 1) πρέπει να όλες οι μεταβλητές $x_i$ να έχουν τιμή 1, δηλαδή προκύπτει ότι:

$$E = x_3 x_2 x_1 x_0$$

# Συγκριτής μεγέθους αριθμών

Για να διαπιστώσουμε εάν ο Α είναι μεγαλύτερος ή μικρότερος από τον Β, ελέγχουμε την τιμή των ψηφίων των αντίστοιχων θέσεων, ξεκινώντας από τα ψηφία της πιο σημαντικής θέσης.

Εάν τα δύο πιο σημαντικά ψηφία είναι ίσα, ελέγχουμε το ζεύγος ψηφίων της αμέσως λιγότερο σημαντικής θέσης.

Εάν το ψηφίο του Α είναι 1 και το αντίστοιχο του Β είναι 0, συμπεραίνουμε ότι Α > Β, ενώ εάν το ψηφίο του Α είναι 0 και το αντίστοιχο ψηφίο του Β είναι 1, συμπεραίνουμε ότι Α < Β.

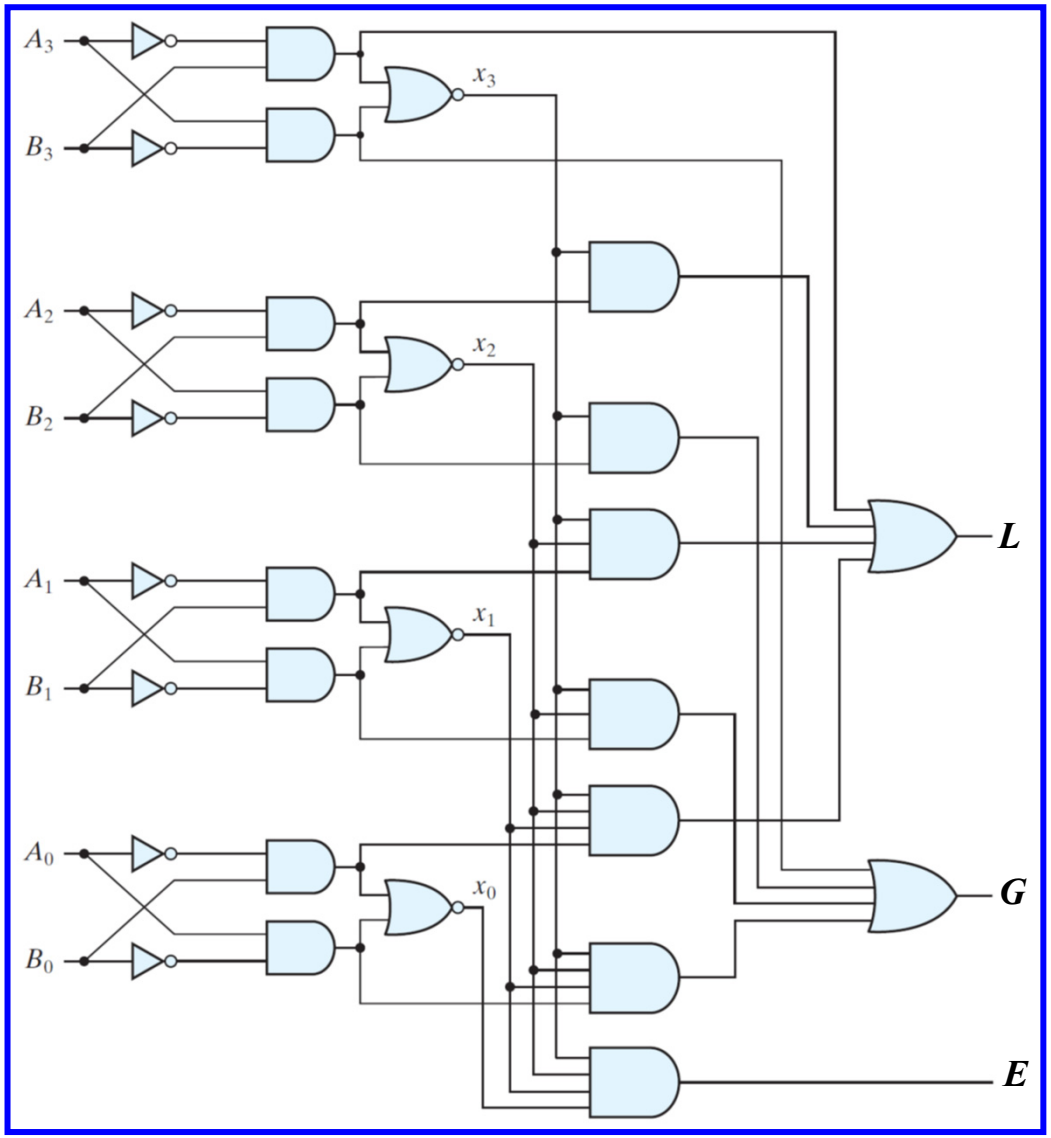
Η ακολουθιακή αυτή διαδικασία σύγκρισης των ψηφίων, εκφράζεται με δύο λογικές συναρτήσεις:

$$G = A_3B'_3 + x_3A_2B'_2 + x_3x_2A_1B'_1 + x_3x_2x_1A_0B'_0$$
$$L = A'_3B_3 + x_3A'_2B_2 + x_3x_2A'_1B_1 + x_3x_2x_1A'_0B_0$$

# Συγκριτής μεγέθους αριθμών

Κατά την υλοποίηση των συναρτήσεων E, G και L γίνεται προσπάθεια επαναχρησιμοποίησης λογικών γινομένων, ώστε να καταλήξουμε σε κύκλωμα μειωμένου κόστους.

Η σύγκριση μη προσημασμένων, αλλά και προσημασμένων αριθμών μπορεί να επιτευχθεί και μέσω της της αφαίρεσής τους



# Κωδικοποιητές

Τα ψηφιακά συστήματα έχουν τη δυνατότητα να χειρίζονται διακριτά στοιχεία πληροφορίας που ανήκουν σε ένα πεπερασμένο σύνολο, αφού το καθένα από αυτά μπορεί να παρασταθεί με μία ακολουθία δυαδικών ψηφίων, ακολουθώντας τους κανόνες ενός δυαδικού κώδικα.

Κάθε διαφορετικό στοιχείο πληροφορίας αντιστοιχίζεται σε μία μοναδική ακολουθία ή συνδυασμό δυαδικών ψηφίων.

Για την παράσταση ενός συνόλου $2^N$ στοιχείων πληροφορίας, όπως, για παράδειγμα, αριθμών ή γραμμάτων, απαιτείται δυαδικός κώδικας που να χρησιμοποιεί τουλάχιστον Ν δυαδικά ψηφία.

Οι κωδικοποιητές (encoders) είναι συνδυαστικά κυκλώματα τα οποία παράγουν στην έξοδό τους ψηφιακά δεδομένα σε πιο συμπαγή μορφή από εκείνη των δεδομένων που λαμβάνουν στην είσοδό τους.

# Κωδικοποιητές

Ένας κωδικοποιητής, λαμβάνει στην είσοδό του **έως** $2^N$ δυαδικά ψηφία και παράγει στην έξοδό του Ν δυαδικά ψηφία.

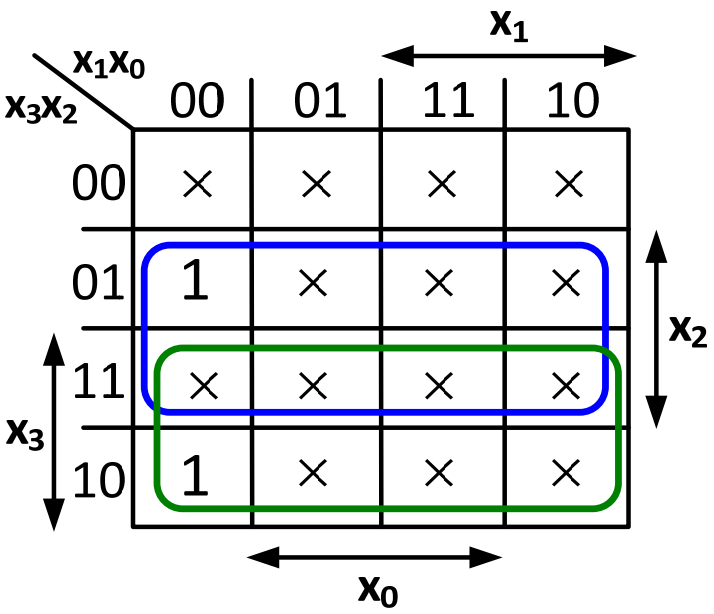
Λόγω του ότι το πλήθος των ψηφίων εξόδου δεν επαρκεί για την παράσταση του μεγαλύτερου πλήθους των δεδομένων εισόδου, μόνο ένα από τα ψηφία εισόδου μπορεί να έχει λογική τιμή 1, έτσι ώστε το κύκλωμα να παράγει στην έξοδό του ένα δυαδικό αριθμό Ν ψηφίων, που αντιστοιχεί στο ψηφίο της εισόδου με λογική τιμή 1.

# Απλός κωδικοποιητής 4-σε-2

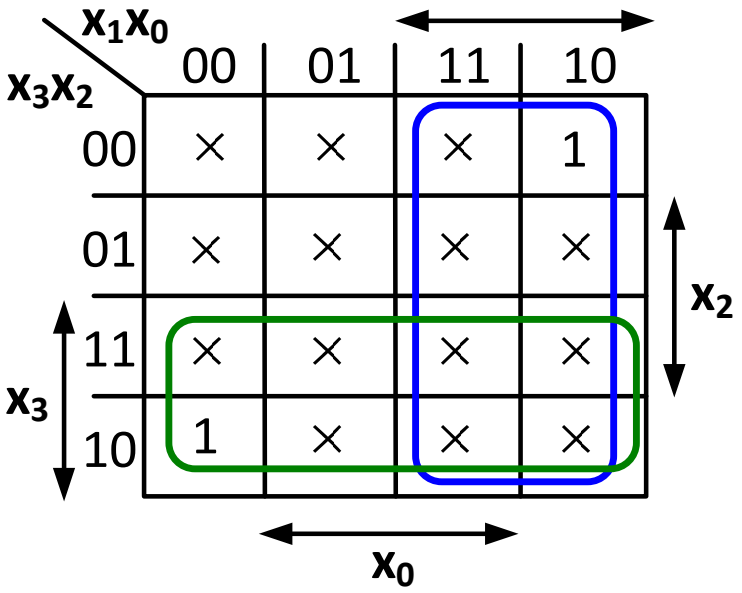

| $x_3$ | $x_2$ | $x_1$ | $x_0$ | $y_1$ | $y_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | × | × |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | × | × |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | × | × |
| 0 | 1 | 1 | 0 | × | × |
| 0 | 1 | 1 | 1 | × | × |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | × | × |
| 1 | 0 | 1 | 0 | × | × |
| 1 | 0 | 1 | 1 | × | × |
| 1 | 1 | 0 | 0 | × | × |
| 1 | 1 | 0 | 1 | × | × |
| 1 | 1 | 1 | 0 | × | × |
| 1 | 1 | 1 | 1 | × | × |

Αφού μόνο ένα από τα ψηφία εισόδου μπορεί να έχει λογική τιμή 1, 12 από τους συνολικά 16 συνδυασμούς εισόδων δεν επιτρέπονται και οι αντίστοιχοι ελαχιστόροι αποτελούν αδιάφορους όρους.

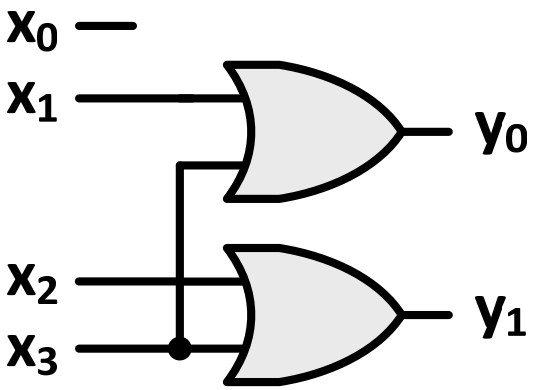
# Απλός κωδικοποιητής 4-σε-2




$y_1 = x_2 + x_3$




$y_0 = x_1 + x_3$



# Απλός κωδικοποιητής 4-σε-2

Στη λειτουργία του απλού κωδικοποιητή εντοπίζονται 2 προβλήματα.

Το πρώτο αφορά το γεγονός ότι οι έξοδοι του κυκλώματος που υλοποιήθηκε μηδενίζονται, όταν όλες οι είσοδοι έχουν λογική τιμή 0, με αποτέλεσμα αυτός ο συνδυασμός τιμών των εισόδων να μην μπορεί να διακριθεί από την περίπτωση όπου μόνο η είσοδος $x_0$ λογική τιμή 1.

Το δεύτερο πρόβλημα οφείλεται στο ότι δεν προβλέπονται οι περιπτώσεις όπου περισσότερες από μία εισόδους έχουν τιμή 1.

Το πρώτο πρόβλημα αντιμετωπίζεται με την προσθήκη μιας επιπλέον εξόδου εγκυρότητας (V) που λαμβάνει τιμή 0 όταν όλες οι είσοδοι είναι 0 και τιμή 1 σε όλες τις υπόλοιπες περιπτώσεις.

Οι υπόλοιπες δύο έξοδοι δεν ορίζονται όταν μηδενίζεται η V, συνεπώς ο συνδυασμός μηδενικών εισόδων αποτελεί αδιάφορη λογική συνθήκη για τις εξόδους αυτές.

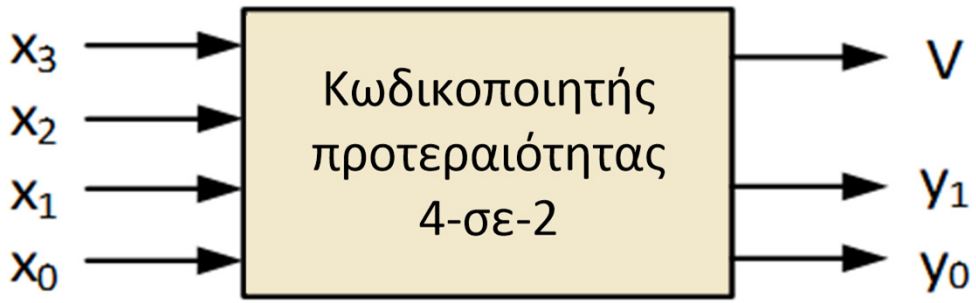
# Κωδικοποιητής προτεραιότητας 4-σε-2

Το δεύτερο πρόβλημα λύνεται εάν ο κωδικοποιητής τροποποιηθεί ώστε να υποστηρίζει προκαθορισμένη προτεραιότητα εισόδων και τότε αναφέρεται ως κωδικοποιητής προτεραιότητας (priority encoder).

Έτσι όταν η τιμή περισσότερων εισόδων είναι 1, η έξοδος του κυκλώματος καθορίζεται από την είσοδο με τη μεγαλύτερη προτεραιότητα.

Για παράδειγμα, μπορεί να καθοριστεί ως είσοδος με τη μεγαλύτερη προτεραιότητα η είσοδος $x_3$ και να ακολουθούν σε σειρά προτεραιότητας οι είσοδοι $x_2$, $x_1$ και $x_0$.

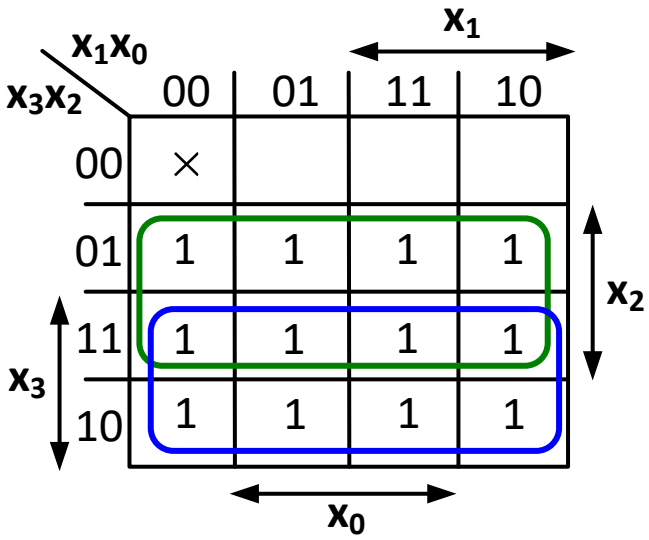
# Κωδικοποιητής προτεραιότητας 4-σε-2



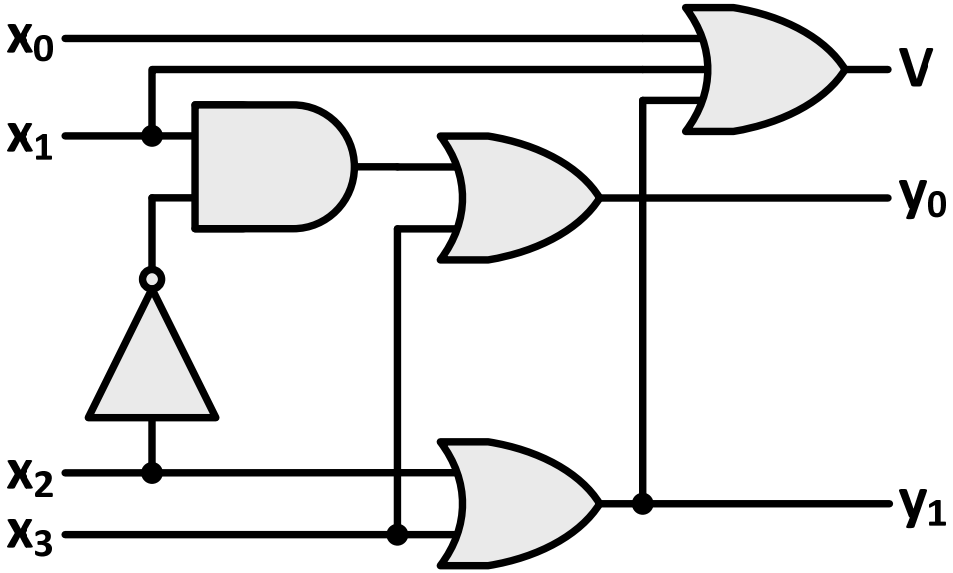

| $x_3$ | $x_2$ | $x_1$ | $x_0$ | $y_1$ | $y_0$ | V |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X | X | 0 |
| 0 | 0 | 0 | **1** | **0** | **0** | 1 |
| 0 | 0 | **1** | 0 | **0** | **1** | 1 |
| 0 | 0 | **1** | 1 | 0 | 1 | 1 |
| 0 | **1** | 0 | 0 | **1** | **0** | 1 |
| 0 | **1** | 0 | 1 | 1 | 0 | 1 |
| 0 | **1** | 1 | 0 | 1 | 0 | 1 |
| 0 | **1** | 1 | 1 | 1 | 0 | 1 |
| **1** | 0 | 0 | 0 | **1** | **1** | 1 |
| **1** | 0 | 0 | 1 | 1 | 1 | 1 |
| **1** | 0 | 1 | 0 | 1 | 1 | 1 |
| **1** | 0 | 1 | 1 | 1 | 1 | 1 |
| **1** | 1 | 0 | 0 | 1 | 1 | 1 |
| **1** | 1 | 0 | 1 | 1 | 1 | 1 |
| **1** | 1 | 1 | 0 | 1 | 1 | 1 |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 |

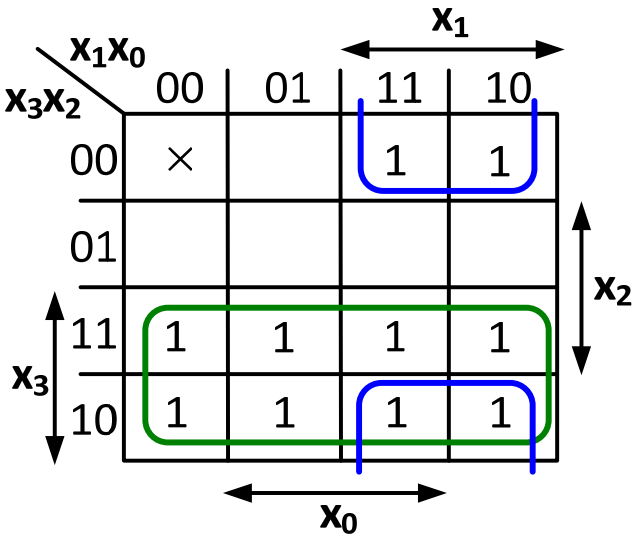
$$V' = x'_3x'_2x'_1x'_0 \Rightarrow V = x_3 + x_2 + x_1 + x_0$$

# Κωδικοποιητής προτεραιότητας 4-σε-2




$y_1 = x_2 + x_3$




$y_0 = x_3 + x_1x'_2$

# Παράδειγμα απλού κωδικοποιητή

## Σύνθεση κωδικοποιητή των δεκαδικών ψηφίων σε δυαδικούς αριθμούς

Ο ζητούμενος κωδικοποιητής διαθέτει 10 εισόδους, μία για καθένα από τα 10 ψηφία (0 έως 9) και 4 εξόδους στις οποίες λαμβάνονται οι αντίστοιχοι δυαδικοί αριθμοί. Επομένως, πρόκειται για απλό κωδικοποιητή 10-σε-4.

Λόγω του ότι το πλήθος των ψηφίων εξόδου δεν επαρκεί για την παράσταση του μεγαλύτερου πλήθους των δεδομένων εισόδου, μόνο ένα από τα ψηφία εισόδου μπορεί να έχει λογική τιμή 1, έτσι ώστε το κύκλωμα να παράγει στην έξοδό του ένα δυαδικό αριθμό 4 ψηφίων, που αντιστοιχεί στο δεκαδικό ψηφίο της εισόδου με λογική τιμή 1.

# Παράδειγμα απλού κωδικοποιητή

| $X_9$ | $X_8$ | $X_7$ | $X_6$ | $X_5$ | $X_4$ | $X_3$ | $X_2$ | $X_1$ | $X_0$ | $y_3$ | $y_2$ | $y_1$ | $y_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 0 |
| 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 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 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 |

$y_3 = x_8 + x_9$

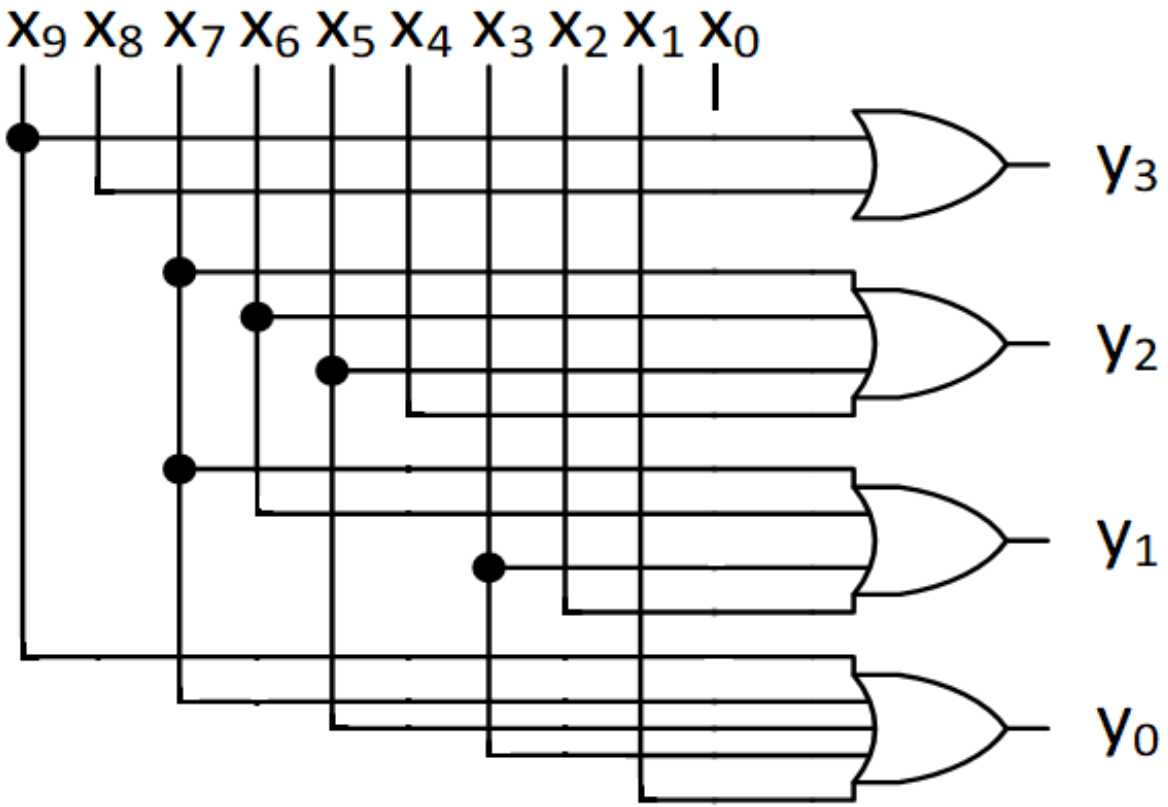
$y_2 = x_4 + x_5 + x_6 + x_7$

$y_1 = x_2 + x_3 + x_6 + x_7$

$y_0 = x_1 + x_3 + x_5 + x_7 + x_9$

Αφού μόνο ένα από τα ψηφία εισόδου μπορεί να έχει λογική τιμή 1, 1014 από τους συνολικά 1024 (= $2^{10}$) συνδυασμούς εισόδων δεν επιτρέπονται και οι αντίστοιχοι ελαχιστόροι είναι αδιάφοροι όροι.

# Παράδειγμα απλού κωδικοποιητή



# Αποκωδικοποιητές

Οι αποκωδικοποιητές (decoders) είναι συνδυαστικά κυκλώματα τα οποία εκτελούν λειτουργία αντίστροφη από εκείνη των κωδικοποιητών.

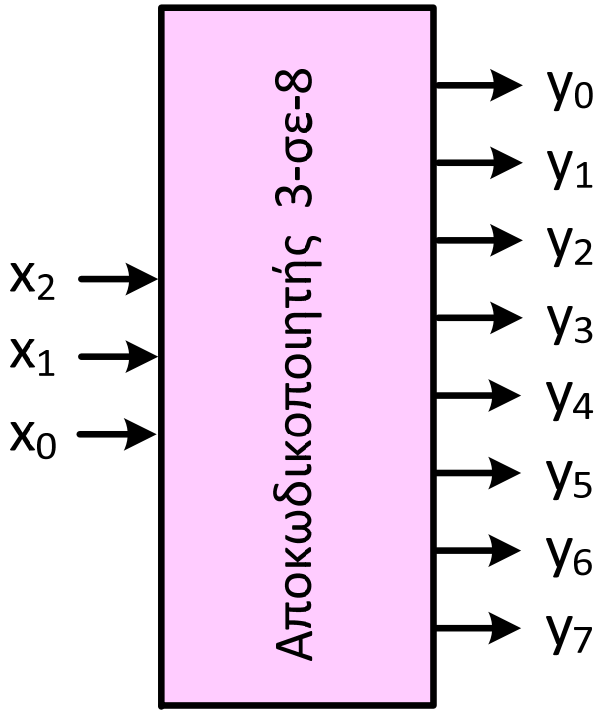
Λαμβάνουν στην είσοδό τους N δυαδικά ψηφία και παράγουν στην έξοδό τους **έως** $2^N$ δυαδικά ψηφία.

Για κάθε συνδυασμό λογικών τιμών εισόδου, μόνο το ψηφίο εξόδου που αντιστοιχεί σε αυτόν λαμβάνει λογική τιμή 1.

Ουσιαστικά, ένας αποκωδικοποιητής N-σε-$2^N$ αποτελεί γεννήτρια ελαχιστόρων, αφού κάθε έξοδος αντιστοιχεί σε έναν ελαχιστόρο N μεταβλητών εισόδου.

Το λογικό κύκλωμα του αποκωδικοποιητή σχεδιάζεται με βάση τον πίνακα αλήθειας, χρησιμοποιώντας αντιστροφείς για την παραγωγή των συμπληρωματικών μορφών των εισόδων και λογικές πύλες AND για την παραγωγή των ελαχιστόρων.

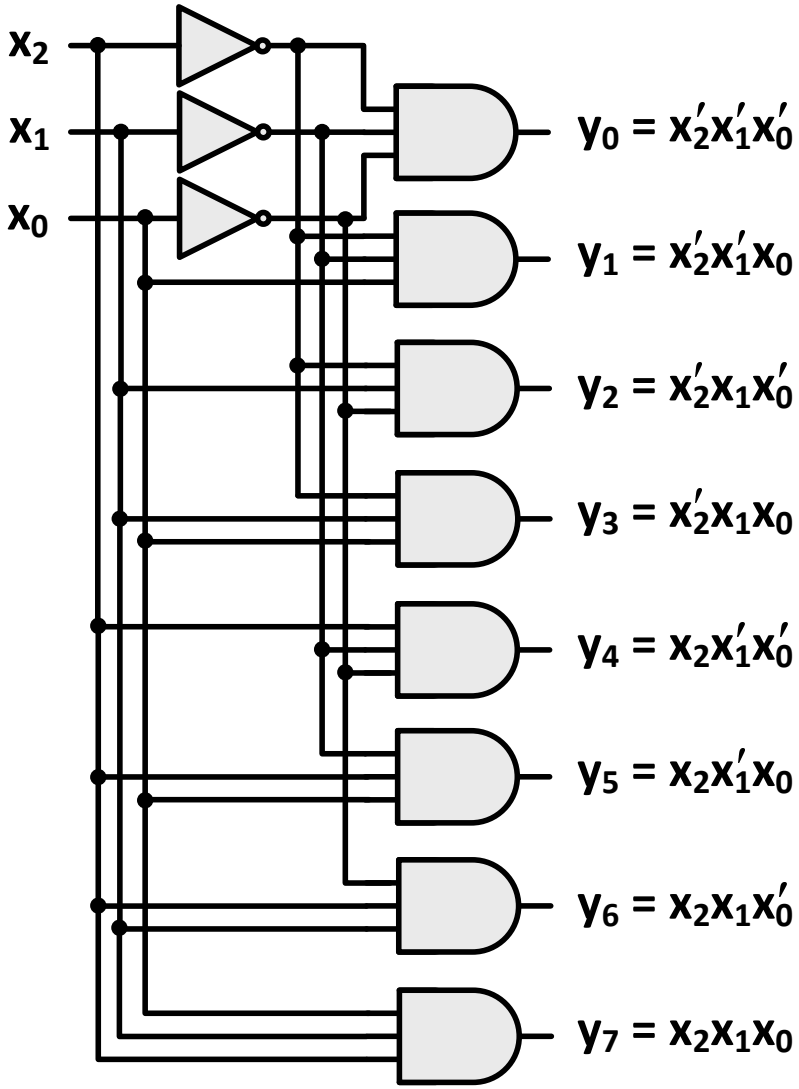
# Αποκωδικοποιητής 3-σε-8




| $x_2$ | $x_1$ | $x_0$ | $y_0$ | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |

Για κάθε συνδυασμό τιμών των μεταβλητών εισόδου, μόνο μία έξοδος λαμβάνει λογική τιμή 1 και παράγει τον αντίστοιχο ελαχιστόρο.

# Αποκωδικοποιητής 3-σε-8



# Υλοποίηση λογικών συναρτήσεων με αποκωδικοποιητή

Αφού ένας αποκωδικοποιητής N-σε-$2^N$ αποτελεί και γεννήτρια ελαχιστόρων, προκύπτει ότι συνδυάζοντάς τον με μία λογική πύλη OR, η οποία παράγει το λογικό άθροισμα κατάλληλων εξόδων του, μπορούμε να υλοποιήσουμε οποιαδήποτε λογική συνάρτηση σε μορφή αθροίσματος ελαχιστόρων.

Για να γίνει αυτό, πρέπει το πλήθος των εισόδων του αποκωδικοποιητή να ισούται με το πλήθος των μεταβλητών της συνάρτησης και το πλήθος των εισόδων της πύλης OR να ισούται με το πλήθος των ελαχιστόρων που συμμετέχουν στη συνάρτηση.

# Υλοποίηση λογικών συναρτήσεων με αποκωδικοποιητή

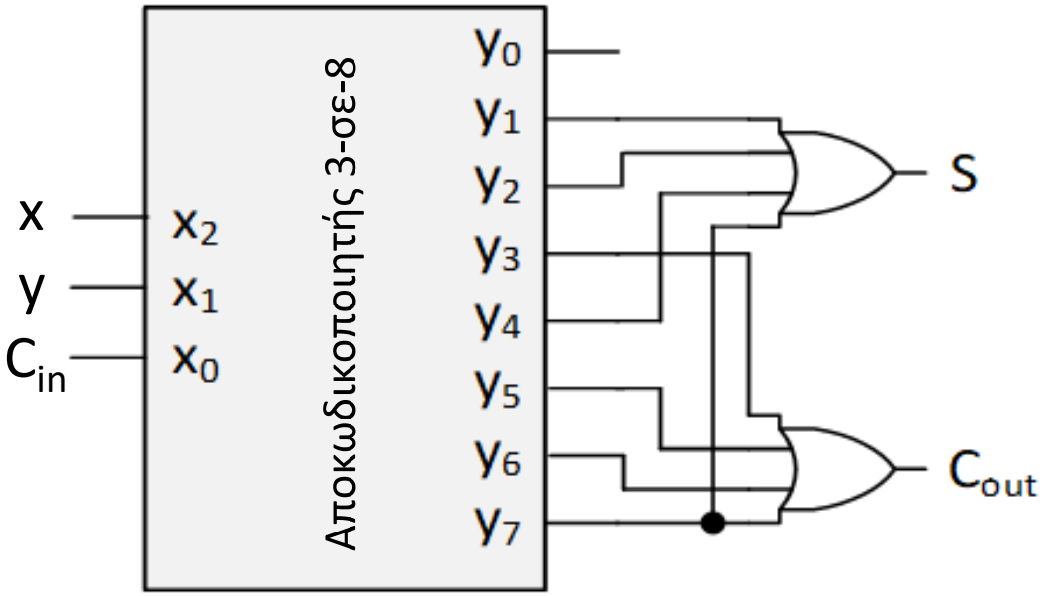
**Παράδειγμα:** Υλοποίηση πλήρους αθροιστή (συνάρτηση αθροίσματος και συνάρτηση κρατουμένου εξόδου) με κατάλληλο αποκωδικοποιητή και πύλες OR.

Ο πλήρης αθροιστής έχει 3 εισόδους (x, y, $C_{in}$) και 2 εξόδους ($C_{out}$, S).

Επομένως για την υλοποίησή του απαιτούνται ένας αποκωδικοποιητής 3-σε-8 και 2 πύλες OR.

Λογικές συναρτήσεις των 2 εξόδων του πλήρους αθροιστή:

$S = \Sigma(1,2,4,7)$ και $C_{out} = \Sigma(3,5,6,7)$



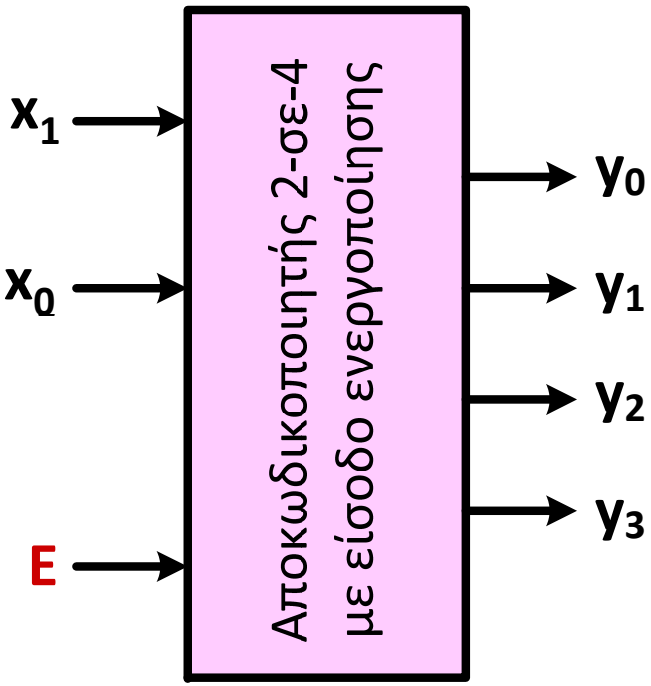
# Αποκωδικοποιητής με είσοδο ενεργοποίησης

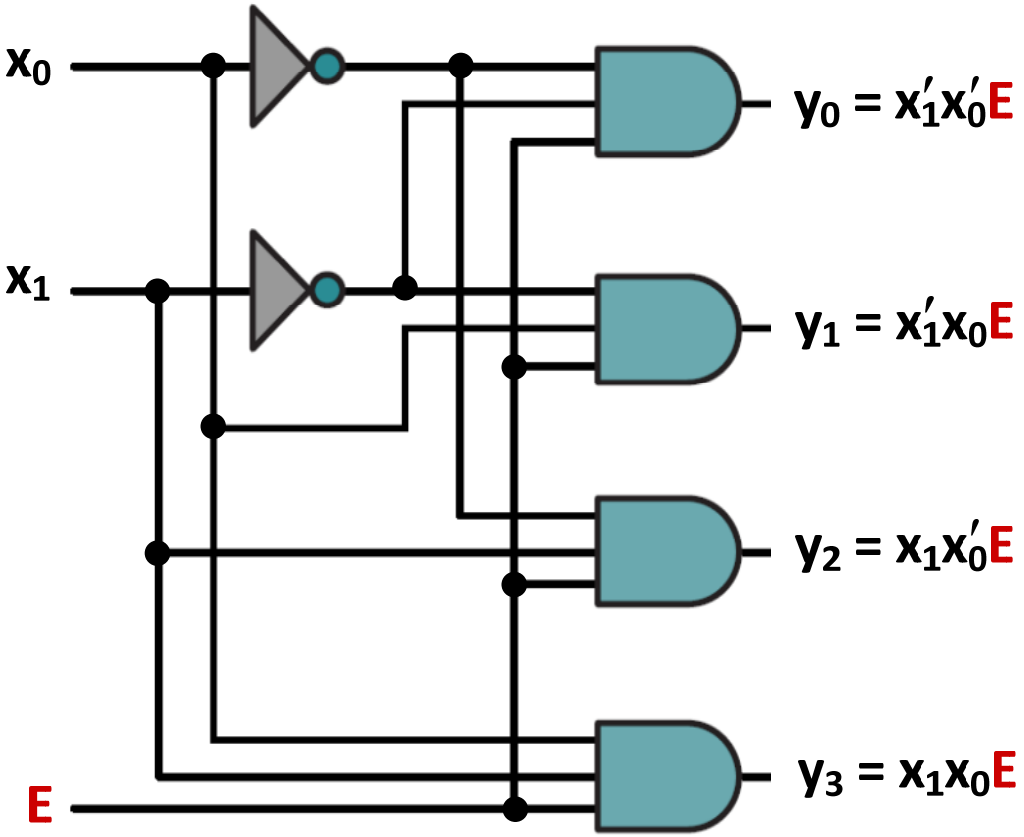
Εκτός από τα ψηφία εισόδου που πρόκειται να αποκωδικοποιηθούν, το κύκλωμα ενός αποκωδικοποιητή μπορεί να περιλαμβάνει μία ακόμη είσοδο ενεργοποίησης (enable input) E, ανάλογα με την τιμή της οποίας να επιτρέπεται ή όχι η αποκωδικοποίηση.

Η προσθήκη αυτή προϋποθέτει τη χρήση πυλών AND με μία επιπλέον είσοδο.

Όταν E = 0 οι έξοδοι του κυκλώματος μηδενίζονται, ενώ όταν E = 1 οι είσοδοί του αποκωδικοποιούνται με βάση τον πίνακα αλήθειας που περιγράφει τη λειτουργία του.

# Αποκωδικοποιητής με είσοδο ενεργοποίησης



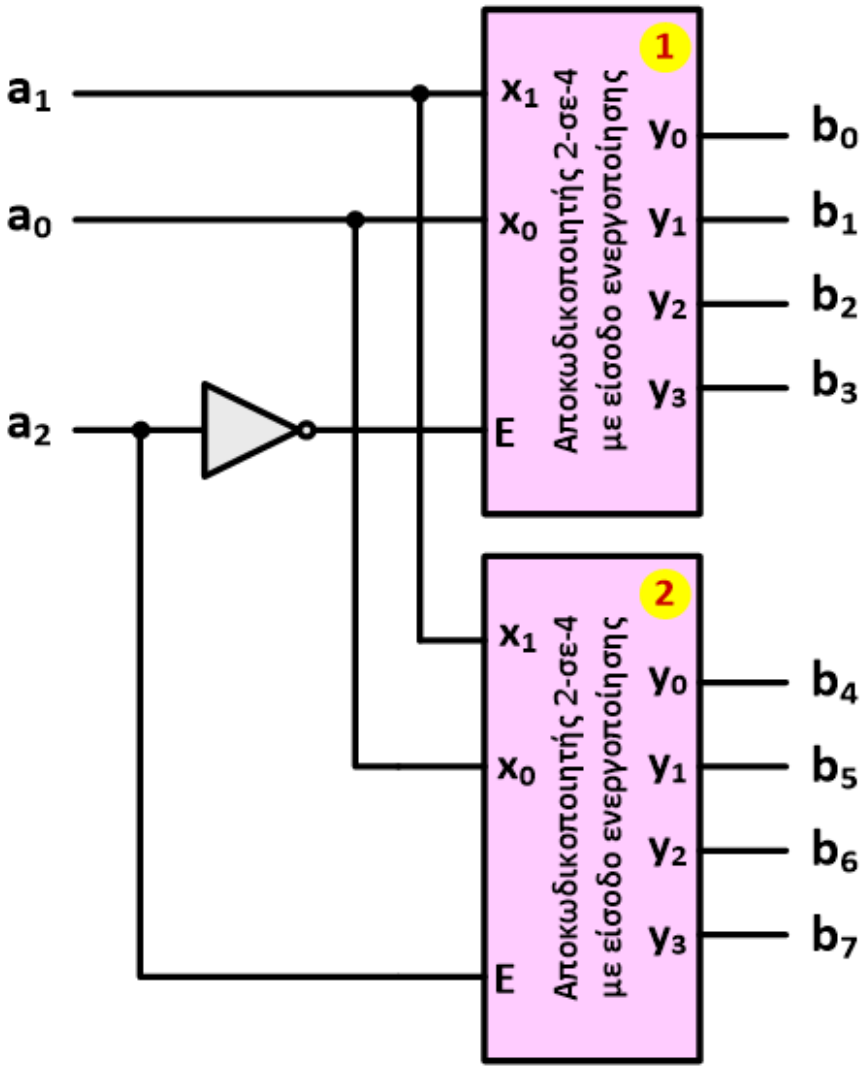



# Σύνθεση πολύπλοκων αποκωδικοποιητών

Η είσοδος Ε δίνει τη δυνατότητα συνδυασμού αποκωδικοποιητών για τη σύνθεση αποκωδικοποιητών με περισσότερες εισόδους.

Στη σχεδίαση αποκωδικοποιητή 3-σε-8 με 2 αποκωδικοποιητές 2-σε-4, όταν $a_2 = 0$, ενεργοποιείται ο πρώτος αποκωδικοποιητής 2-σε-4 και παράγει στις εξόδους $b_0$ έως $b_3$ τους ελαχιστόρους $m_0$ έως $m_3$.

Όταν $a_2 = 1$, ενεργοποιείται ο δεύτερος αποκωδικοποιητής 2-σε-4 και παράγει στις εξόδους $b_4$ έως $b_7$ τους ελαχιστόρους $m_4$ έως $m_7$.



# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Μια χρήσιμη εφαρμογή των αποκωδικοποιητών είναι η μετατροπή αλφαριθμητικών χαρακτήρων σε μορφή κατάλληλη για προβολή σε ηλεκτρονικές ψηφιακές συσκευές, όπως είναι οι αριθμομηχανές και τα ψηφιακά ρολόγια.

Σε τέτοιες συσκευές χρησιμοποιούνται συνήθως ενδείκτες 7 τμημάτων (7-segment displays) που αποτελούνται από 7 διόδους φωτοεκπομπής.

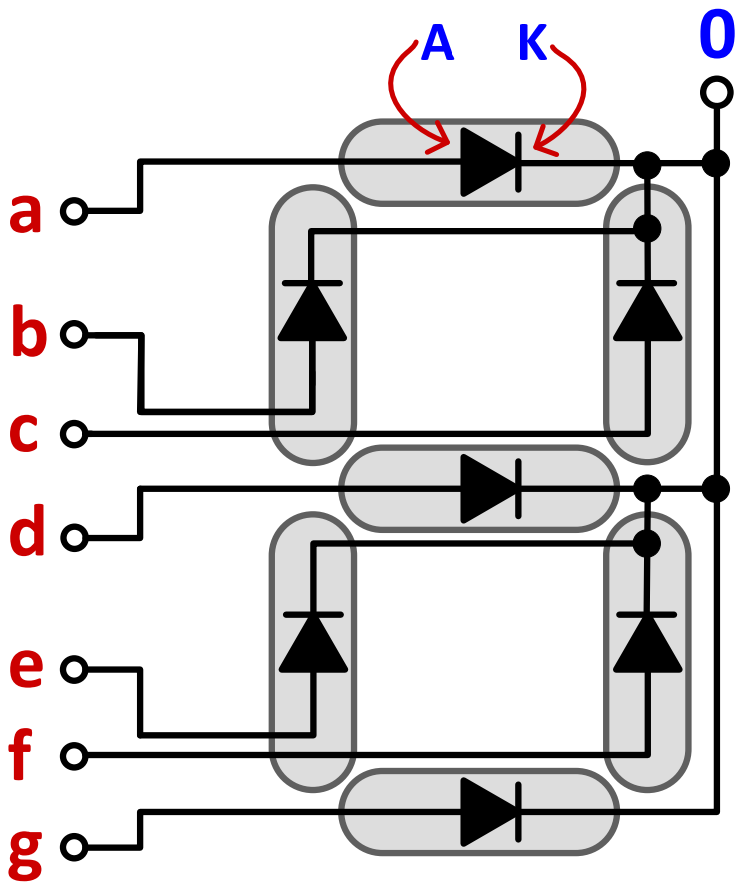
Κάθε τμήμα του ενδείκτη ενεργοποιείται (φωτοβολεί) ανεξάρτητα και ο ενδείκτης απεικονίζει δεκαδικά ψηφία ή/και άλλους χαρακτήρες.

# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Καθεμία από τις διόδους φωτοβολεί, όταν στην άνοδό (Α) εφαρμόζεται επαρκώς μεγαλύτερη τάση από εκείνη που εφαρμόζεται στην κάθοδο (Κ).

Οι κάθοδοι των διόδων του ενδείκτη 7 τμημάτων τροφοδοτούνται με λογική τιμή 0, που αντιστοιχεί στη χαμηλή στάθμη τάσης (γείωση).

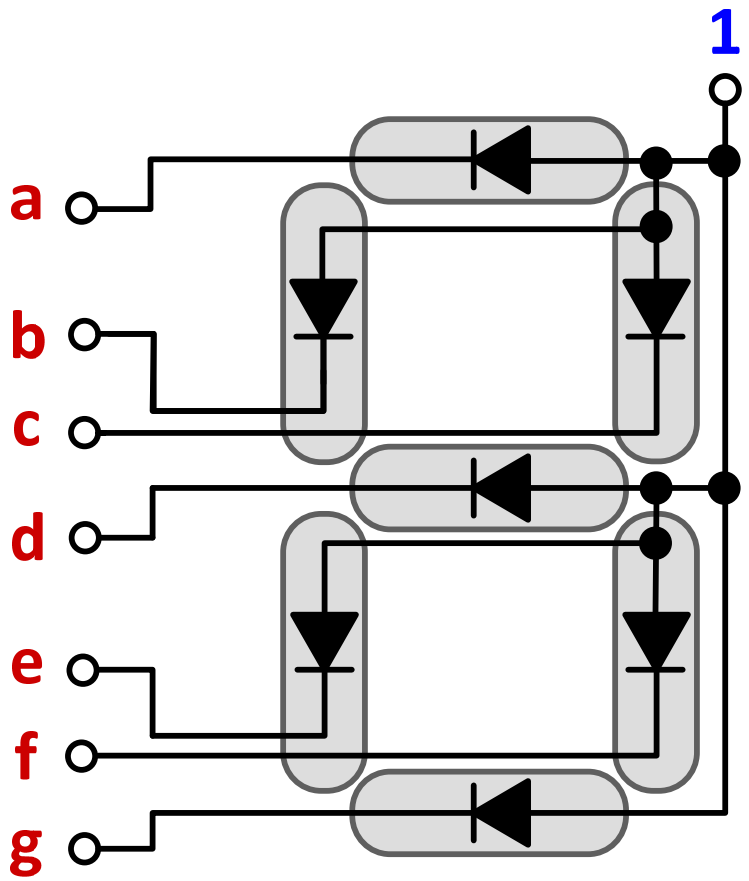
Αυτό έχει ως αποτέλεσμα, όταν η άνοδος μίας διόδου τροφοδοτηθεί με λογική τιμή 1 (που αντιστοιχεί στην υψηλή στάθμη τάσης), αυτή να φωτοβολεί και το αντίστοιχο τμήμα του ενδείκτη να «ανάβει».



# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Εναλλακτικά, ο ενδείκτης μπορεί να δομηθεί έτσι ώστε οι άνοδοι των διόδων των τμημάτων του να τροφοδοτούνται μόνιμα με λογικό 1, που αντιστοιχεί στην υψηλή στάθμη τάσης.

Αυτό έχει ως αποτέλεσμα, όταν η κάθοδος μίας διόδου τροφοδοτηθεί με λογική τιμή 0 (που αντιστοιχεί στη χαμηλή στάθμη τάσης), αυτή να φωτοβολεί και το αντίστοιχο τμήμα του ενδείκτη να «ανάβει».



# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

**Παράδειγμα 1:** Κατάστρωση του πίνακα αλήθειας αποκωδικοποιητή των δεκαδικών ψηφίων BCD σε ενδείκτη 7 τμημάτων και μελέτη επιλογών υλοποίησής του.

Ο κώδικας BCD κωδικοποιεί τα δεκαδικά ψηφία 0 έως 9 σε δυαδική μορφή, χρησιμοποιώντας 4 δυαδικά ψηφία για κάθε δεκαδικό ψηφίο.

Επομένως, ο ζητούμενος αποκωδικοποιητής έχει 4 εισόδους και 7 εξόδους (για την τροφοδότηση των 7 τμημάτων του ενδείκτη)

| Δεκαδικός | $x_3$ | $x_2$ | $x_1$ | $x_0$ | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 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 |

Οι τιμές των εξόδων για τους υπόλοιπους 6 συνδυασμούς των εισόδων είναι αδιάφορες.

# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Για την υλοποίηση του ζητούμενου αποκωδικοποιητή, έχουμε 2 επιλογές.

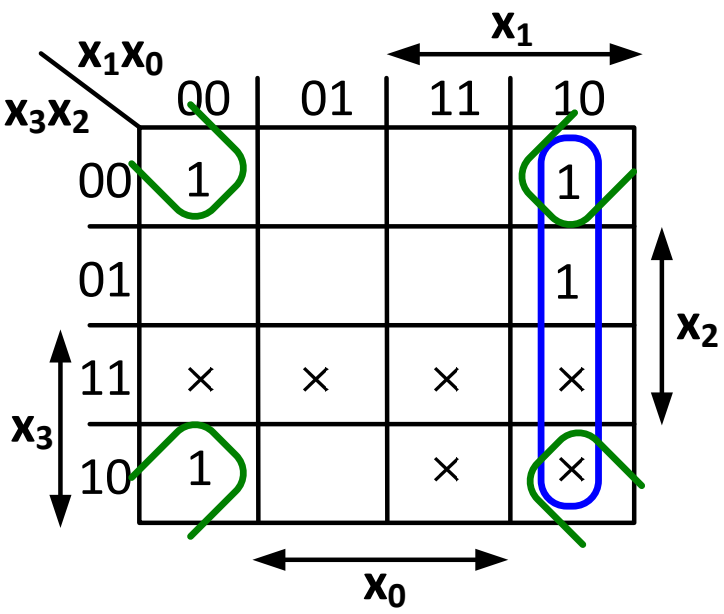
Η πρώτη επιλογή είναι να καταστρώσουμε για τις 7 εξόδους (a, b, c, d, e, f, g) ισάριθμους χάρτες Karnaugh, να εξάγουμε τις ελαχιστοποιημένες λογικές συναρτήσεις τους και να τις υλοποιήσουμε με τον μικρότερο δυνατό αριθμό λογικών πυλών.

Η δεύτερη επιλογή είναι να χρησιμοποιήσουμε έναν τυποποιημένο αποκωδικοποιητή 4-σε-16 (ή να συνθέσουμε έναν αποκωδικοποιητή 4-σε-10) στις εξόδους του οποίου λαμβάνουμε τους ελαχιστόρους που αντιστοιχούν στους 10 συνδυασμούς των 4 εισόδων που κωδικοποιούν τα 10 δεκαδικά ψηφία.

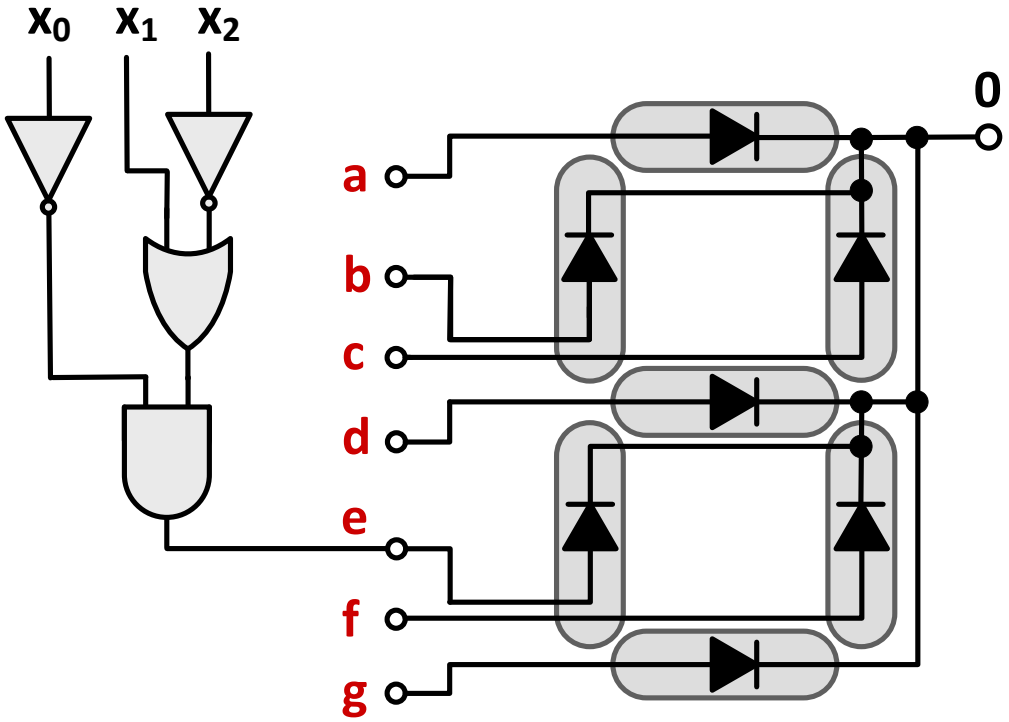
Στη συνέχεια, για την παραγωγή καθεμιάς από τις 7 εξόδους (a, b, c, d, e, f, g) χρησιμοποιούμε μια πύλη OR, ώστε να λάβουμε το άθροισμα ελαχιστόρων που αντιστοιχεί σε κάθε έξοδο, σύμφωνα με τον πίνακα αλήθειας.

# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Υλοποίηση της συνάρτησης της εξόδου e με την πρώτη επιλογή.




$$e = x_1x'_0 + x'_2x'_0 = (x_1 + x'_2)x'_0$$

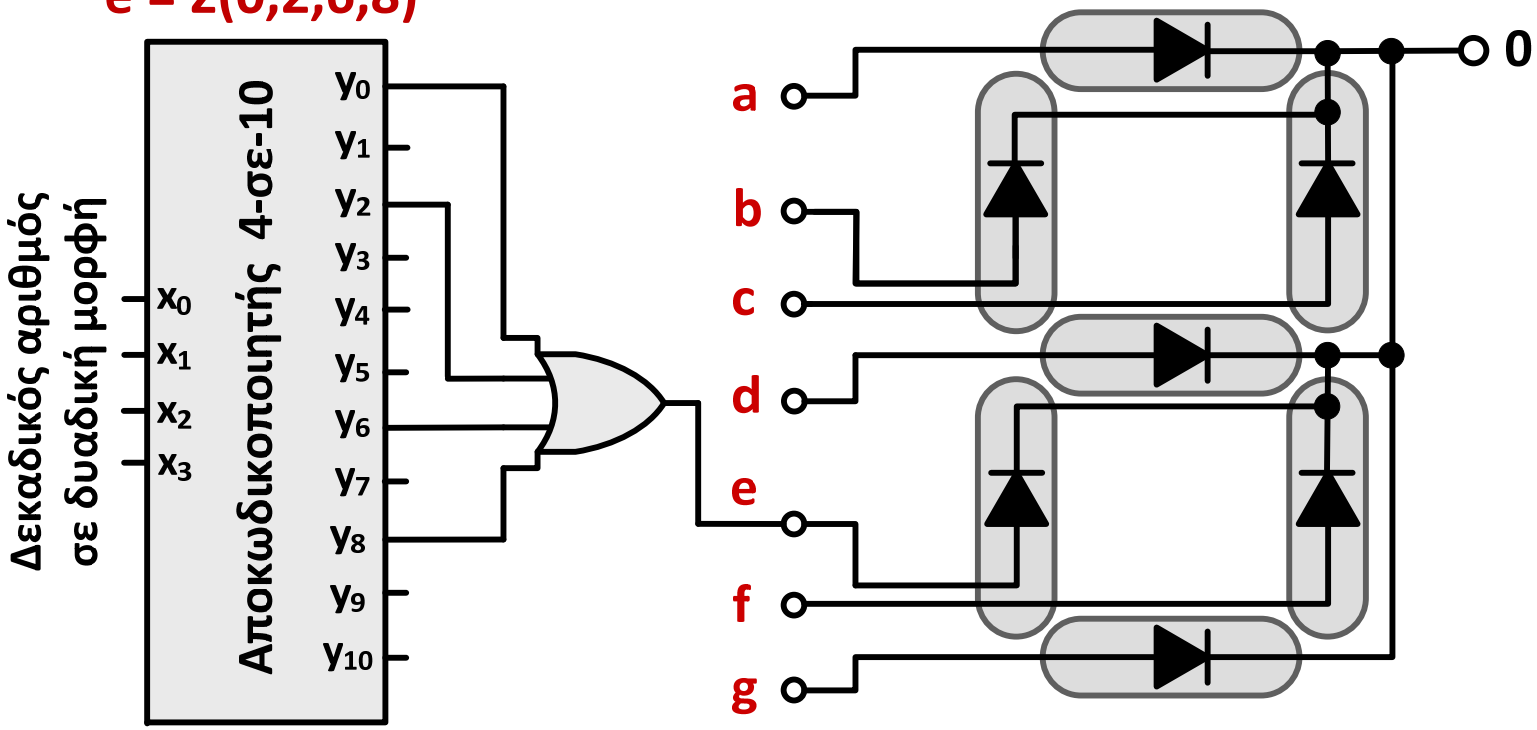



Με όμοιο τρόπο υλοποιούνται οι συναρτήσεις των υπόλοιπων εξόδων.

# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Υλοποίηση της συνάρτησης της εξόδου e με την δεύτερη επιλογή.

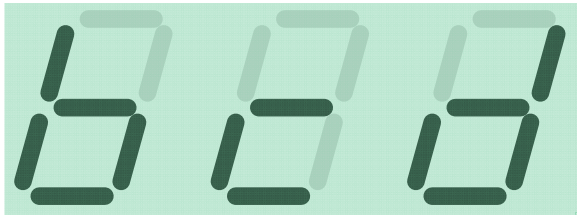
**e = Σ(0,2,6,8)**




Με όμοιο τρόπο υλοποιούνται οι συναρτήσεις των υπόλοιπων εξόδων και επειδή στον πίνακα αλήθειας οι μονάδες είναι περισσότερες από τα μηδενικά, μπορούμε να χρησιμοποιήσουμε την εναλλακτική δομή του ενδείκτη, στην οποία οι δίοδοι φωτοβολούν όταν εφαρμόζεται λογική τιμή 0 στην κάθοδό τους.

# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

**Παράδειγμα 2:** Υλοποίηση με τις προαναφερόμενες δύο επιλογές ενός αποκωδικοποιητή που να εμφανίζει σε ενδείκτη 7 τμημάτων τους παρακάτω χαρακτήρες:



Αφού έχουμε 3 χαρακτήρες, απαιτούνται 2 μεταβλητές εισόδου, από τις οποίες προκύπτουν 4 συνδυασμοί.

Έτσι, ο ζητούμενος αποκωδικοποιητής διαθέτει 2 εισόδους και 7 εξόδους (a, b, c, d, e, f, g).

Για την υλοποίηση του αποκωδικοποιητή, θα χρησιμοποιήσουμε τους 3 συνδυασμούς (έναν για κάθε χαρακτήρα) και οι τιμές των εξόδων που αντιστοιχούν στον τέταρτο συνδυασμό είναι αδιάφορες.

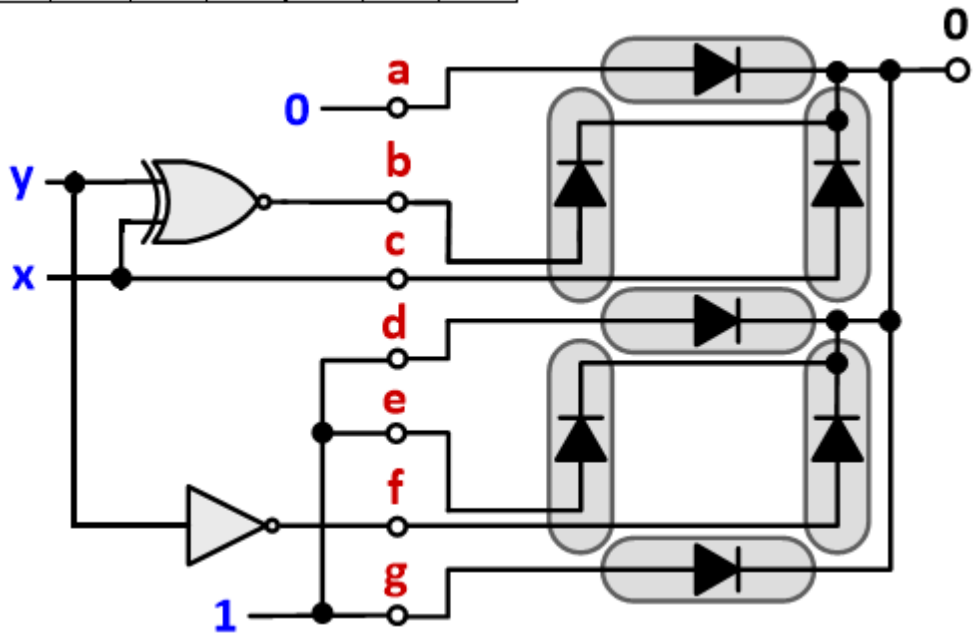
# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Υλοποίηση των συναρτήσεων των 7 εξόδων με την πρώτη επιλογή.

| x | y | | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | b | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | c | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | d | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | | × | × | × | × | × | × | × |

$a = 0,\ b = (x \oplus y)',\ c = x,$

$d = e = g = 1,\ f = y'$

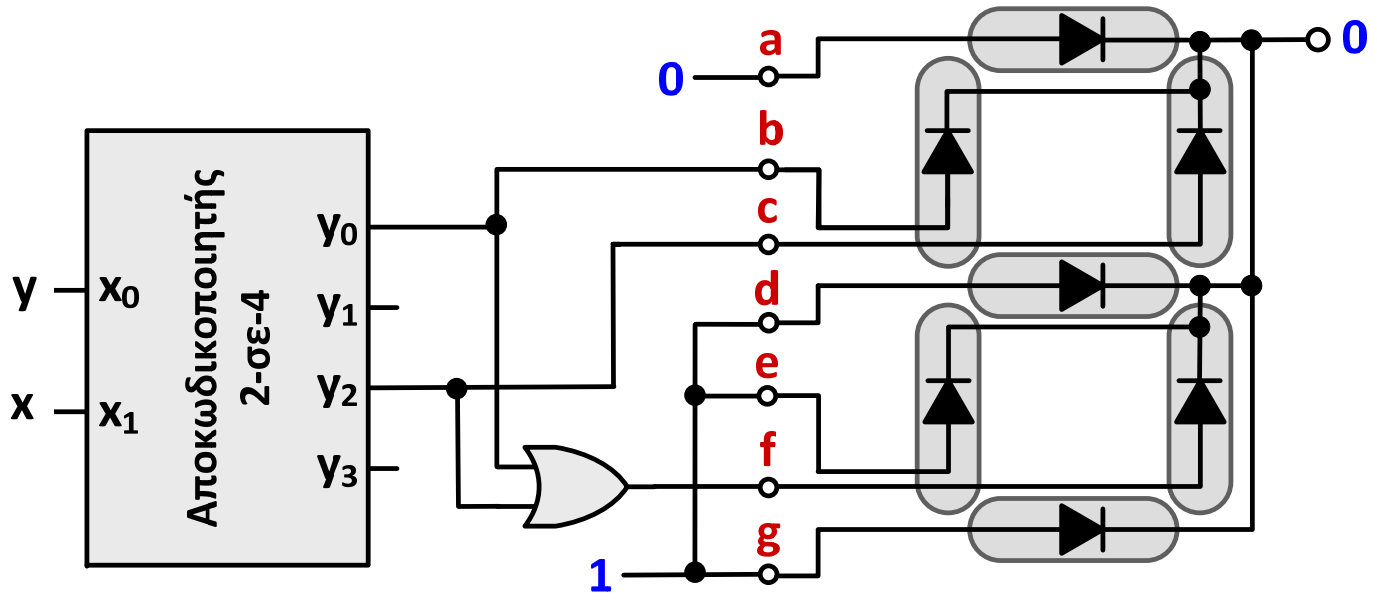


# Αποκωδικοποιητές χαρακτήρων σε ενδείκτη 7 τμημάτων

Υλοποίηση των συναρτήσεων των 7 εξόδων με τη δεύτερη επιλογή.

Για την υλοποίηση του ζητούμενου αποκωδικοποιητή θα χρησιμοποιήσουμε έναν τυποποιημένο αποκωδικοποιητή 2-σε-4, λαμβάνοντας υπόψη ότι οι συναρτήσεις των εξόδων όπως προκύπτουν από τον πίνακα αλήθειας είναι οι εξής:

$$a = 0,\ b = m_0,\ c = m_2,\ d = e = g = 1,\ f = m_0 + m_2$$



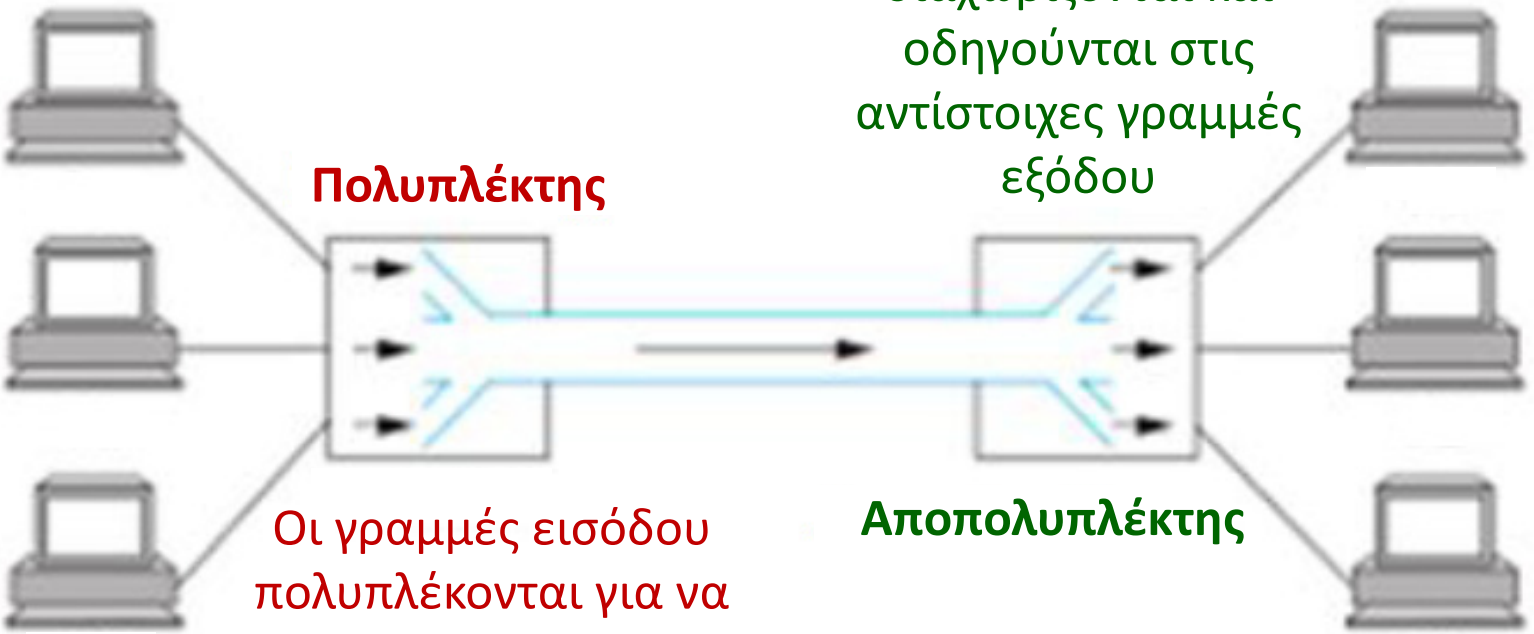
# Πολυπλεξία και αποπολυπλεξία

Στις τηλεπικοινωνίες και τα δίκτυα υπολογιστών πολυπλεξία (multiplexing) είναι η τεχνική που επιτρέπει σε ψηφιακά δεδομένα από διαφορετικές πηγές, να μεταδοθούν μέσα από το ίδιο φυσικό μέσο (π.χ. καλώδιο επικοινωνίας), το οποίο έτσι διαμοιράζεται σε πολλαπλούς χρήστες.

Για παράδειγμα, στην πολυπλεξία διαίρεσης χρόνου (time division multiplexing, TDM), ο χρόνος μετάδοσης χωρίζεται σε χρονοθυρίδες (time slots) και κάθε πακέτο δεδομένων μεταδίδεται στο φυσικό μέσο σε συγκεκριμένη χρονοθυρίδα.

Η αντίστροφη διαδικασία (αποπολυπλεξία, demultiplexing) διενεργείται από κάθε δέκτη, για να απομονωθεί και να ληφθεί, το ζητούμενο πακέτο δεδομένων.

# Πολυπλεξία και αποπολυπλεξία



# Πολυπλέκτες

Οι πολυπλέκτες (multiplexers) ως συνδυαστικά κυκλώματα, λαμβάνουν στις εισόδους τους $2^N$ δυαδικά ψηφία και μεταφέρουν στη μοναδική έξοδό τους την τιμή ενός από τα ψηφία εισόδου.

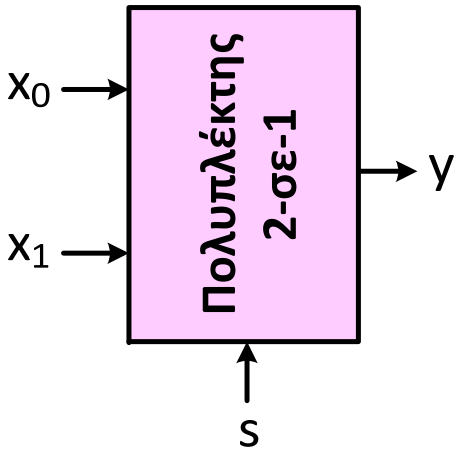
Επομένως, είναι κυκλώματα με $2^N$ εισόδους και μία έξοδο (πολυπλέκτες $2^N$-σε-1).

Για την επιλογή του ψηφίου εισόδου που θα μεταφερθεί στην έξοδό τους, οι πολυπλέκτες διαθέτουν N πρόσθετες εισόδους, οι οποίες αναφέρονται ως είσοδοι επιλογής (select inputs), έτσι ώστε να διακρίνονται από τις $2^N$ εισόδους δεδομένων (data inputs).

# Πολυπλέκτης 2-σε-1

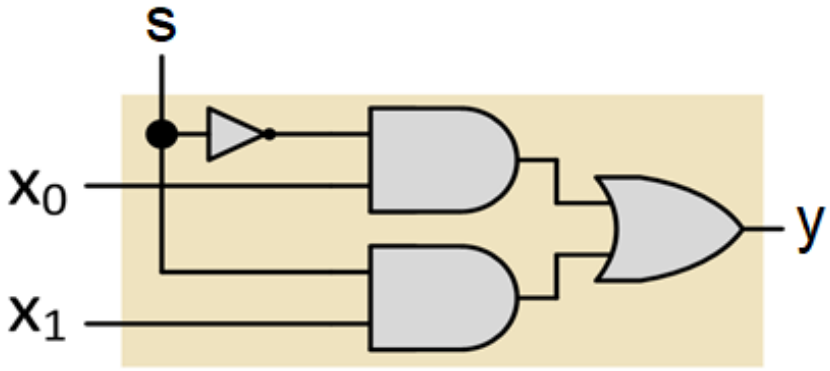
Ο πολυπλέκτης 2-σε-1 είναι ο απλούστερος δυνατός και διαθέτει 2 γραμμές εισόδου δεδομένων ($x_0$, $x_1$), μία έξοδο **y** και μια γραμμή επιλογής **s**.

Όταν **s = 0**, μεταφέρεται στην έξοδο y η είσοδος $x_0$, ενώ όταν **s = 1**, μεταφέρεται στην έξοδο y η είσοδος $x_1$.




| s | y |
|---|---|
| 0 | $x_0$ |
| 1 | $x_1$ |

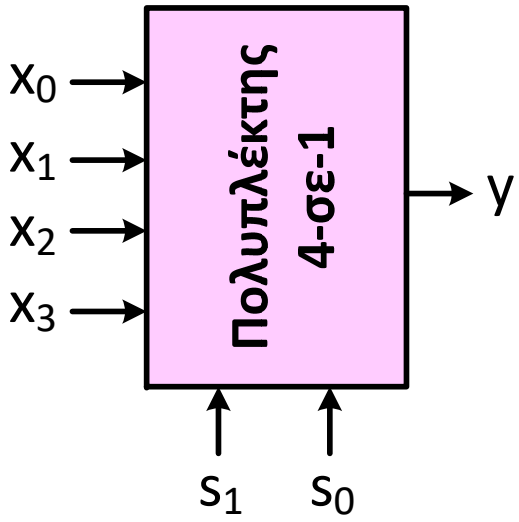
$$y = s'x_0 + sx_1$$



# Πολυπλέκτης 4-σε-1

Ο πολυπλέκτης 4-σε-1 διαθέτει 4 γραμμές εισόδου δεδομένων ($x_0$, $x_1$, $x_2$, $x_3$) μία έξοδο **y** και δύο γραμμές ($s_0$, $s_1$).

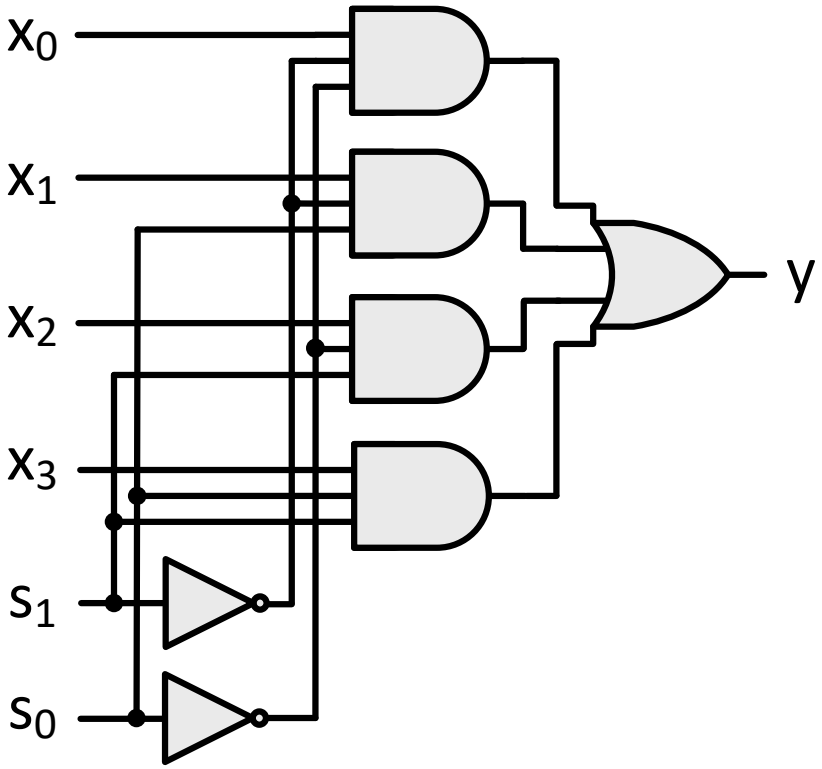
Για καθέναν από τους 4 δυνατούς συνδυασμούς τιμών των 2 εισόδων επιλογής ($s_0$, $s_1$), μεταφέρεται στην έξοδο y η αντίστοιχη γραμμή εισόδου.




| $s_1$ | $s_0$ | y |
|---|---|---|
| 0 | 0 | $x_0$ |
| 0 | 1 | $x_1$ |
| 1 | 0 | $x_2$ |
| 1 | 1 | $x_3$ |

$$y = s'_1 s'_0 x_0 + s'_1 s_0 x_1 + s_1 s'_0 x_2 + s_1 s_0 x_3$$

# Πολυπλέκτης 4-σε-1




Με όμοιο τρόπο μπορούμε να συνθέσουμε πολυπλέκτες με μεγαλύτερο πλήθος εισόδων δεδομένων (1-σε-8, 1-σε-16 κ.ο.κ.), τηρώντας πάντα την αναλογία $2^N$ / N για τις εισόδους δεδομένων και επιλογής, αντίστοιχα.

# Σύνθεση πολυπλεκτών με απλούστερους πολυπλέκτες

Είναι δυνατό να συνθέσουμε πολυπλέκτες πολλών εισόδων με μικρότερους πολυπλέκτες.

Για παράδειγμα, μπορούμε να υλοποιήσουμε έναν πολυπλέκτη 4-σε-1 με πολυπλέκτες 2-σε-1, αρκεί να μετασχηματίσουμε κατάλληλα τη λογική έκφραση της εξόδου ενός πολυπλέκτη 4-σε-1:

$$y = s'_1 s'_0 x_0 + s'_1 s_0 x_1 + s_1 s'_0 x_2 + s_1 s_0 x_3 = s'_1 (s'_0 x_0 + s_0 x_1) + s_1 (s'_0 x_2 + s_0 x_3)$$

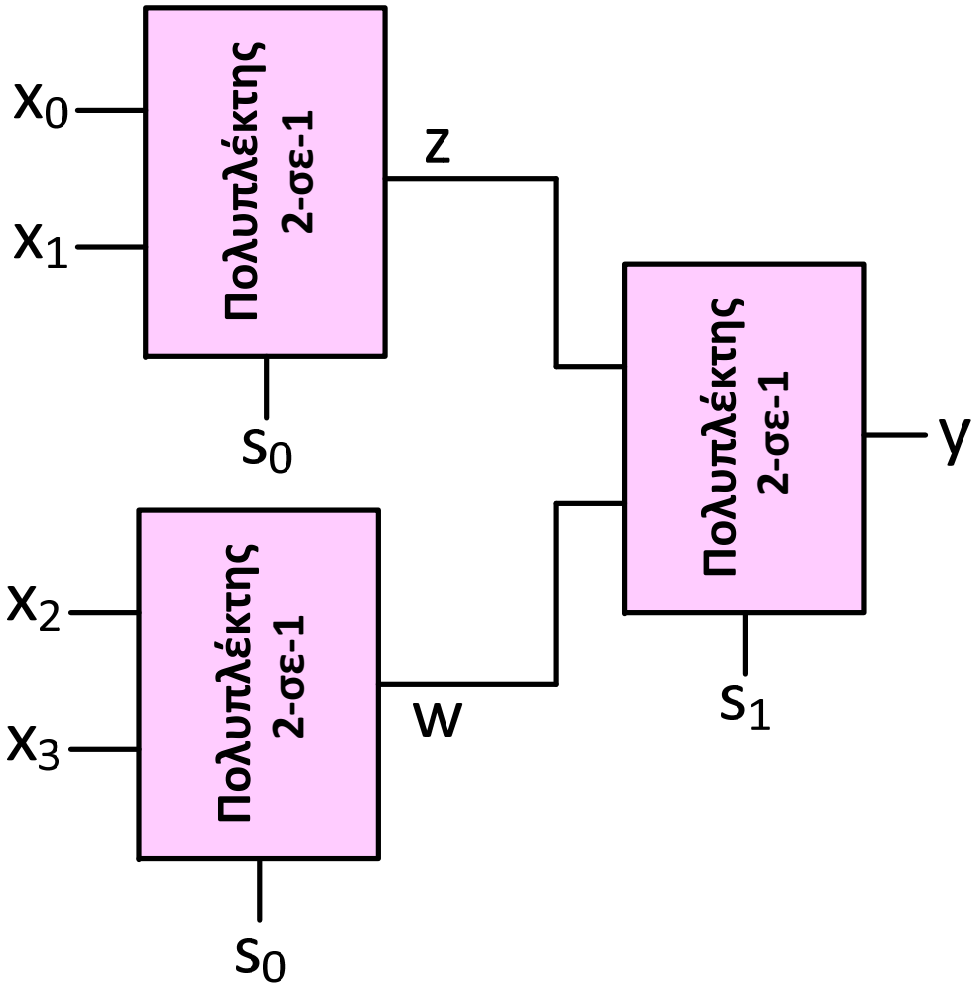
$$= s'_1 z + s_1 w, \text{ όπου: } z = s'_0 x_0 + s_0 x_1 \text{ και } w = s'_0 x_2 + s_0 x_3$$

Παρατηρούμε ότι **z** είναι η έξοδος ενός πολυπλέκτη 2-σε-1 με εισόδους δεδομένων $x_0$ και $x_1$ και είσοδο επιλογής $s_0$.

Ομοίως, **w** είναι η έξοδος ενός πολυπλέκτη 2-σε-1 με εισόδους δεδομένων $x_2$ και $x_3$ και είσοδο επιλογής $s_0$.

Τέλος, **y** είναι η έξοδος ενός πολυπλέκτη 2-σε-1 με εισόδους δεδομένων **z** και **w** (που είναι έξοδοι των 2 προηγούμενων) και είσοδο επιλογής $s_1$.

# Σύνθεση πολυπλεκτών με απλούστερους πολυπλέκτες



# Υλοποίηση λογικών συναρτήσεων με πολυπλέκτη

Από τη σχέση που διέπει τη λειτουργία ενός πολυπλέκτη $2^N$-σε-1, προκύπτει ότι μπορούμε να παράγουμε στην έξοδό του, οποιοδήποτε άθροισμα ελαχιστόρων των N μεταβλητών που αντιστοιχούν στις εισόδους επιλογής, θέτοντας λογική τιμή 1 στις εισόδους δεδομένων που αντιστοιχούν στους ελαχιστόρους που συμμετέχουν στο επιθυμητό άθροισμα και λογική τομή 0 στους ελαχιστόρους που δε συμμετέχουν.

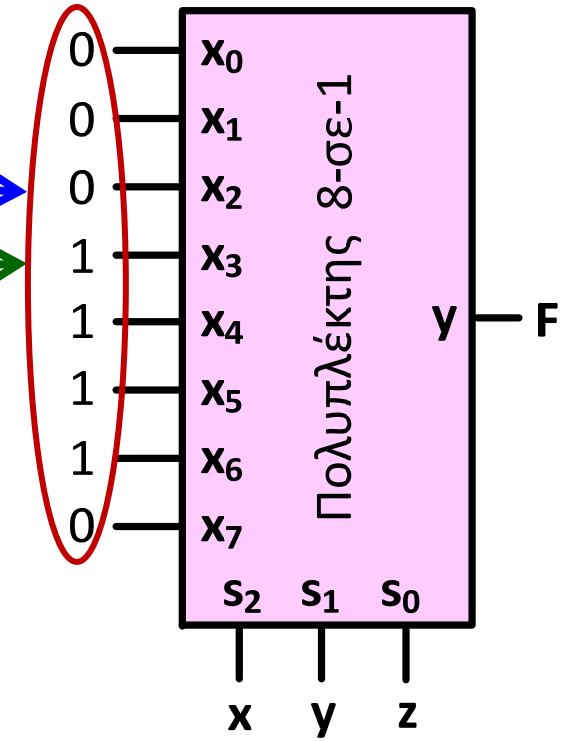
Με τον τρόπο αυτό, χρησιμοποιώντας έναν πολυπλέκτη $2^N$-σε-1, μπορούμε να υλοποιήσουμε οποιαδήποτε λογική συνάρτηση N μεταβλητών, αρκεί να τροφοδοτήσουμε με τις μεταβλητές αυτές τις εισόδους επιλογής του πολυπλέκτη και να θέσουμε τις κατάλληλες λογικές τιμές στις εισόδους δεδομένων του πολυπλέκτη.

# Υλοποίηση λογικών συναρτήσεων με πολυπλέκτη

**Παράδειγμα 1**: Υλοποίηση της λογικής συνάρτησης F(x,y,z) = Σ(3,4,5,6) με πολυπλέκτη 8-σε-1.

F(x,y,z) = Σ(3,4,5,6)

| x | y | z | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |




Πρόκειται για συνάρτηση με N = 3 μεταβλητές, επομένως με βάση τα προαναφερόμενα υλοποιείται με έναν πολυπλέκτη $2^3$-σε 1 (8-σε-1).

# Υλοποίηση λογικών συναρτήσεων με πολυπλέκτη

Ωστόσο, προκύπτει, ότι είναι πιο αποδοτικό να υλοποιήσουμε οποιαδήποτε λογική συνάρτηση N μεταβλητών, με έναν μικρότερο πολυπλέκτη $2^{N-1}$-σε-1, τροφοδοτώντας τις (N – 1) εισόδους επιλογής του πολυπλέκτη με (N – 1) μεταβλητές της συνάρτησης και χρησιμοποιώντας τη μεταβλητή της συνάρτησης που απομένει για την τροφοδότηση εισόδου ή εισόδων δεδομένων του πολυπλέκτη.

Για να επιτευχθεί η υλοποίηση λογικών συναρτήσεων, ενδέχεται να απαιτηθεί η χρήση ενός αντιστροφέα, έτσι ώστε να παράγεται η συμπληρωματική μορφή της μεταβλητής που συμμετέχει στην τροφοδότηση των εισόδων δεδομένων του πολυπλέκτη.

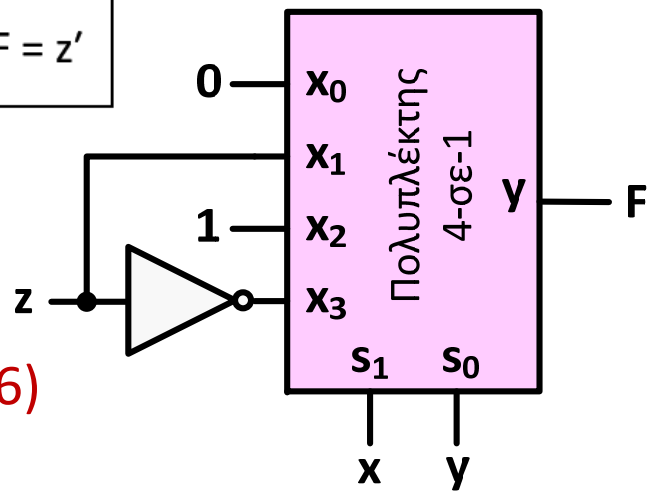
# Υλοποίηση λογικών συναρτήσεων με πολυπλέκτη

**Παράδειγμα 2**: Υλοποίηση της λογικής συνάρτησης F(x,y,z) = Σ(3,4,5,6) με πολυπλέκτη 4-σε-1 και έναν αντιστροφέα (αν απαιτείται).

Τροφοδοτούμε με τις 2 πρώτες μεταβλητές (x, y) τις εισόδους επιλογής του πολυπλέκτη, ενώ τις εισόδους δεδομένων με την κατάλληλη λογική τιμή ή με την κατάλληλη μορφή (κανονική ή συμπληρωματική) της τελευταίας μεταβλητής (z), ανάλογα με την τιμή που λαμβάνει η συνάρτηση F για καθέναν από τους 4 συνδυασμούς τιμών των μεταβλητών x και y.

| x | y | z | F | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | F = 0 |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | F = z |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 1 | F = 1 |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | F = z′ |
| 1 | 1 | 1 | 0 | |

Πίνακας προγραμματισμού του πολυπλέκτη




F(x,y,z) = Σ(3,4,5,6)

# Θεώρημα ανάπτυξης λογικών συναρτήσεων (Shannon)

Κάθε λογική συνάρτηση μπορεί να αναπτυχθεί ως προς μία από τις μεταβλητές που συμμετέχουν σε αυτήν, ως εξής:

$$F(x, y, \ldots, w) = xF(1, y, \ldots, w) + x'F(0, y, \ldots, w)$$

Οι συναρτήσεις $F(1, y, \ldots, w)$ και $F(0, y, \ldots, w)$ προκύπτουν από τη συνάρτηση $F(x, y \ldots, w)$ για $x = 1$ και $x = 0$, αντίστοιχα.

Οι συναρτήσεις αυτές, ως συναρτήσεις πλέον των μεταβλητών y, ..., w, μπορούν με όμοιο τρόπο να αναπτυχθούν ως προς τη μεταβλητή y κ.ο.κ.

Ανάπτυξη συνάρτησης 3 μεταβλητών:

$$F(x, y, z) = xF(1, y, z) + x'F(0, y, z) =$$
$$x[yF(1, 1, z) + y'F(1, 0, z)] + x'[yF(0, 1, z) + y'F(0,0,z)]$$

# Υλοποίηση συναρτήσεων με πολυπλέκτες 2-σε-1

Από την ανάπτυξη μιας λογικής συνάρτησης σύμφωνα με το θεώρημα Shannon, προκύπτει μια μορφή της συνάρτησης που είναι άμεσα υλοποιήσιμη με πολυπλέκτες 2-σε-1:

$$F(x, y, z) = x[yF(1, 1, z) + y'F(1, 0, z)] + x'[yF(0, 1, z) + y'F(0, 0, z)]$$

Οι συναρτήσεις στις αγκύλες υλοποιούνται με έναν πολυπλέκτη 2-σε-1 η καθεμία.

Η είσοδος επιλογής των δύο πολυπλεκτών τροφοδοτείται με τη μεταβλητή y.

Οι είσοδοι δεδομένων των δύο πολυπλεκτών τροφοδοτούνται με τις συναρτήσεις F(1, 1, z), F(1, 0, z), F(0, 1, z), F(0, 0, z), οι οποίες ισούνται με 0 ή 1 ή z ή z'. Οι συναρτήσεις αυτές προκύπτουν εύκολα από τον πίνακα αλήθειας της συνάρτησης F(x, y, z).

Οι έξοδοι των δύο πολυπλεκτών τροφοδοτούν τις εισόδους δεδομένων ενός τρίτου πολυπλέκτη 2-σε-1 με είσοδο επιλογής τη μεταβλητή x.

# Υλοποίηση συναρτήσεων με πολυπλέκτες 2-σε-1

**Παράδειγμα 3**: Υλοποίηση της λογικής συνάρτησης F(x,y,z) = Σ(3,4,5,6) με πολυπλέκτες 2-σε-1 και έναν αντιστροφέα (αν απαιτείται).

$$F(x, y, z) = x[yF(1, 1, z) + y'F(1, 0, z)] + x'[yF(0, 1, z) + y'F(0, 0, z)]$$

| x | y | z | F | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | F = 0 |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | F = z |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 1 | F = 1 |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | F = z′ |
| 1 | 1 | 1 | 0 | |

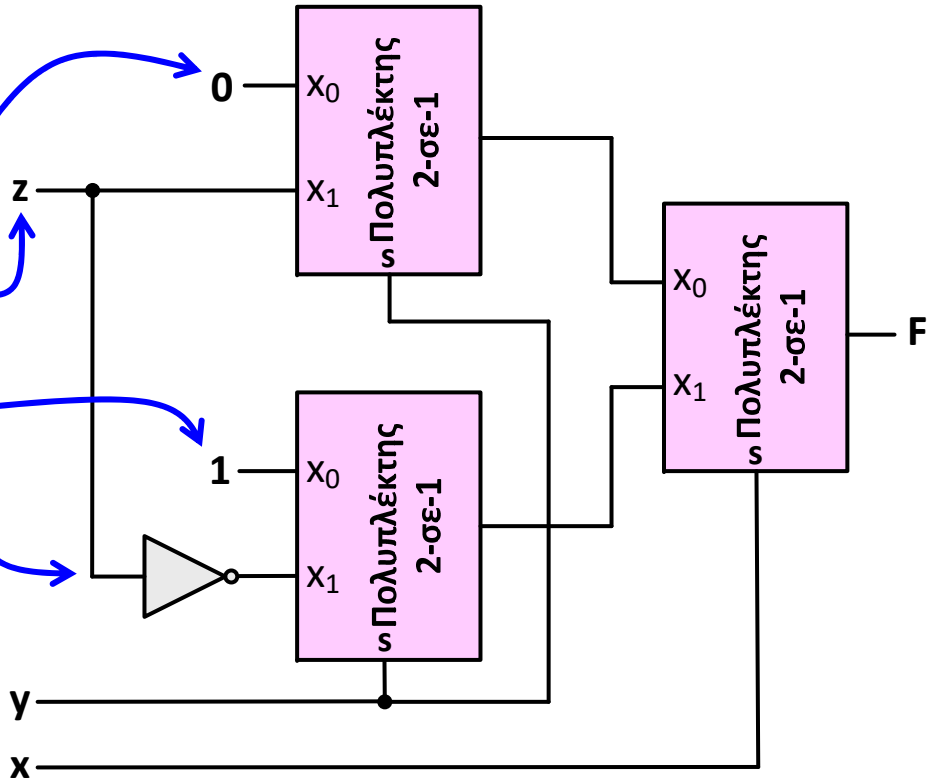
$\Rightarrow F(0, 0, z) = 0$

$\Rightarrow F(0, 1, z) = z$

$\Rightarrow F(1, 0, z) = 1$

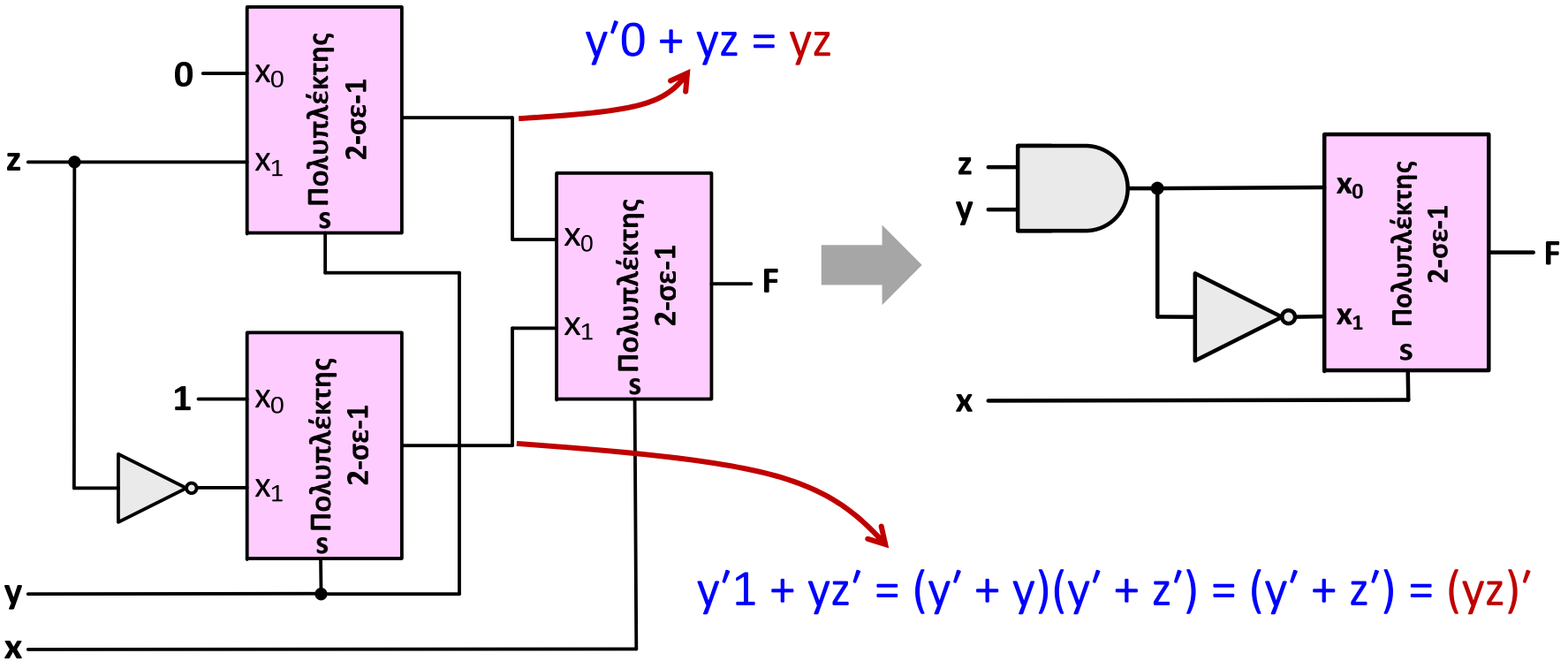
$\Rightarrow F(1, 1, z) = z'$

$$F(x, y, z) = x(yz' + y'1) + x'(yz + y'0)$$



# Υλοποίηση συναρτήσεων με πολυπλέκτες 2-σε-1

Εάν είναι διαθέσιμες πρόσθετες λογικές πύλες, όταν είσοδοι δεδομένων των πολυπλεκτών του πρώτου επιπέδου τροφοδοτούνται με λογικές τιμές 0 ή 1, αντικαθιστώντας πολυπλέκτες με λογικές πύλες, μπορούμε να πετύχουμε απλούστερη υλοποίηση με μικρότερο πλήθος πολυπλεκτών:



# Υλοποίηση συναρτήσεων με πολυπλέκτες 2-σε-1

Με την προαναφερόμενη μεθοδολογία, μπορεί να υλοποιηθεί οποιαδήποτε λογική συνάρτηση N μεταβλητών, χρησιμοποιώντας N – 1 επίπεδα πολυπλεκτών 2-σε-1.

Στο πρώτο επίπεδο συμμετέχουν έως $2^{N-2}$ πολυπλέκτες 2-σε-1, με το πλήθος τους να υποδιπλασιάζεται σε κάθε επόμενο επίπεδο, έως το τελευταίο επίπεδο, το οποίο περιλαμβάνει έναν πολυπλέκτη 2-σε-1.

Στην περίπτωση όπου από την ανάπτυξη μιας λογικής συνάρτησης προκύπτει ότι οι είσοδοι δεδομένων ενός πολυπλέκτη 2-σε-1 τροφοδοτούνται με την ίδια λογική τιμή, μεταβλητή ή συμπληρωματική μορφή μεταβλητής, ο αντίστοιχος πολυπλέκτης απαλείφεται, οδηγώντας σε απλούστερο κύκλωμα υλοποίησης.

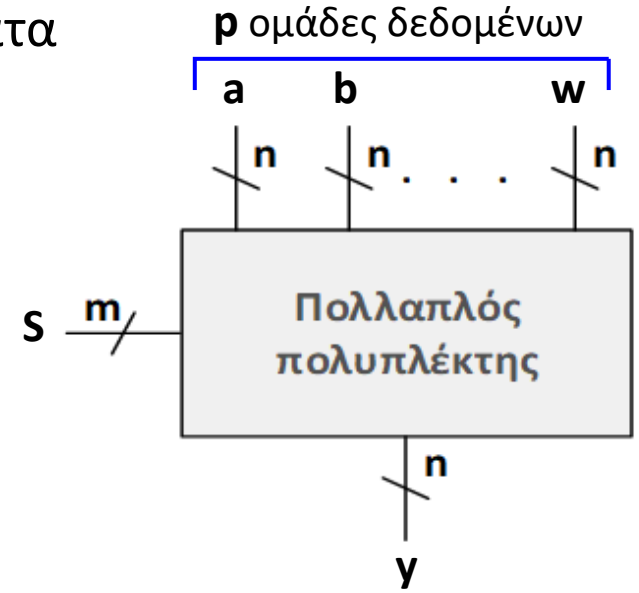
# Πολυπλέκτες επιλογής πολλαπλών εισόδων

Σε αρκετές εφαρμογές, απαιτούνται κυκλώματα για την επιλογή ομάδας δεδομένων.

Στα κυκλώματα αυτά, που αναφέρονται ως πολυπλέκτες επιλογής πολλαπλών εισόδων (multiple-input selection multiplexers) ή απλούστερα ως πολλαπλοί πολυπλέκτες, συνδυάζονται πολυπλέκτες που χρησιμοποιούν κοινές εισόδους επιλογής.

Για τη σύνθεσή τους απαιτείται ο συνδυασμός τόσων πολυπλεκτών όσες είναι οι είσοδοι δεδομένων κάθε ομάδας.

Το πλήθος των εισόδων δεδομένων κάθε πολυπλέκτη ταυτίζεται με το πλήθος των ομάδων δεδομένων.




Πλήθος ψηφίων ομάδας δεδομένων = Πλήθος πολυπλεκτών = **n**

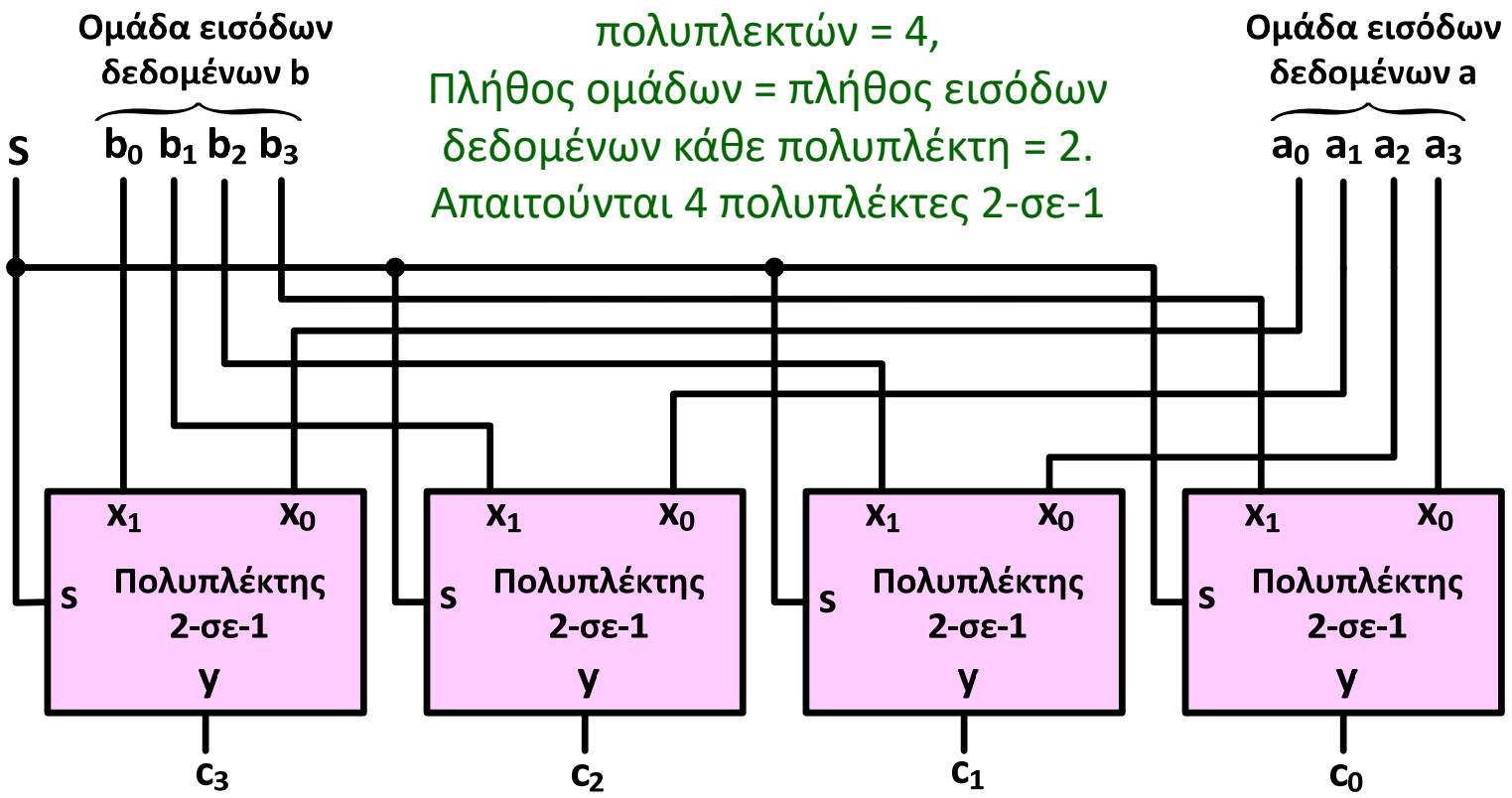
Πλήθος ομάδων = Πλήθος εισόδων δεδομένων κάθε πολυπλέκτη = **p**

Πλήθος εισόδων επιλογής κάθε πολυπλέκτη = **m**, $\mathbf{p \le 2^m}$.

# Πολυπλέκτες επιλογής πολλαπλών εισόδων

**Παράδειγμα:** Σύνθεση ενός τετραπλού πολυπλέκτη 2-σε-1, δηλαδή ενός πολυπλέκτη που να επιλέγει μία από 2 ομάδες ψηφίων που η καθεμία περιλαμβάνει 4 ψηφία και να την μεταφέρει σε 4 εξόδους.

Πλήθος ψηφίων ομάδας = πλήθος πολυπλεκτών = 4,
Πλήθος ομάδων = πλήθος εισόδων δεδομένων κάθε πολυπλέκτη = 2.
Απαιτούνται 4 πολυπλέκτες 2-σε-1



# Αποπολυπλέκτες

Οι αποπολυπλέκτες (demultiplexers) επιτελούν την αντίστροφη λειτουργία από εκείνη των πολυπλεκτών.

Διαθέτουν μία είσοδο δεδομένων και $2^N$ εξόδους δεδομένων, σε μία από τις οποίες μεταφέρεται η είσοδος δεδομένων, ανάλογα με το συνδυασμό τιμών των N εισόδων επιλογής.

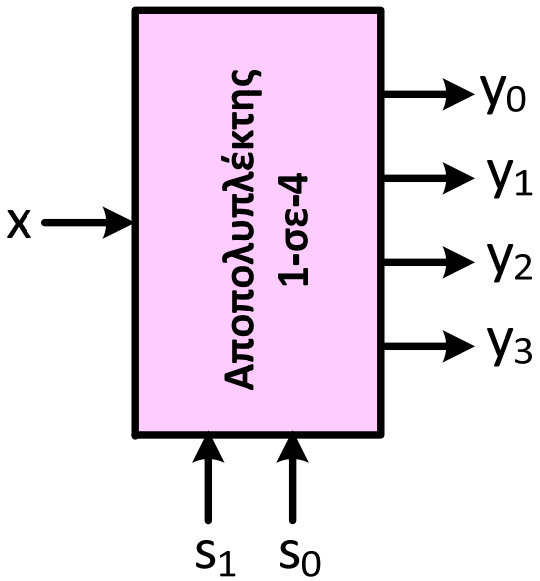
Ο αποκωδικοποιητής με είσοδο ενεργοποίησης που παρουσιάστηκε, γίνεται αποπολυπλέκτης, εάν οι είσοδοι $x_1$, $x_0$ είναι είσοδοι επιλογής και η είσοδος E είναι είσοδος δεδομένων.

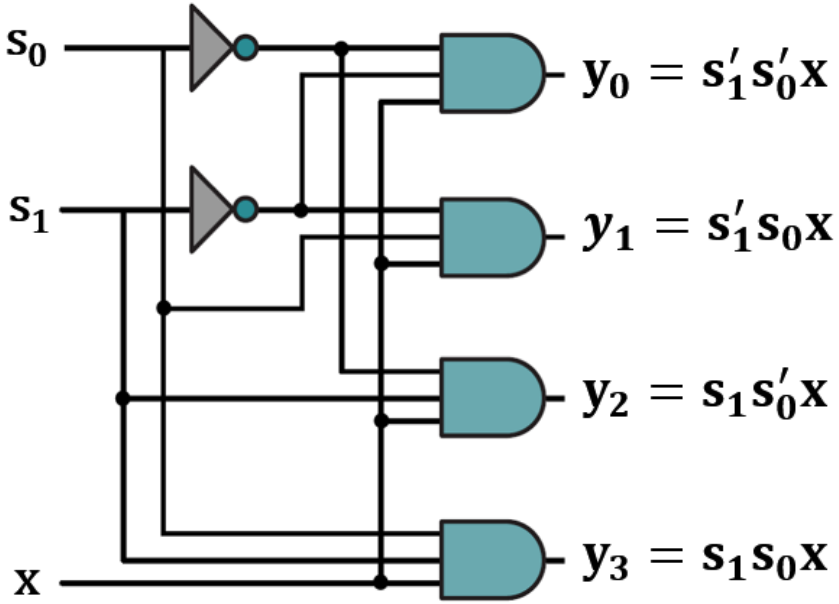
Το πλήθος των εξόδων που μπορούν να ελεγχθούν καθορίζεται από το πλήθος των εισόδων επιλογής.

Έτσι, N είσοδοι επιλογής, μπορούν να ελέγξουν μέχρι $2^N$ εξόδους (αποπολυπλέκτης 1-σε-$2^N$).

# Αποπολυπλέκτες

Ο αποκωδικοποιητής με είσοδο ενεργοποίησης που προαναφέρθηκε, γίνεται αποπολυπλέκτης, εάν οι είσοδοι $x_1$, $x_0$ είναι είσοδοι επιλογής και η είσοδος Ε είναι είσοδος δεδομένων.




Με όμοιο τρόπο μπορούμε να συνθέσουμε αποπολυπλέκτες με μικρότερο ή μεγαλύτερο πλήθος εξόδων (1-σε-2, 1-σε-8, 1-σε-16 κ.ο.κ.), τηρώντας πάντα την αναλογία $2^N$ / N για τις εξόδους και τις εισόδους επιλογής, αντίστοιχα.

# Αποπολυπλέκτες

**Παράδειγμα:**

Σχεδίαση του συνοπτικού διαγράμματος διασύνδεσης ενός πολυπλέκτη 16-σε-1 και ενός αποπολυπλέκτη 1-σε-32 και προσδιορισμός του πλήθους των εισόδων επιλογής του πολυπλέκτη και του αποπολυπλέκτη.

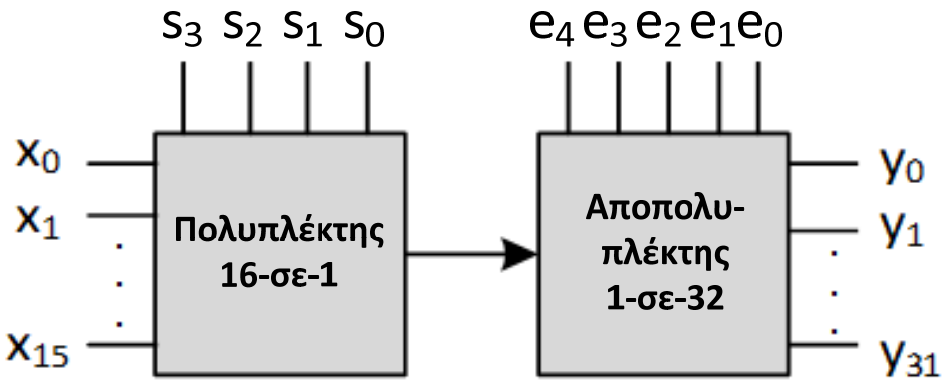
Προσδιορισμός των συνδυασμών τιμών των εισόδων επιλογής του πολυπλέκτη και του αποπολυπλέκτη, εάν είναι επιθυμητή η μεταφορά της τιμής της εισόδου δεδομένων $x_{13}$ του πολυπλέκτη στην έξοδο $y_{27}$ του αποπολυπλέκτη.

# Αποπολυπλέκτες

Για τη διασύνδεση του πολυπλέκτη με τον αποπολυπλέκτη απαιτείται απλώς η σύνδεση της μοναδικής εξόδου του πολυπλέκτη στην μοναδική είσοδο του αποπολυπλέκτη.

Για να επιλεγεί μία από τις 16 εισόδους δεδομένων του πολυπλέκτη απαιτούνται 4 είσοδοι επιλογής, αφού $2^4 = 16$. Ομοίως, για να επιλεγεί μία από τις 32 εξόδους του αποπολυπλέκτη απαιτούνται 5 είσοδοι επιλογής, αφού $2^5 = 32$.

Για να επιλεγεί η είσοδος $x_{13}$ του πολυπλέκτη πρέπει $s_3s_2s_1s_0 = 1101$ (= $13_{10}$), ενώ για να επιλεγεί η έξοδος $y_{27}$ του αποπολυπλέκτη πρέπει $e_4e_3e_2e_1e_0 = 11011$ (= $27_{10}$).



# Βιβλιογραφία


M. Mano, M.D. Ciletti, Ψηφιακή σχεδίαση (6η έκδοση), Εκδόσεις Παπασωτηρίου, 2018.

Μ. Ρουμελιώτης, Σ. Σουραβλάς, Ψηφιακή σχεδίαση: αρχές και εφαρμογές. Εκδόσεις Τζιόλα, 2013.

V.P. Nelson, H. Troy Nagle, J. David Irwin, B.D. Carrol, Ανάλυση και σχεδίαση κυκλωμάτων ψηφιακής λογικής, Εκδόσεις Επίκεντρο, 2007.

J.F. Wakerly, Ψηφιακή σχεδίαση: αρχές και πρακτικές (5η έκδοση), Εκδόσεις Κλειδάριθμος, 2019.

Λ. Μπισδούνης, Ψηφιακά συστήματα, Βασικές εξειδικεύσεις σε αρχιτεκτονική & δίκτυα υπολογιστών, Ελληνικό Ανοικτό Πανεπιστήμιο, 2015 (ανατύπωση 2017).