Ψηφιακή Σχεδίαση

Συνδυαστική λογική

ΕΛΕΥΘΕΡΙΟΣ ΚΟΣΜΑΣ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2019-2020 | ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Περίληψη

στην παρούσα διάλεξη...

- θα αναφέρουμε τις κατηγορίες κυκλωμάτων

- θα μελετήσουμε τα συνδυαστικά κυκλώματα
- θα συζητήσουμε τη διαδικασία ανάλυσης και σχεδίασης συνδυαστικών κυκλωμάτων

- θα μελετήσουμε τον τρόπο σχεδίασης του δυαδικού αθροιστή-αφαιρέτη
- θα συζητήσουμε για του δεκαδικούς αθροιστές, σχεδιάζοντας τον αθροιστή BCD
- θα μελετήσουμε τον τρόπο σχεδίασης του δυαδικού πολλαπλασιαστή και του συγκριτή μεγέθους

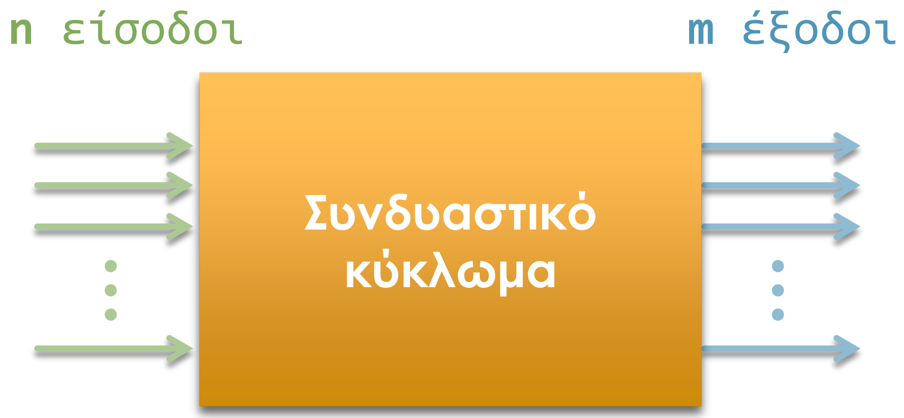
- θα μελετήσουμε τα συνδυαστικά κυκλώματα του αποκωδικοποιητή, του αποπλέκτη, του κωδικοποιητή και του πολυπλέκτη

Εισαγωγή

τα λογικά κυκλώματα που χρησιμοποιούνται στα ψηφιακά συστήματα είναι δύο κατηγοριών:

1. συνδυαστικά
   - συντίθεται από λογικές πύλες
   - ανα πάσα χρονική στιγμή, οι έξοδοι καθορίζονται μόνο από τον τρέχοντα συνδυασμό εισόδων
   - εκτελεί μία λειτουργία που μπορεί να προσδιοριστεί λογικά από ένα σύνολο συναρτήσεων Boole
2. ακουλουθιακά
   - επιπρόσθετα των λογικών πυλών, υπάρχουν και στοιχεία μνήμης
   - οι έξοδοι εξαρτώνται όχι μόνο από τις εισόδους, αλλά και από την κατάσταση των στοιχείων μνήμης
   - η κατάσταση των στοιχείων μνήμης εξαρτάται από προηγούμενες εισόδους

Συνδυαστικά κυκλώματα



- κάθε συνδυαστικό κύκλωμα
  - δέχεται εισόδους
  - έχει δομή στην οποία χρησιμοποιούνται λογικές πύλες
  - παράγει εξόδους

- ένα συνδυαστικό κύκλωμα μπορεί να περιγραφεί
  A. με έναν πίνακα αληθείας
     - για **n** μεταβλητές εισόδου → υπάρχουν 2^n πιθανοί συνδυασμοί των τιμών των εισόδων
  B. από **m** συναρτήσεις Boole, καθεμία εκ των οποίων
     - καθορίζει την τιμή μιας μεταβλητής εξόδου
     - έχει ως ανεξάρτητες μεταβλητές τις **n** μεταβλητές εισόδου

Συνδυαστικά κυκλώματα

Διαδικασία ανάλυσης

Συνδυαστικά κυκλώματα
Διαδικασία ανάλυσης

<u>σκοπός</u>: καθορισμός της λειτουργίας που εκτελεί το (υπό ανάλυση) κύκλωμα

συνήθως,

- ξεκινάμε από το λογικό διάγραμμα του κυκλώματος και
- καταλήγουμε σε
  1. ένα σύνολο από συναρτήσεις Boole, ή
  2. σε έναν πίνακα αληθείας, ή
  3. σε μία λεκτική περιγραφή της λειτουργίας του κυκλώματος

Συνδυαστικά κυκλώματα

Διαδικασία ανάλυσης - Βήματα

1ο βήμα: βεβαιώνουμε ότι το κύκλωμα είναι συνδυαστικό και όχι ακολουθιακό

- το λογικό διάγραμμα ενός συνδυαστικού κυκλώματος
  - ✓ έχει λογικές πύλες
  - ✗ δεν έχει βρόχους ανάδρασης ή στοιχεία μνήμης
    - ο βρόχος ανάδρασης είναι: μία σύνδεση από την έξοδο μιας πύλης στην είσοδο μιας δεύτερης πύλης, της οποίας η έξοδος οδηγείται σε μία από της εισόδους της πρώτης πύλης
    - η ύπαρξη βρόχου ανάδρασης → καθιστά το κύκλωμα ακολουθιακό

2ο βήμα: προσδιορίζουμε τις συναρτήσεις Boole των εξόδων ή τον πίνακα αληθείας του κυκλώματος

3ο βήμα: Εάν ζητείται η λειτουργία του κυκλώματος → ερμηνεύουμε κατάλληλα είτε τις συναρτήσεις είτε τον πίνακα αληθείας

Συνδυαστικά κυκλώματα

Διαδικασία ανάλυσης - Προσδιορισμός συναρτήσεων Boole

1. δίνουμε ονόματα σε όλες τις εξόδους των πυλών που παριστάνουν συναρτήσεις εισόδων

2. δίνουμε ονόματα στις εξόδους των πυλών που έχουν ως εισόδους:
   - είτε εξόδους των πυλών που ονομάσαμε προηγουμένως
   - είτε μεταβλητές εισόδου,

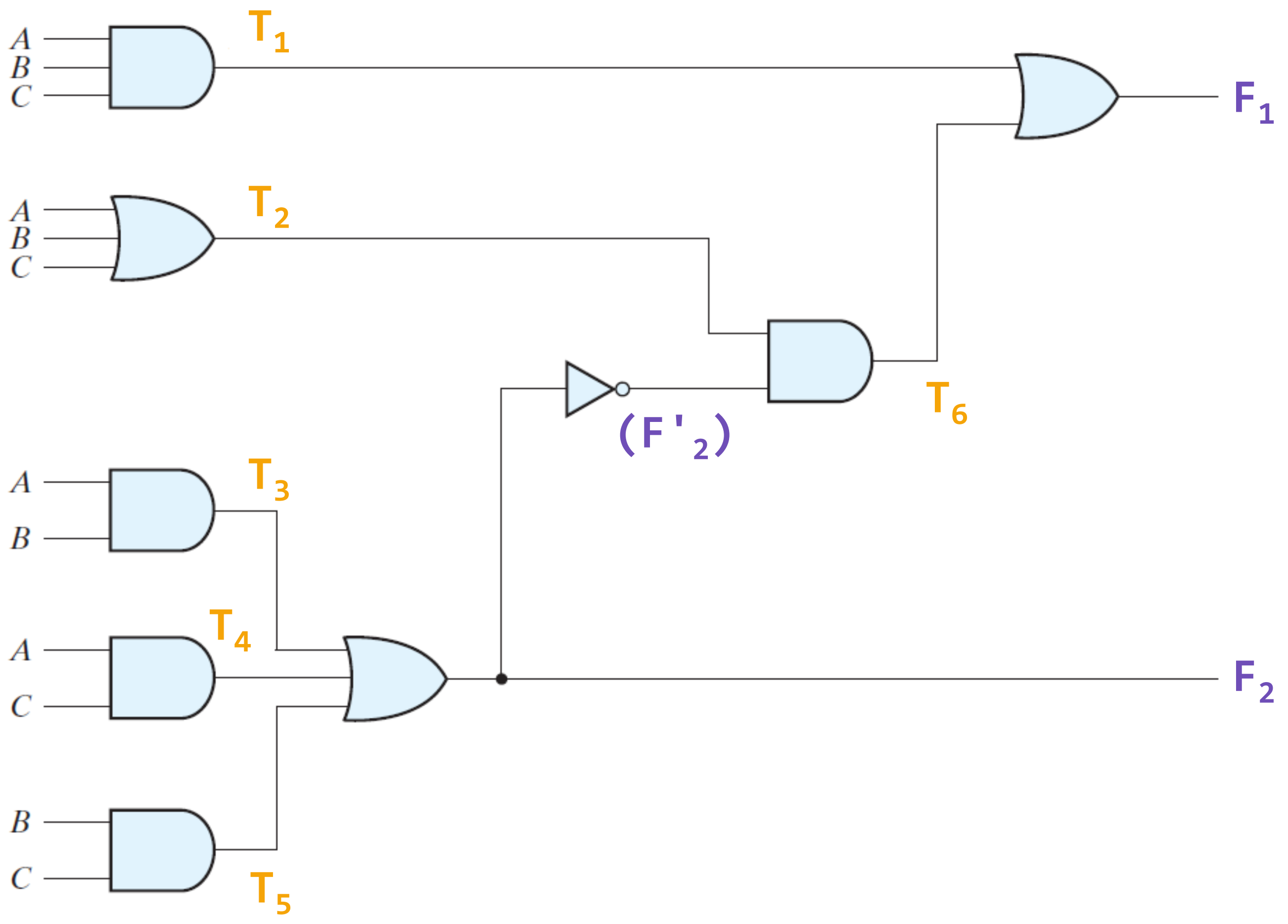
   και προσδιορίζουμε τις αντίστοιχες συναρτήσεις Boole (ως προς τα δοθέντα ονόματα)

3. επαναλαμβάνουμε τη διαδικασία του δεύτερου βήματος, μέχρι να προσδιορίσουμε τις συναρτήσεις Boole όλων των εξόδων του κυκλώματος

4. αντικαθιστούμε, από το τέλος προς την αρχή, τα ονόματα των συναρτήσεων → με τις αλγεβρικές τους μορφές → παίρνουμε τις συναρτήσεις Boole των εξόδων (ως προς τις μεταβλητές εισόδου)

Συνδυαστικά κυκλώματα

Διαδικασία ανάλυσης - Προσδιορισμός συναρτήσεων Boole - Παράδειγμα



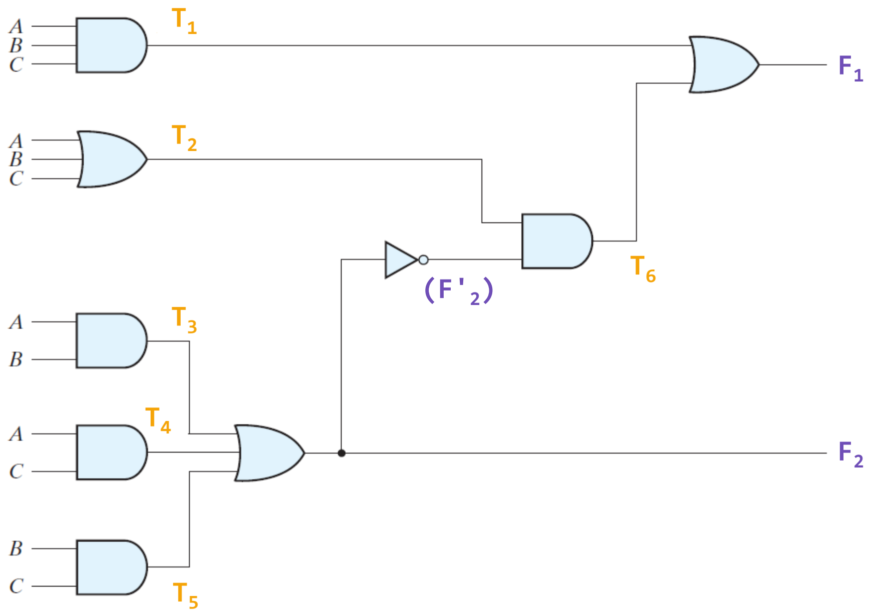
- $T_1 = ABC$
- $T_2 = A + B + C$
- $T_3 = AB$
- $T_4 = AC$
- $T_5 = BC$
- $F_2 = T_3 + T_4 + T_5$
  $= AB + AC + BC$
- $T_6 = T_2F_2'$
  $= (A + B + C)(AB + AC + BC)'$
  $= \ldots = A'BC' + A'B'C + AB'C'$
- $F_1 = T_1 + T_6$
  $= A'BC' + A'B'C + AB'C' + ABC$

Συνδυαστικά κυκλώματα
Διαδικασία ανάλυσης - Προσδιορισμός πίνακα αληθείας

- εάν βρεθούν οι συναρτήσεις Boole των εξόδων του κυκλώματος → ο προσδιορισμός του αντίστοιχου πίνακα αληθείας είναι μία απλή διαδικασία
- για να πάρουμε τον πίνακα αληθείας απευθείας από το λογικό διάγραμμα:
  1. καθορίζουμε των αριθμό των μεταβλητών εισόδου του κυκλώματος
     - για **n** μεταβλητές εισόδου → υπάρχουν **2^n** πιθανοί συνδυασμοί των τιμών των εισόδων
  2. δίνουμε ονόματα στις εξόδους των ενδιάμεσων πυλών
  3. παράγουμε τον πίνακα αληθείας για εκείνες τις εξόδους πυλών, οι οποίες εξαρτώνται μόνο από τις μεταβλητές εισόδου
  4. προχωράμε στην παραγωγή του πίνακα αληθείας για εκείνες τις εξόδους πυλών, οι οποίες εξαρτώνται από ενδιάμεσες μεταβλητές που καθορίστηκαν προηγουμένως και
  5. επαναλαμβάνουμε τη διαδικασία του τέταρτου βήματος, μέχρι να προσδιοριστούν πλήρως οι στήλες τιμών όλων των εξόδων

Συνδυαστικά κυκλώματα

Διαδικασία ανάλυσης - Προσδιορισμός πίνακα αληθείας - Παράδειγμα



A	B	C	T_3	T_4	T_5	F_2	F'_2	T_2	T_6	T_1	F_1
0	0	0	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	1	1	0	1
0	1	0	0	0	0	0	1	1	1	0	1
0	1	1	0	0	1	1	0	1	0	0	0
1	0	0	0	0	0	0	1	1	1	0	1
1	0	1	0	1	0	1	0	1	0	0	0
1	1	0	1	0	0	1	0	1	0	0	0
1	1	1	1	1	1	1	0	1	0	1	1

Συνδυαστικά κυκλώματα

Διαδικασία σχεδίασης

Συνδυαστικά κυκλώματα
Διαδικασία σχεδίασης

- ξεκινά από τις προδιαγραφές ενός προς σχεδίαση κυκλώματος και καταλήγει σε
  A. ένα λογικό διάγραμμα ή
  B. σε ένα σύνολο συναρτήσεων Boole
     - βάσει του οποίου μπορούμε να σχεδιάσουμε το λογικό διάγραμμα

- αποτελείται από τα εξής στάδια:
  1. από τις προδιαγραφές → καθορίζουμε τον απαιτούμενο αριθμό εισόδων και εξόδων
  2. κατασκευάζουμε τον πίνακα αληθείας που περιγράφει τη σχέση εισόδων και εξόδων
  3. για κάθε έξοδο, βρίσκουμε την απλοποιημένη συνάρτηση Boole
     - συναρτήσει των μεταβλητών εισόδου
  4. σχεδιάζουμε το λογικό διάγραμμα και επαληθεύουμε την ορθότητα της σχεδίασης

Συνδυαστικά κυκλώματα

Διαδικασία σχεδίασης - Παράδειγμα - Μετατροπή κωδικοποίησης

- Θέλουμε να μετατρέψουμε ένα δυαδικά κωδικοποιημένο ψηφίο (BCD) → σε ψηφίο κωδικοποιημένο κατά τον κώδικα συν-3
  - 1ο στάδιο:
    - έχουμε **10** δεκαδικά ψηφία (`0..9`)
    - για τη δυαδική αναπαράστασή τους χρησιμοποιούμε **4** δυαδικά ψηφία

    επομένως, τόσο το πλήθος των εισόδων όσο και το πλήθος των εξόδων είναι **4**

  - 2ο στάδιο: κατασκευή πίνακα αληθείας
    - δίνουμε αυθαίρετα ονόματα στις εισόδους και στις εξόδους
    - π.χ. **A**, **B**, **C**, **D** και **w**, **x**, **y**, **z**

Συνδυαστικά κυκλώματα

Διαδικασία σχεδίασης - Παράδειγμα - Μετατροπή κωδικοποίησης (ΙΙ)

2ο στάδιο: κατασκευή πίνακα αληθείας

- ενώ υπάρχουν **16** συνδυασμοί μπιτ για τέσσερις δυαδικές μεταβλητές, χρησιμοποιούνται μόνο οι **10**
  - οι **6** συνδυασμοί που δεν παρουσιάζονται θεωρούνται συνθήκες αδιαφορίας
- οπότε, ισχύει:
  - `w(A,B,C,D) = Σ(5,6,7,8,9)`
  - `x(A,B,C,D) = Σ(1,2,3,4,9)`
  - `y(A,B,C,D) = Σ(0,3,4,7,8)`
  - `z(A,B,C,D) = Σ(0,2,4,6,8)`

  με συνθήκες αδιαφορίας:
  - `d(A,B,C,D) = Σ(10,11,12,13,14,15)`

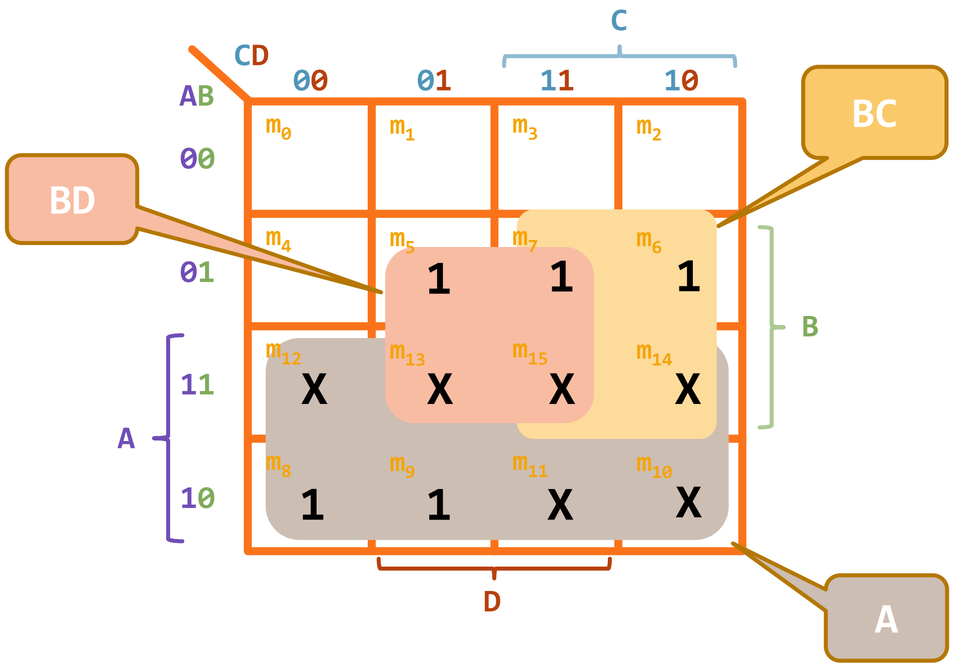
είσοδος σε BCD				έξοδος σε κώδικα συν-3			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Συνδυαστικά κυκλώματα

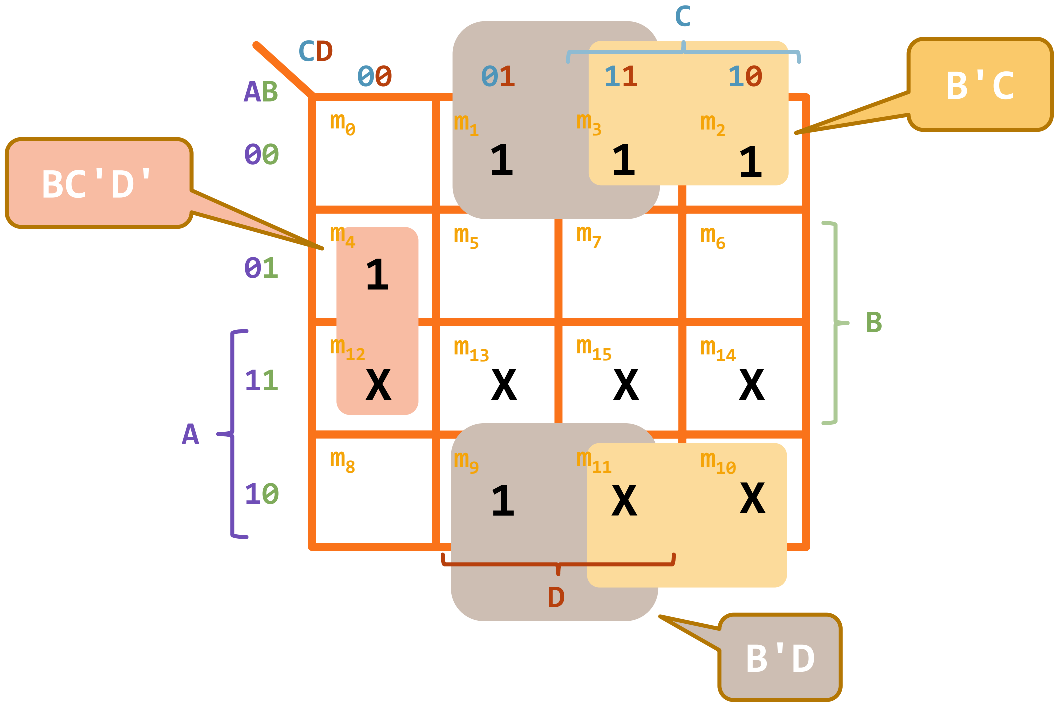
Διαδικασία σχεδίασης - Παράδειγμα - Μετατροπή κωδικοποίησης (ΙΙΙ)

<u>3ο στάδιο</u>: εύρεση απλοποιημένης συνάρτησης Boole για κάθε έξοδο

```
d(A,B,C,D) = Σ(10,11,12,13,14,15)
w(A,B,C,D) = Σ(5,6,7,8,9)
           = A + BC + BD
```




```
d(A,B,C,D) = Σ(10,11,12,13,14,15)
x(A,B,C,D) = Σ(1,2,3,4,9)
           = B'D + B'C + BC'D'
```

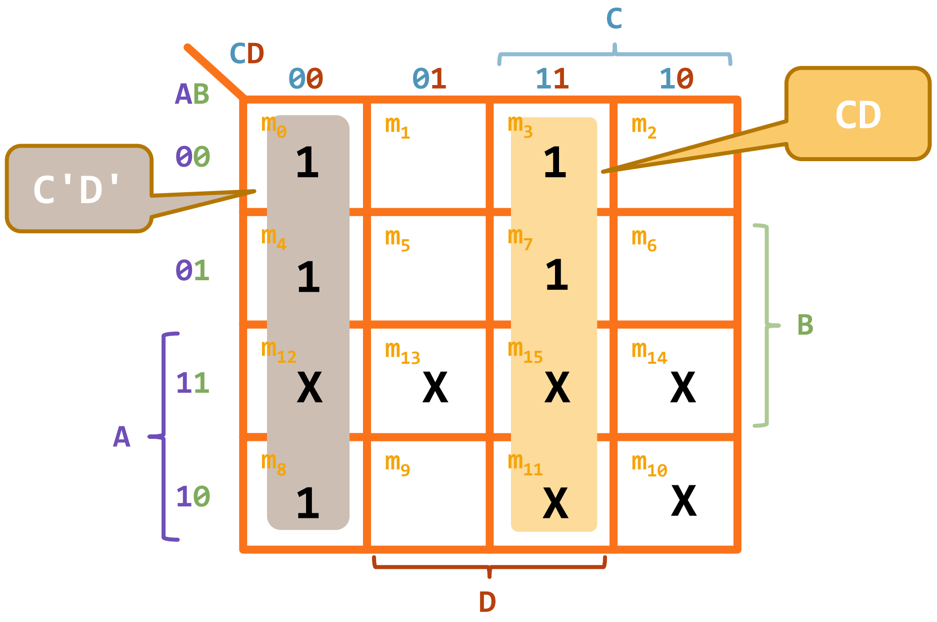


Συνδυαστικά κυκλώματα

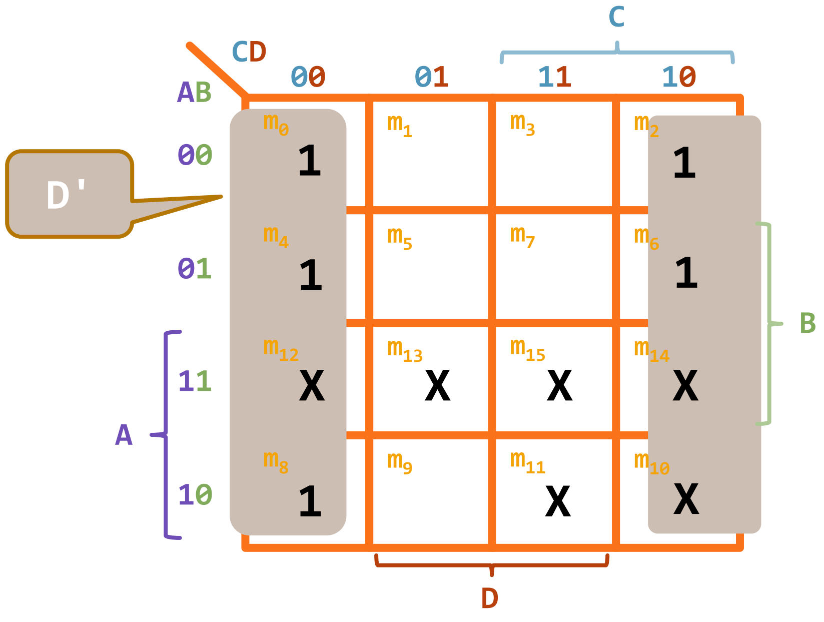
Διαδικασία σχεδίασης - Παράδειγμα - Μετατροπή κωδικοποίησης (IV)

3ο στάδιο: εύρεση απλοποιημένης συνάρτησης Boole για κάθε έξοδο

```
d(A,B,C,D) = Σ(10,11,12,13,14,15)
y(A,B,C,D) = Σ(0,3,4,7,8)
           = C'D' + CD
```




```
d(A,B,C,D) = Σ(10,11,12,13,14,15)
z(A,B,C,D) = Σ(0,2,4,6,8)
           = D'
```

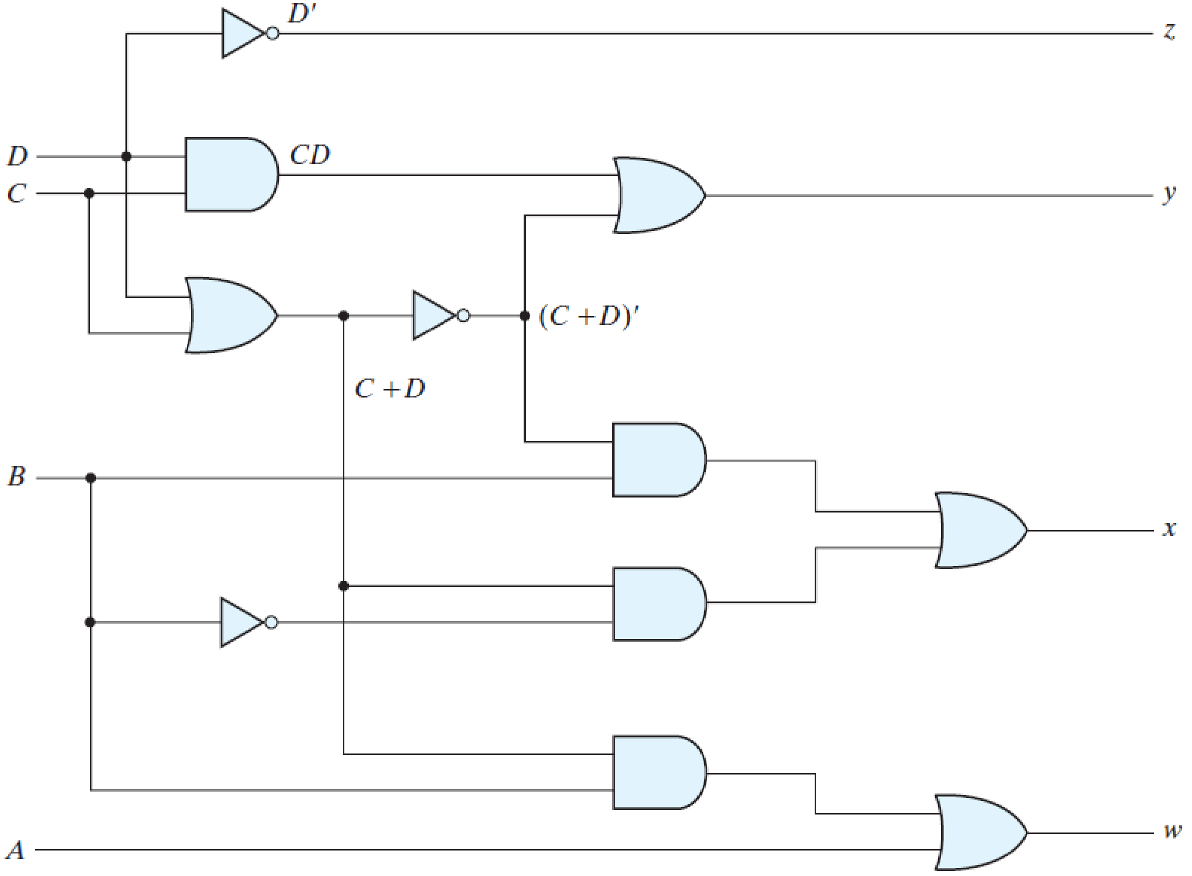


Συνδυαστικά κυκλώματα

Διαδικασία σχεδίασης - Παράδειγμα - Μετατροπή κωδικοποίησης (V)

<u>3ο στάδιο</u>: εύρεση απλοποιημένης συνάρτησης Boole για κάθε έξοδο

- w(A,B,C,D) = A + BC + BD
  = A + B(C + D)
- x(A,B,C,D) = B'D + B'C + BC'D'
  = B'(C + D) + BC'D'
  = B'(C + D) + B(C + D)'
- y(A,B,C,D) = C'D' + CD
  = (C + D)' + CD
- z(A,B,C,D) = D'



<u>4ο στάδιο</u>: σχεδίαση λογικού διαγράμματος

Δυαδικός Αθροιστής-Αφαιρέτης

Εισαγωγή

Πρόσθεση

- η πιο βασική αριθμητική πράξη είναι η πρόσθεση δύο δυαδικών ψηφίων
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1

  προκύπτει ένα ψηφίο

  - 1 + 1 = 10 → **προκύπτουν δύο ψηφία, όπου το πιο σημαντικό να είναι το κρατούμενο**

- όταν οι προσθετέοι έχουν περισσότερα από ένα δυαδικά ψηφία → το κρατούμενο που προκύπτει από τα ψηφία ίδιας τάξης, προστίθεται στα αμέσως πιο σημαντικά ψηφία

Πρόσθεση

Ημιαθροιστής και Πλήρης Αθροιστής

- το συνδυαστικό κύκλωμα που εκτελεί την πρόθεση δύο δυαδικών ψηφίων, ονομάζεται ημιαθροιστής
  - πιθανώς παράγεται κρατούμενο

- το συνδυαστικό κύκλωμα που εκτελεί την πρόσθεση τριών δυαδικών ψηφίων
  - δύο ψηφίων των προσθετέων ίδιας τάξης και
  - του κρατουμένου από το αμέσως προηγούμενο ζεύγος ψηφίων

  ονομάζεται πλήρης αθροιστής

✍ το όνομα του ημιαθροιστή προκύπτει από το γεγονός ότι ένας πλήρης αθροιστής υλοποιείται με το συνδυασμό δύο ημιαθροιστών

Δυαδικός αθροιστής-αφαιρέτης

- ένα συνδυαστικό κύκλωμα που εκτελεί τις αριθμητικές πράξεις:
  1. πρόσθεση
  2. αφαίρεση

  δυαδικών αριθμών

- θα υλοποιήσουμε αυτό το κύκλωμα με χρήση της ακόλουθης ιεραρχικής μεθόδου σχεδίασης:
  a) θα σχεδιάσουμε τον ημιαθροιστή
  b) θα χρησιμοποιήσουμε **2** ημιαθροιστές → για να υλοποιήσουμε τον πλήρη αθροιστή
  c) θα συνδέσουμε **n** πλήρεις αθροιστές σε σειρά → για να υλοποιήσουμε ένα δυαδικό αθροιστή δύο αριθμών των **n** bit
  d) θα χρησιμοποιήσουμε ένα συμπληρωματικό κύκλωμα → για να υλοποιήσουμε την πράξη της αφαίρεσης

Ημιαθροιστής

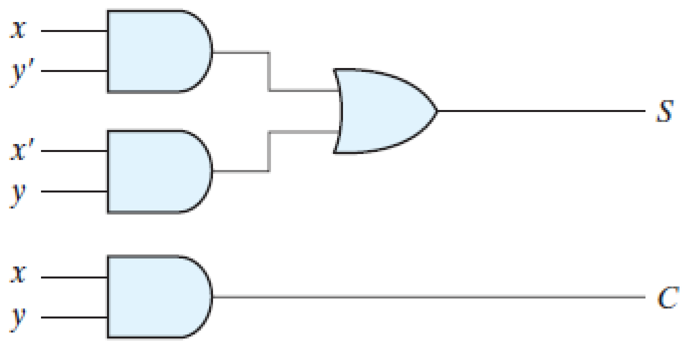
Σχεδίαση

Ημιαθροιστής

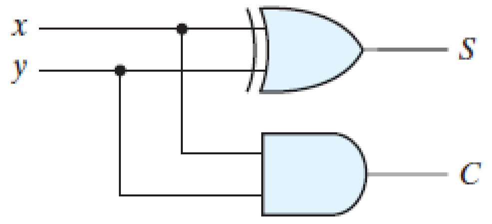
Σχεδίαση συνδυαστικού κυκλώματος

- εκτελεί την πρόθεση δύο δυαδικών ψηφίων, όπου πιθανώς παράγεται κρατούμενο
- από την περιγραφή του ημιαθροιστή διαπιστώνουμε ότι το κύκλωμα πρέπει να έχει:
  - δύο δυαδικές εισόδους **x** και **y** → παριστάνουν τα δυαδικά ψηφία των προσθετέων
  - δύο δυαδικές εξόδους **S** και **C** → για την τιμή αθροίσματος (**S**) και κρατουμένου (**C**)
- απλοποιημένες εκφράσεις:
  - $S = x'y + xy' = x \oplus y$
  - $C = xy$

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



υλοποίηση ημιαθροιστή με πύλες OR, AND και NOT



υλοποίηση ημιαθροιστή με πύλες XOR και AND

Πλήρης αθροιστής

Σχεδίαση

Πλήρης αθροιστής
Σχεδίαση συνδυαστικού κυκλώματος

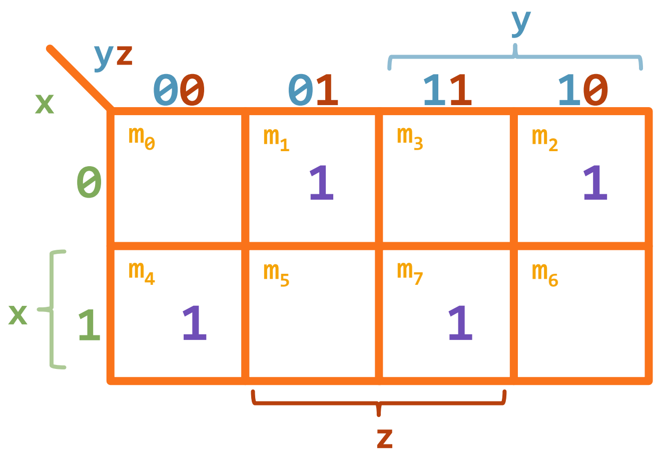
- υπολογίζει το άθροισμα τριών δυαδικών ψηφίων → επομένως, θα πρέπει να έχει:
  - τρεις δυαδικές εισόδους **x**, **y** και **z** → παριστάνουν:
    - τα δύο δυαδικά ψηφία ίδια τάξης (**x**, **y**)των προσθετέων και
    - το κρατούμενο (**z**) από την προηγούμενη, αμέσως λιγότερο σημαντική θέση
  - δύο δυαδικές εξόδους **S** και **C** → για την τιμή αθροίσματος (**S**) και κρατουμένου (**C**)
- κατασκευή πίνακα αληθείας
- οπότε, ισχύει:
  - `S(x,y,z) = Σ(1,2,4,7)`
  - `C(x,y,z) = Σ(3,5,6,7)`

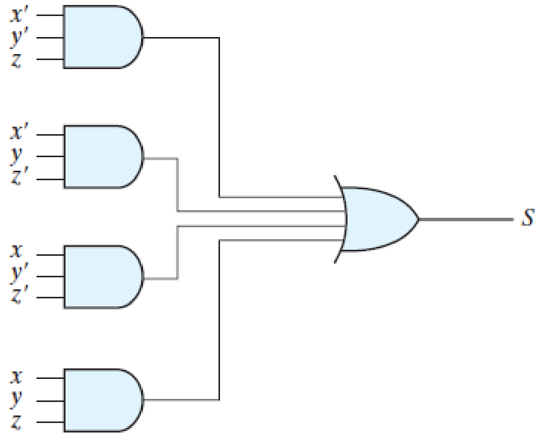
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Πλήρης αθροιστής

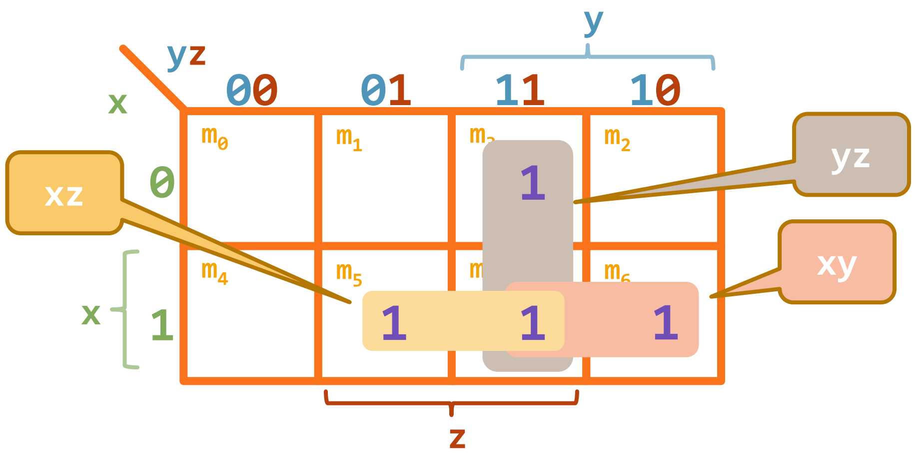
Σχεδίαση συνδυαστικού κυκλώματος - Απλοποίηση & Υλοποίηση

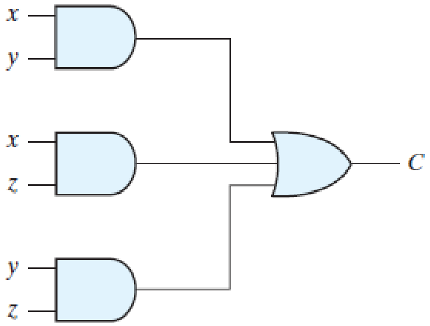
- S(x,y,z) = Σ(1,2,4,7)
  = x'y'z + x'yz' + xy'z' + xyz





- C(x,y,z) = Σ(3,5,6,7)
  = xy + xz + yz





υλοποίηση πλήρη αθροιστή
με πύλες OR, AND και NOT

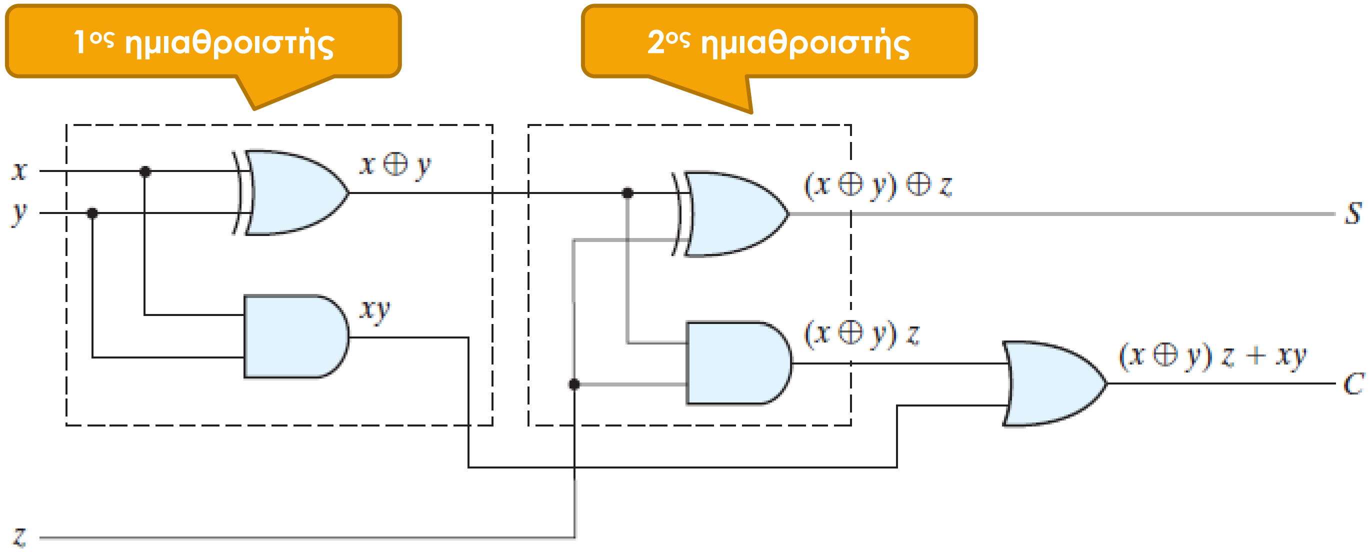
Πλήρης αθροιστής

Σχεδίαση συνδυαστικού κυκλώματος - Υλοποίηση (II)

- $S(x,y,z) = \Sigma(1,2,4,7)$
  $= x'y'z + x'yz' + xy'z' + xyz$
  $= z(x'y' + xy) + z'(xy' + x'y)$
  $= z(xy' + x'y)' + z'(xy' + x'y)$
  $= z(x \oplus y)' + z'(x \oplus y)$
  $= z \oplus (x \oplus y) = (x \oplus y) \oplus z$

- $C(x,y,z) = \Sigma(3,5,6,7)$
  $= xy + xz(y + y') + yz(x + x')$
  $= xy + xyz + xy'z + xyz + x'yz$
  $= xy + xy'z + x'yz$
  $= xy + z(xy' + x'y)$
  $= xy + (x \oplus y)z$

υλοποίηση **πλήρη αθροιστή** με δύο ημιαθροιστές και μία πύλη **OR**



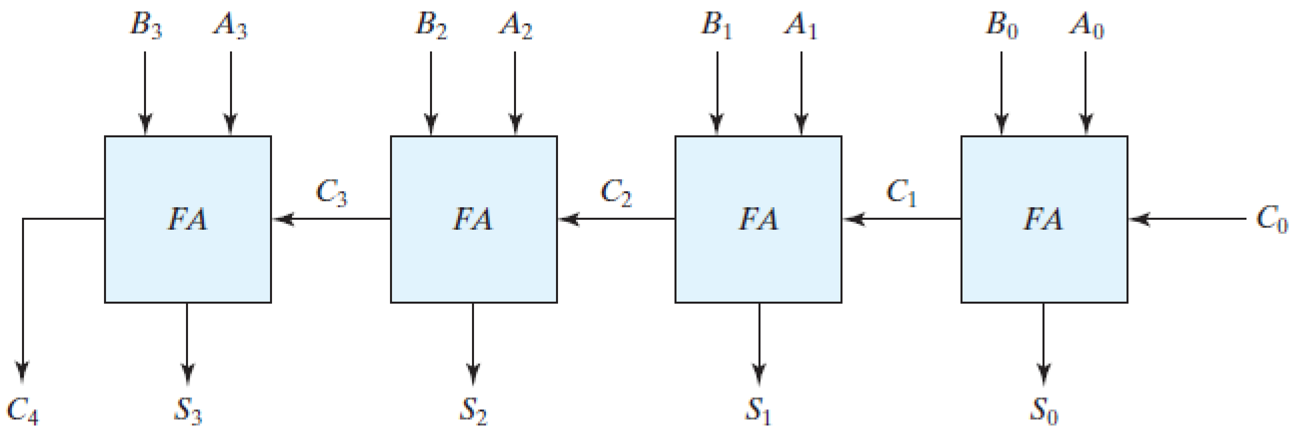
Δυαδικός Αθροιστής

Σχεδίαση

Δυαδικός αθροιστής
Σχεδίαση συνδυαστικού κυκλώματος

- ο δυαδικός αθροιστής είναι ένα ψηφιακό κύκλωμα που παράγει το αριθμητικό άθροισμα δύο δυαδικών αριθμών
- μπορεί να κατασκευαστεί με πλήρεις αθροιστές, ως εξής:
  - συνδέουμε τους πλήρεις αθροιστές (FA) στη σειρά
  - συνδέουμε το κρατούμενο εξόδου κάθε πλήρους αθροιστή με το κρατούμενο εισόδου του επόμενου πλήρους αθροιστή

υλοποίηση δυαδικού αθροιστή δύο τετραψήφιων δυαδικών αριθμών με τέσσερις πλήρεις αθροιστές σε σειρά



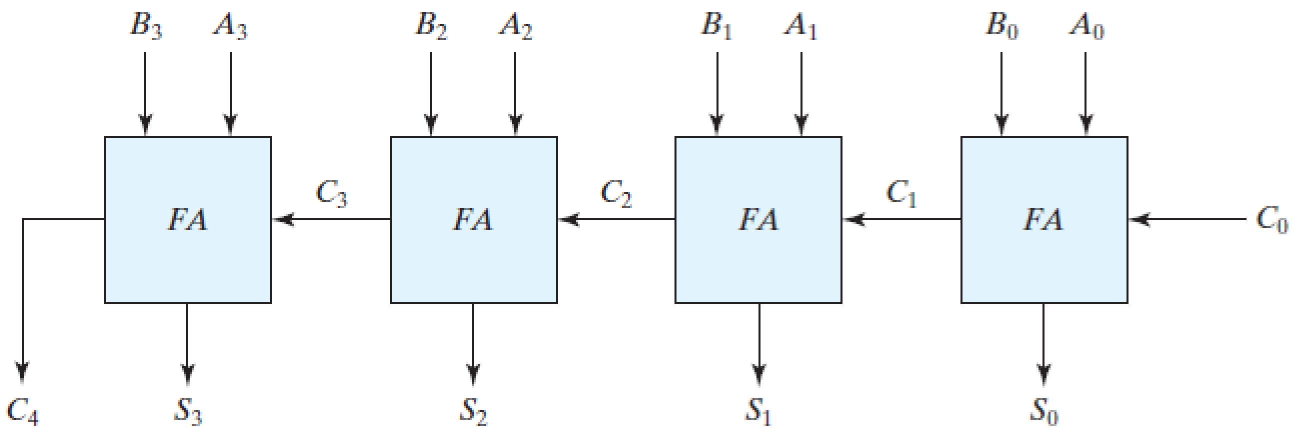
- ο πρώτος δυαδικός αριθμός είναι ο **A** ($A_3A_2A_1A_0$)
- ο δεύτερος δυαδικός αριθμός είναι ο **B** ($B_3B_2B_1B_0$)
- το αρχικό κρατούμενο είναι το C_0
- τα C_1, C_2, C_3 είναι τα ενδιάμεσα κρατούμενα
- το άθροισμα είναι: $S_3S_2S_1S_0$ με τελικό κρατούμενο C_4

Δυαδικός αθροιστής

Σχεδίαση συνδυαστικού κυκλώματος - Αρθροιστής ριπής κρατουμένου

- επειδή η διαδοχική μεταφορά του κρατούμενου μεταξύ των πλήρων αθροιστών θυμίζει ριπή → η συγκεκριμένη υλοποίηση ονομάζεται αθροιστής ριπής κρατούμενου
- για να εμφανιστούν στις εξόδους του κυκλώματος τα σωστά ψηφία → πρέπει πρώτα να παραχθούν όλα τα κρατούμενα

δυαδικός αθροιστής ριπής κρατούμενου δύο τετραψήφιων δυαδικών αριθμών

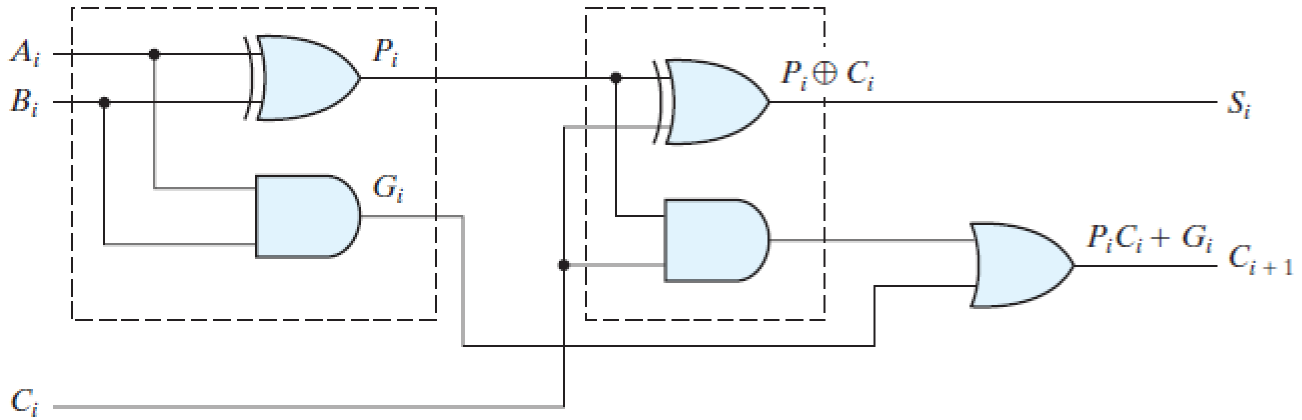


- ✍ με την κλασική μέθοδο σχεδίασης → απαιτείται ένας πίνακας 2^9 γραμμών, καθώς υπάρχουν 9 είσοδοι
- ✍ με την ιεραρχική μέθοδο υλοποίησης → πετύχαμε μια απλή και εύκολα κατανοήσιμη υλοποίηση

Δυαδικός αθροιστής
Διάδοση κρατούμενου

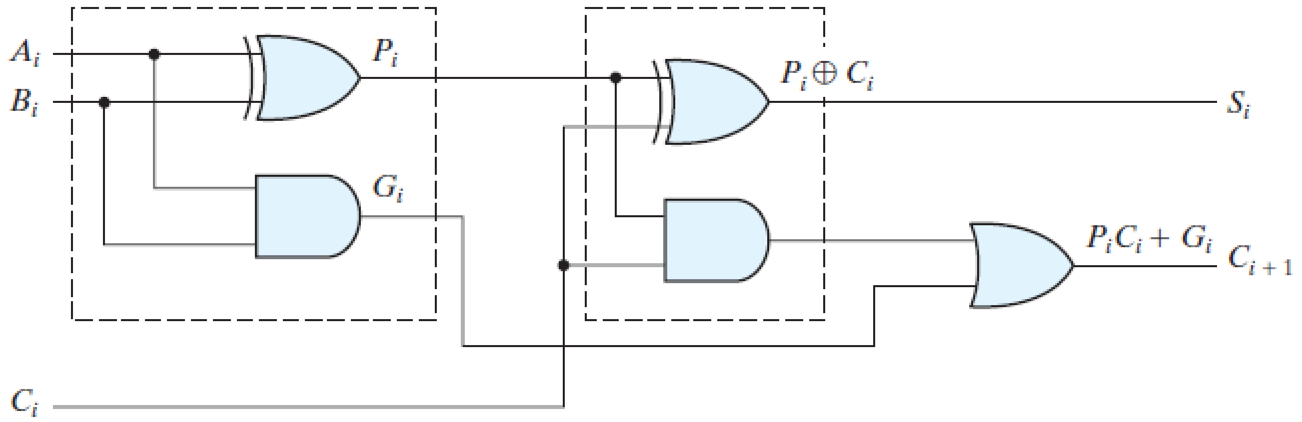
- κατά την πρόσθεση δύο δυαδικών αριθμών, πρέπει όλα τα ψηφία τόσο του πρώτου όσο και του δεύτερου προσθετέου να είναι ταυτόχρονα διαθέσιμα
  - όμως, όπως σε οποιαδήποτε συνδυαστικό κύκλωμα, όλα τα αντίστοιχα σήματα πρέπει να προλάβουν να διαδοθούν μέσω των πυλών
  - η καθυστέρηση διάδοσης εξαρτάται από το πλήθος των πυλών που περνάνε τα σήματα
- επανασχεδιάζουμε τον πλήρη αθροιστή ώστε να φαίνονται τα P_i και G_i
  - τα σήματα στα P_i και G_i καταλήγουν στις τελικές τους σταθερές τιμές μετά τη διάδοση των σημάτων A_i και B_i μέσα από τις αντίστοιχες πύλες, μόνο μία για κάθε σήμα

πλήρης αθροιστής με εμφανή τα P_i και G_i



Δυαδικός αθροιστής
Διάδοση κρατούμενου (ΙΙ)

- για να εμφανιστεί στην έξοδο του κυκλώματος το σωστό ψηφίο (S_i) → πρέπει πρώτα να παραχθεί το κρατούμενο C_i
  - το σήμα κρατουμένου εισόδου C_i διαδίδεται μέσω μίας πύλης **AND** και μίας πύλης **OR** → και καταλήγει ως σήμα κρατουμένου εξόδου C_{i+1}
  - δεδομένου ότι υπάρχουν **n** πλήρεις αρθοιστές σε έναν δυαδικό αθροιστή των **n** bit → υπάρχουν **2n** τέτοια επίπεδα πυλών ανάμεσα στην είσοδο και στην έξοδο
- ο χρόνος διάδοσης των κρατούμενων περιορίζει την ταχύτητα εκτέλεσης της πρόσθεσης
  - πιθανή λύση: χρήση ταχύτερων πυλών
    - **φυσικοί περιορισμοί**
  - λύση: μείωση καθυστέρησης διάδοσης των κρατουμένων
    - αύξηση της πολυπλοκότητας (πλήθος πυλών) του πλήρη αθροιστή



πλήρης αθροιστής με εμφανή τα P_i και G_i

Δυαδικός αθροιστής

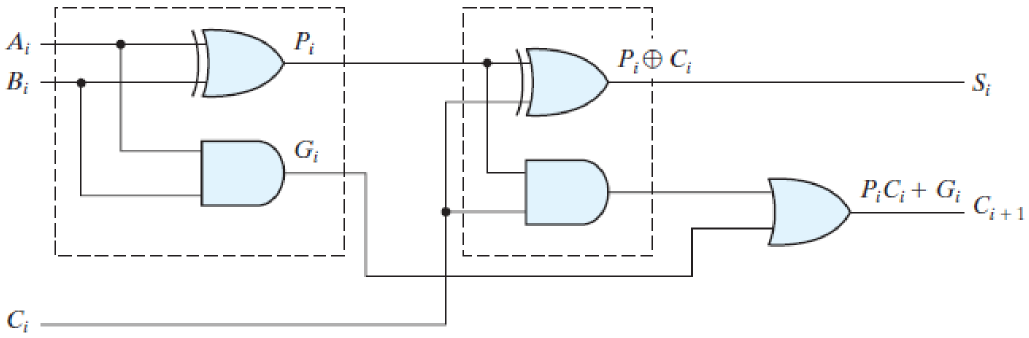
Διάδοση κρατούμενου - Πρόβλεψη κρατουμένου

ισχύουν:

a) $P_i = A_i \oplus B_i$ και $G_i = A_iB_i$

- το G_i ονομάζεται σήμα παραγωγής κρατούμενου, καθώς επιτρέπει την παραγωγή κρατούμενου **1** όταν τα A_i και B_i είναι **1**
  - ανεξάρτητα από την τιμή του κρατούμενου C_i
- το P_i ονομάζεται σήμα διάδοσης κρατούμενου, καθώς προσδιορίζει εάν ένα κρατούμενο στο στάδιο **i** θα περάσει στο στάδιο **i+1**

b) $S_i = P_i \oplus C_i$ και $C_{i+1} = G_i + P_iC_i$



πλήρης αθροιστής με εμφανή τα P_i και G_i

οπότε:

- C_0 = κρατούμενο εισόδου
- $C_1 = G_0 + P_0C_0$
- $C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$
- $C_3 = G_2 + P_2C_2 = G_2 + P_2(G_1 + P_1G_0 + P_1P_0C_0) = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

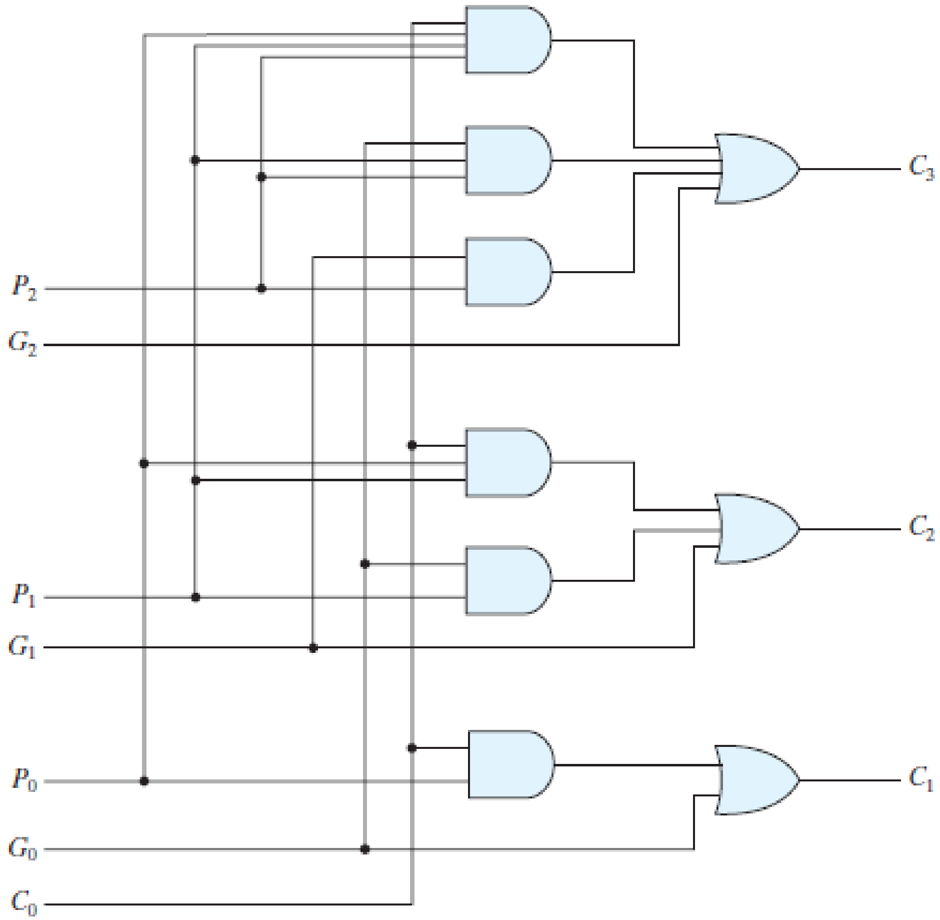
Δυαδικός αθροιστής

Διάδοση κρατούμενου - Πρόβλεψη κρατουμένου (ΙΙ)

- C_0 = κρατούμενο εισόδου
- $C_1 = G_0 + P_0C_0$
- $C_2 = G_1 + P_1G_0 + P_1P_0C_0$
- $C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

- τα κρατούμενα υπολογίζονται ταυτόχρονα

- το όφελος που προκύπτει από την επιτάχυνση της πράξης της πρόσθεσης αντισταθμίζει το κόστος της επιπλέον πολυπλοκότητας του υλικού (αύξηση του αριθμού των πυλών)



λογικό διάγραμμα
γεννήτριας πρόβλεψης κρατούμενων

Δυαδικός αθροιστής (τεσσάρων ψηφίων)

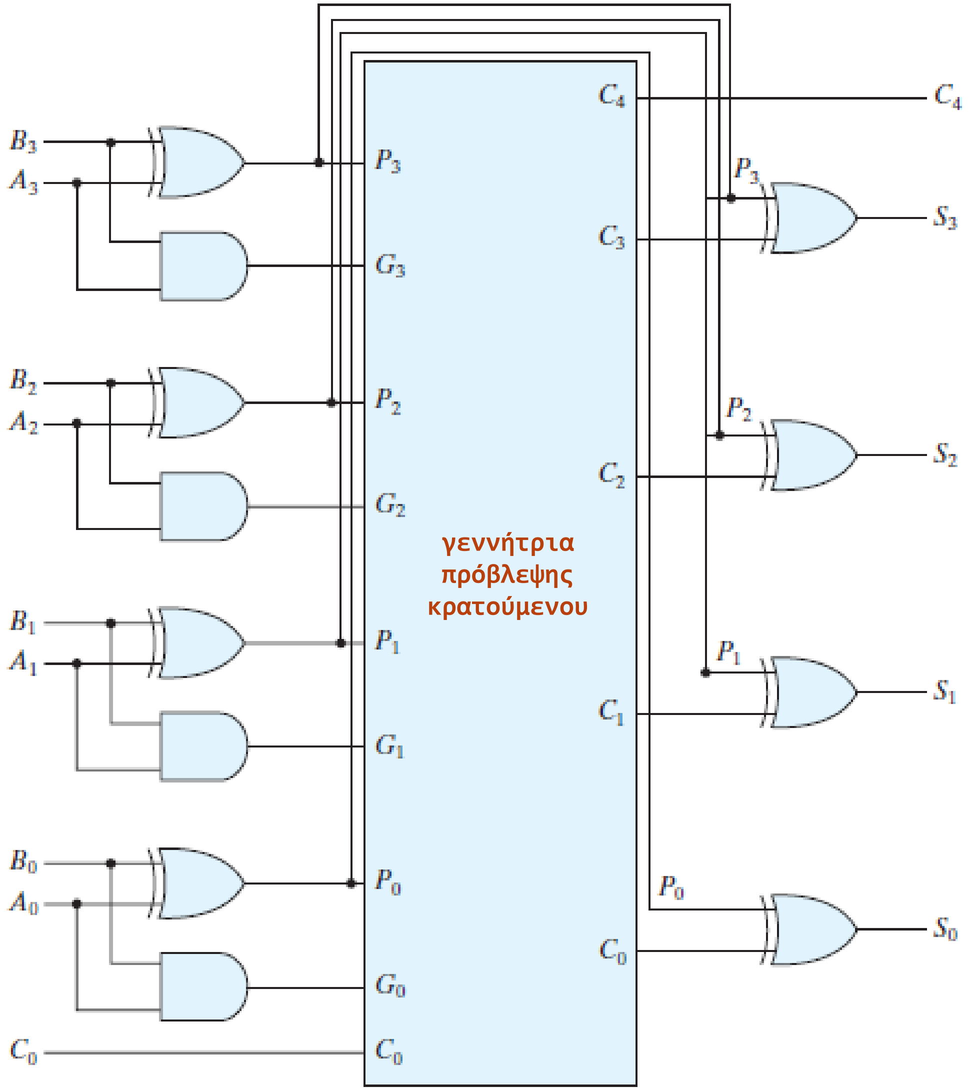
Υλοποίηση με πρόβλεψη κρατουμένου

ισχύουν:

a) $P_i = A_i \oplus B_i$

b) $G_i = A_i B_i$

c) $S_i = P_i \oplus C_i$



λογικό διάγραμμα δυαδικού αθροιστή τεσσάρων ψηφίων που βασίζεται στη μέθοδο της πρόβλεψης κρατούμενων

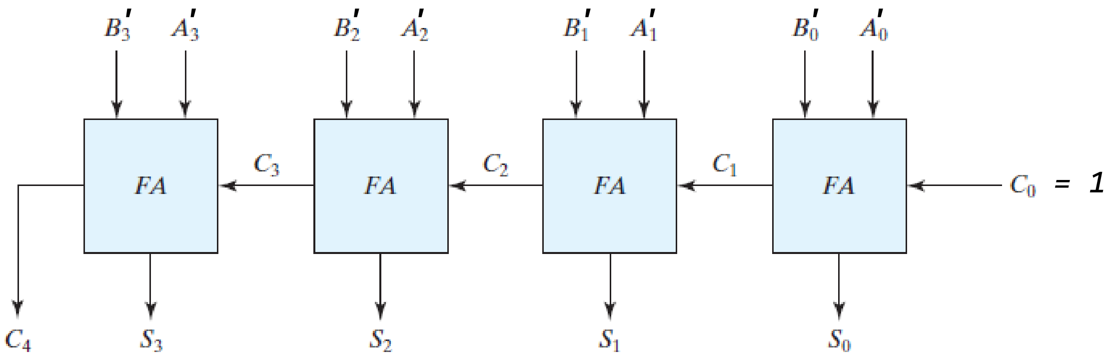
Δυαδικός αθροιστής-αφαιρέτης

Σχεδίαση

Δυαδικός αφαιρέτης
Σχεδίαση

- η αφαίρεση μη προσημασμένων δυαδικών αριθμών γίνεται με πολύ πρακτικό τρόπο, με χρήση των συμπληρωμάτων
  - π.χ. **A-B → A+(το συμπλήρωμα του B ως προς 2)**
  - **(συμπλήρωμα ως προς 2 του B) = (συμπλήρωμα ως προς 1 του B) + 1**
    - το **συμπλήρωμα ως προς 1** μπορεί να υλοποιηθεί με αντιστροφείς
    - το **1** μπορεί να προστεθεί στο συνολικό άθροισμα ως κρατούμενο εισόδου
- ψηφιακό κύκλωμα
  - αποτελείται από έναν δυαδικό αθροιστή
  - χρησιμοποιούνται αντιστροφείς στις εισόδους των ψηφίων των αριθμών

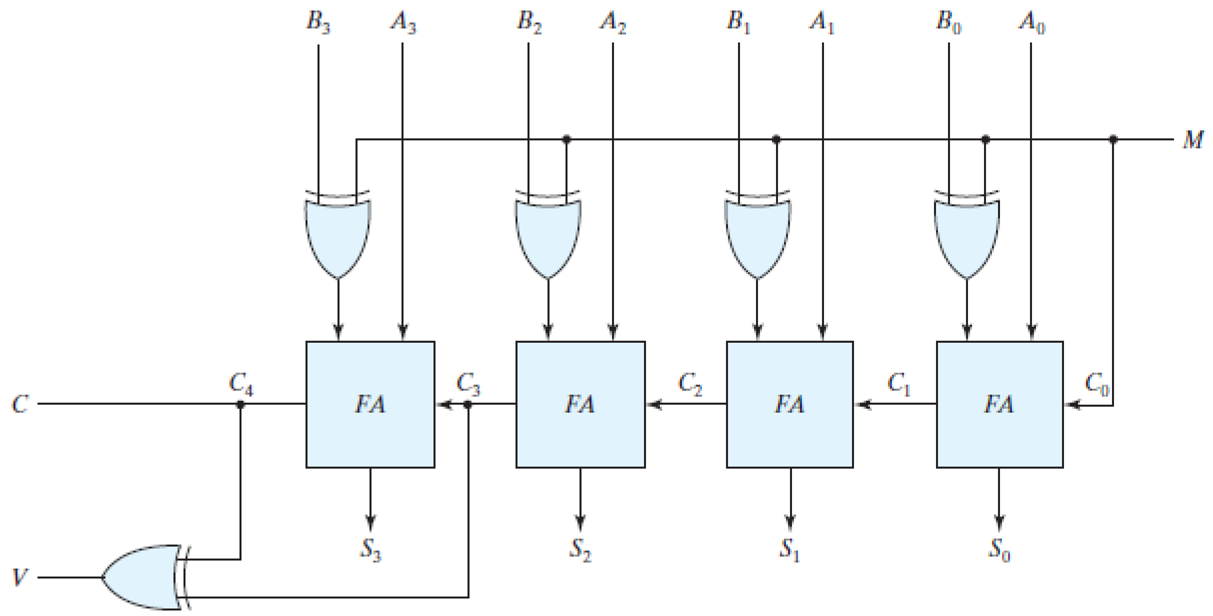
    (χάριν ευκολίας στην παρουσίαση, υποθέτουμε ότι είναι διαθέσιμα τα συμπληρώματα των εισόδων)
  - για μη προσημασμένους αριθμούς υπολογίζει
    - **A-B**, αν **A** ≥ **B** ή
    - το **συμπλήρωμα ως προς 2** του (**B-A**) αν **A** ≤ **B**
  - για προσημασμένους αριθμούς υπολογίζει το **A-B**, εφόσον δεν παρουσιάζεται υπερχείλιση



δυαδικός αφαιρέτης ριπής κρατουμένου δύο τετραψήφιων δυαδικών αριθμών

Δυαδικός αθροιστής-αφαιρέτης
Σχεδίαση

- οι πράξεις της πρόσθεσης και της αφαίρεσης μπορούν να υλοποιηθούν από το ίδιο κύκλωμα, με έναν μόνο κοινό δυαδικό αθροιστή
  - με τη χρήση μίας επιπλέον πύλης **XOR** για κάθε πλήρη αθροιστή
- η είσοδος ελέγχου M καθορίζει την πράξη που θα εκτελεστεί
  a) M = 0 → εκτελείται πρόσθεση
  b) M = 1 → εκτελείται αφαίρεση
- κάθε πύλη **XOR** λαμβάνει ως εισόδους το M και ένα από τα bit του B
  a) όταν M = 0 → B ⊕ 0 = B, (οπότε οι πλήρεις αθροιστές τροφοδοτούνται με τα bit του B) και C_0 = 0 → εκτελείται A+B
  b) όταν M = 1 → B ⊕ 1 = B', (οπότε οι πλήρεις αθροιστές τροφοδοτούνται με τα bit του B συμπληρωμένα) και C_0 = 1 → εκτελείται A+(το **συμπλήρωμα** του B ως προς 2)



δυαδικός αθροιστής-αφαιρέτης ριπής κρατούμενου δύο τετραψήφιων δυαδικών αριθμών

Δυαδικός αθροιστής-αφαιρέτης

Σχεδίαση - Παρατήρηση

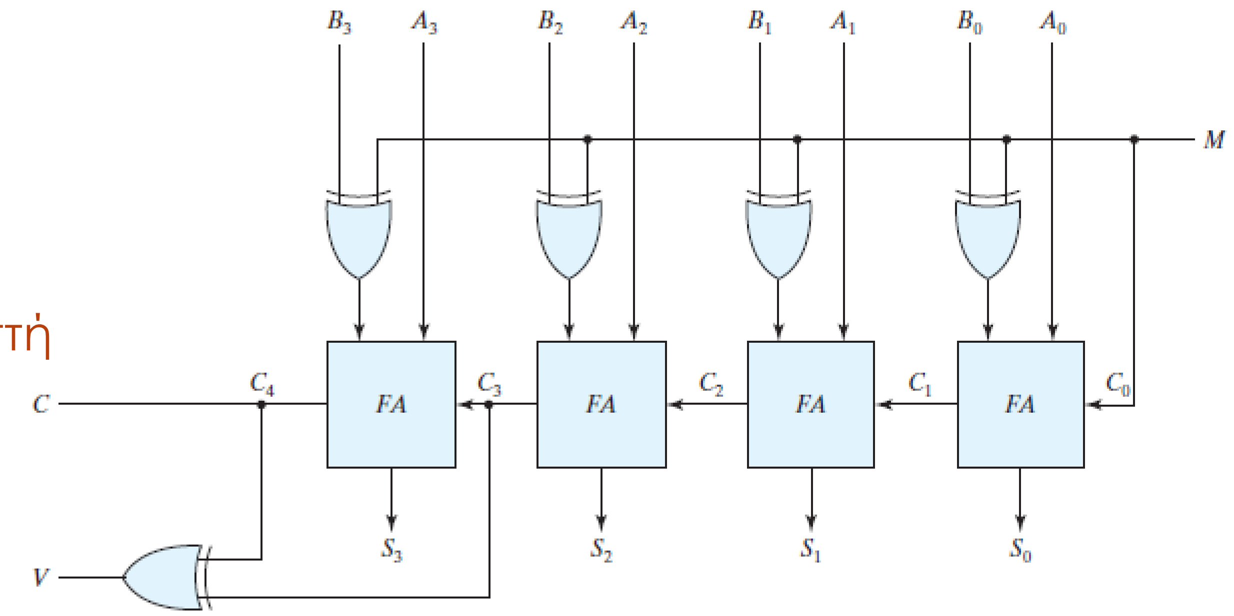
- οι δυαδικοί αριθμοί στο σύστημα προσημασμένου συμπληρώματος
  - προστίθενται και αφαιρούνται
  - σύμφωνα με τους ίδιους βασικούς κανόνες πρόσθεσης και αφαίρεσης

  που ισχύουν για τους μη προσημασμένους δυαδικούς αριθμούς

- οπότε, μπορεί να χρησιμοποιηθεί το ίδιο ψηφιακό κύκλωμα

☞ είναι ευθύνη του χρήστη ή του προγραμματιστή να ερμηνεύσει κατάλληλα τα αποτελέσματα μίας τέτοιας πρόσθεσης ή αφαίρεσης

- ανάλογα με το αν οι αριθμοί είναι προσημασμένοι ή μη προσημασμένοι



δυαδικός αθροιστής-αφαιρέτης ριπής κρατούμενου δύο τετραψήφιων δυαδικών αριθμών

Δυαδικός αθροιστής-αφαιρέτης

Υπερχείλιση (overflow)

- συμβαίνει όταν προστίθενται δύο αριθμοί με **n** ψηφία και το άθροισμά τους έχει **n+1** ψηφία
  - ισχύει για τους δυαδικούς και για τους δεκαδικούς αριθμούς
  - ισχύει για προσημασμένους και μη προσημασμένους αριθμούς
- δε δημιουργεί πρόβλημα όταν η πράξη γίνεται με το μυαλό μας, π.χ. με χαρτί και μολύβι
  - δεν υπάρχει κάποιο πρακτικό όριο στο χώρο που γράφουμε το άθροισμα
- δημιουργεί πρόβλημα στους υπολογιστές
  - ο αριθμός των bit που καταλαμβάνει κάθε αριθμός είναι περιορισμένος
  - ☞ είναι σημαντικό να εντοπιστεί

Δυαδικός αθροιστής-αφαιρέτης
Υπερχείλιση (overflow) - Εντοπισμός

1. πρόσθεση μη προσημασμένων αριθμών
   - η υπερχείλιση εντοπίζεται από το τελικό κρατούμενο της πιο σημαντικής θέσης

✍ προσημασμένοι αριθμοί
   - το πιο σημαντικό bit δηλώνει το πρόσημο
   - οι αρνητικοί αριθμοί είναι σε μορφή **συμπληρώματος ως προς 2**

2. πρόσθεση προσημασμένων αριθμών
   - το bit προσήμου αντιμετωπίζεται ως μέρος του αριθμού
   - η παραγωγή τελικού κρατούμενου δε σημαίνει απαραίτητα ότι συνέβη υπερχείλιση

Δυαδικός αθροιστής-αφαιρέτης

Υπερχείλιση (overflow) - Εντοπισμός - Προσημασμένοι αριθμοί

- η υπερχείλιση (κατά την πρόσθεση) δύο αριθμών μπορεί να συμβεί μόνο αν είναι οι αριθμοί είναι ομόσημοι

π.χ. έστω ότι έχουμε δύο καταχωρητές των 8 bit

τελικό κρατούμενο

πρόσημο

1. πρόσθεση αριθμών +70 και +80
   - $(+70)_{10} = (01000110)_2$
   - $(+80)_{10} = (01010000)_2$

κρατούμενα	0	1	0	0	0	0	0	0	0
		0	1	0	0	0	1	1	0
+		0	1	0	1	0	0	0	0
άθροισμα		1	0	0	1	0	1	1	0

2. πρόσθεση αριθμών -70 και -80
   - $(-70)_{10} = (10111010)_2$
   - $(-80)_{10} = (10110000)_2$

κρατούμενα	1	0	1	1	0	0	0	0	0
		1	0	1	1	1	0	1	0
+		1	0	1	1	0	0	0	0
άθροισμα		0	1	1	0	1	0	1	0

Δυαδικός αθροιστής-αφαιρέτης

Υπερχείλιση (overflow) - Εντοπισμός - Προσημασμένοι αριθμοί (II)

✍ <u>παρατηρήστε ότι</u>: εάν το κρατούμενο που παράγει ο αρθοιστής, μετά το bit προσήμου, θεωρηθεί ως το bit προσήμου → το αποτέλεσμα είναι σωστό

- ωστόσο, δε χωρά σε 8 bit καταχωρητές → υπερχείλιση

μπορούμε να εντοπίσουμε την υπερχείλιση εως εξής:

- παρατηρούμε
  - το κρατούμενο στη θέση bit προσήμου
  - το τελικό κρατούμενο

και εάν τα δύο αυτά κρατούμενα είναι διαφορετικά → έχει συμβεί υπερχείλιση

τελικό κρατούμενο

πρόσημο

κρατούμενα	0	1	0	0	0	0	0	0	0
		0	1	0	0	0	1	1	0
+		0	1	0	1	0	0	0	0
άθροισμα		1	0	0	1	0	1	1	0

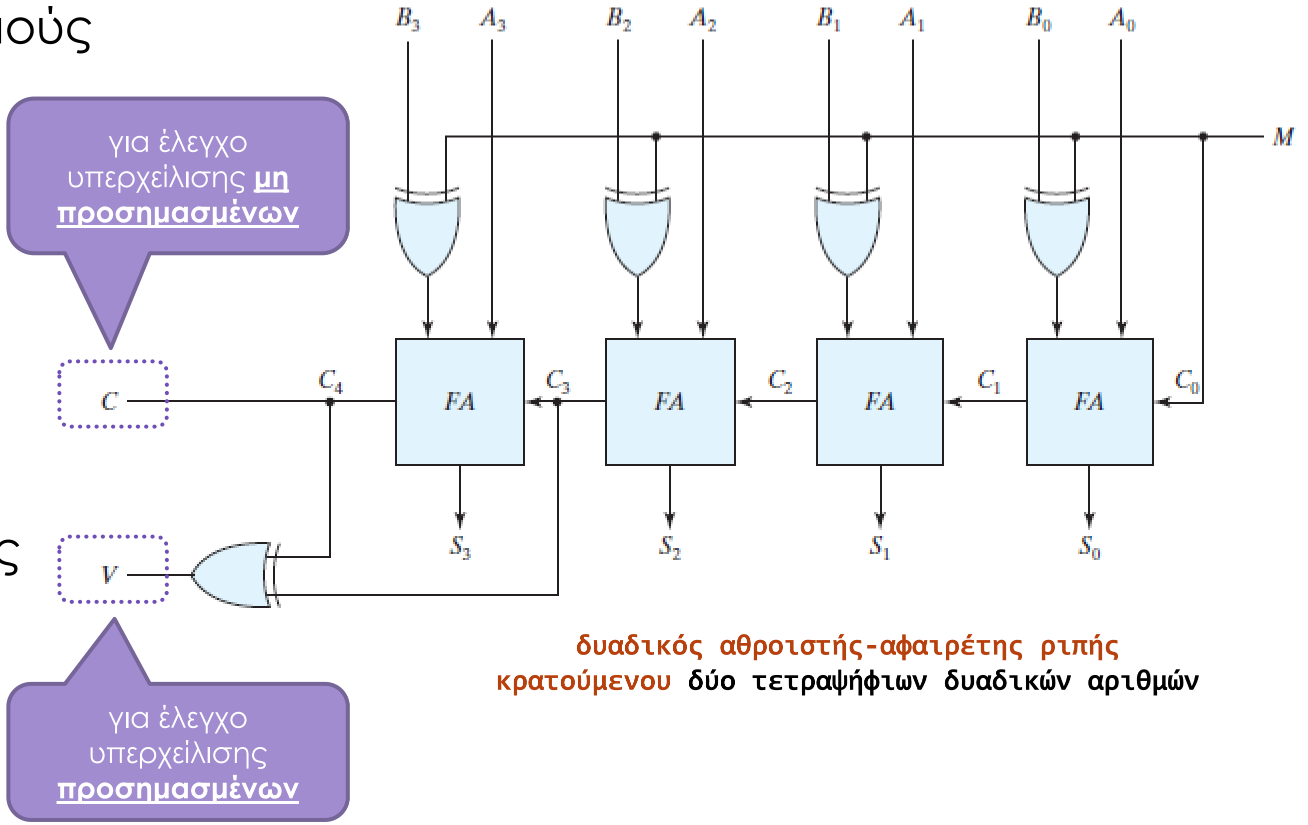
κρατούμενα	1	0	1	1	0	0	0	0	0
		1	0	1	1	1	0	1	0
+		1	0	1	1	0	0	0	0
άθροισμα		0	1	1	0	1	0	1	0

Δυαδικός αθροιστής-αφαιρέτης

Υπερχείλιση (overflow) - Εντοπισμός - Υλοποίηση

- το κύκλωμα του δυαδικού αθροιστή-αφαιρέτη έχει τις επιπλέον εξόδους **C** και **V**
- για μη προσημασμένους δυαδικούς αριθμούς
  - εάν ισχύει **C = 1**,
    - είτε έχει προκύψει μη μηδενικό κρατούμενο από την πρόσθεση
    - είτε έχει προκύψει μη μηδενικό δανεικό από την αφαίρεση

  → άρα, συνέβη υπερχείλιση
- για προσημασμένους δυαδικούς αριθμούς
  - εάν ισχύει **V = 1** → έχει συμβεί υπερχείλιση



δυαδικός αθροιστής-αφαιρέτης ριπής κρατούμενου δύο τετραψήφιων δυαδικών αριθμών

Δεκαδικός αθροιστής

Σχεδίαση

Δεκαδικός αθροιστής

- οι υπολογιστές ή αριθμομηχανές που εκτελούν αριθμητικές πράξεις απευθείας στο δεκαδικό σύστημα → χρησιμοποιούν δυαδικό κώδικα για την παράσταση των δεκαδικών αριθμών

- στη δυαδική πρόσθεση
  - εξετάσαμε την πρόσθεση δύο (ανάλογης τάξης) bit και ενός (προηγούμενου) κρατουμένου
  - το αντίστοιχο ψηφιακό κύκλωμα → προσθέτει τρία bit και παράγει δύο bit

- στη δεκαδική πρόσθεση
  - κάθε δεκαδικό ψηφίο κωδικοποιείται σε τέσσερα δυαδικά ψηφία
  - το αντίστοιχο ψηφιακό κύκλωμα
    - έχει εννέα εισόδους (2 δεκαδικά ψηφία και 1 κρατούμενο εισόδου)
    - πέντε εξόδους (1 δεκαδικό ψηφίο και 1 κρατούμενο εξόδου)

Αθροιστής BCD
Σχεδίαση

- εξετάζουμε την περίπτωση κωδικοποίησης των δεκαδικών ψηφίων με χρήση κώδικα BCD
- έστω ότι χρησιμοποιούμε τον δυαδικό αθροιστή τεσσάρων ψηφίων για την πρόσθεση δύο κωδικοποιημένων δεκαδικών ψηφίων
  1. καταγράφουμε όλες τις δυνατές εξόδους του δυαδικού αθροιστή
     - δεκαδικά ψηφία: **0**, **1**, ..., **9**
     - BCD ψηφία: **0000**, **0001**, ..., **1001**
     - ελάχιστη τιμή εξόδου του δυαδικού αθροιστή 0_{10}: κρατούμενο 0 | 0 0 0 0 ψηφία
       - επειδή: $0_{10} + 0_{10} + 0_{10} = 0_{10}$
     - μέγιστη τιμή εξόδου του δυαδικού αθροιστή 19_{10}: κρατούμενο 1 | 0 0 1 1 ψηφία
       - επειδή: $9_{10} + 9_{10} + 1_{10} = 19_{10}$
  2. καταγράφουμε όλες τις επιθυμητές εξόδους ενός αθροιστή BCD

Αθροιστής BCD

Σχεδίαση (II)

δυαδικό άθροισμα και άθροισμα BCD είναι ίδια

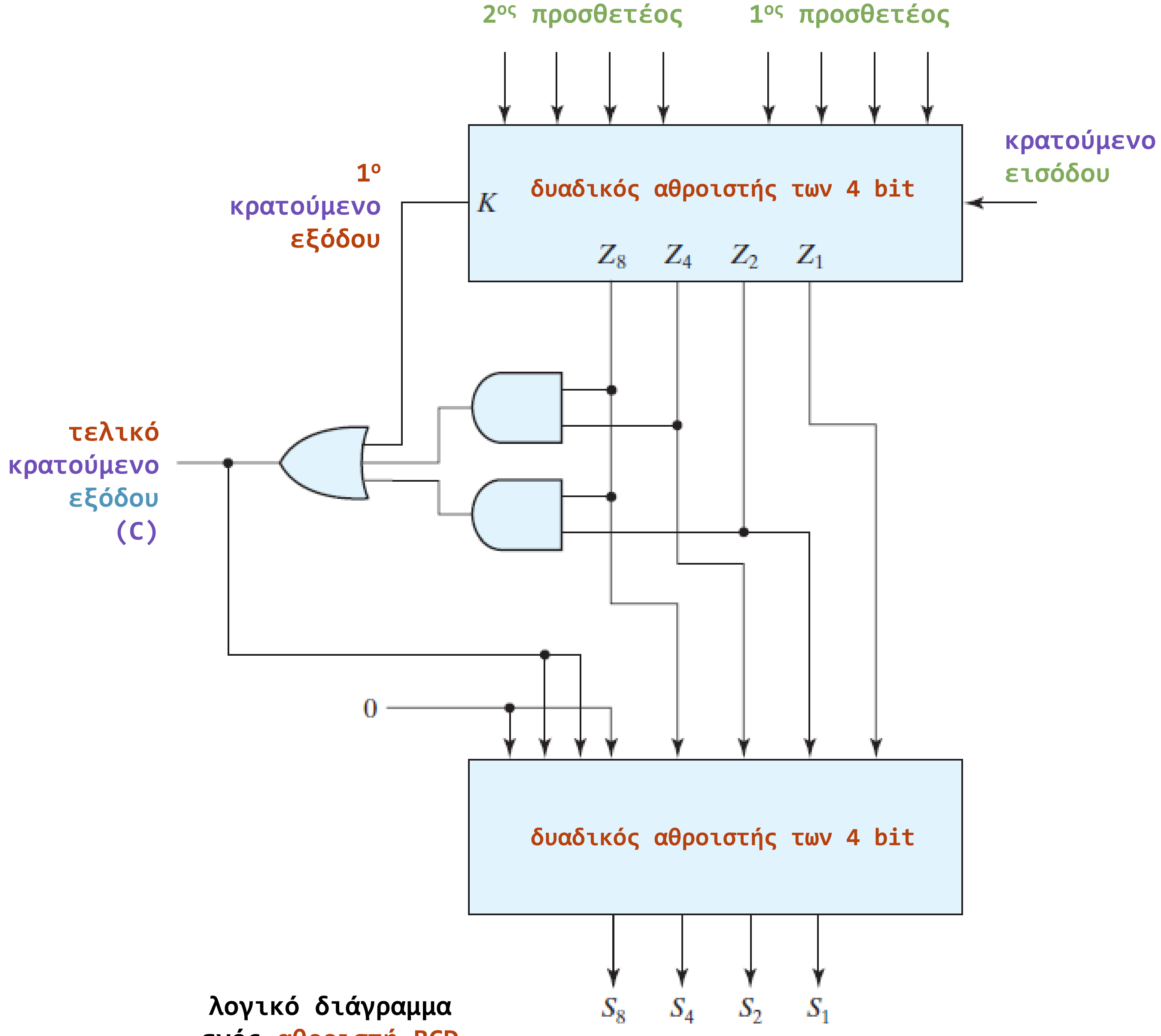
- ✓ δυαδικό άθροισμα μικρότερο του 9_{10} → **σωστή** παράσταση BCD
- ✗ εάν το δυαδικό είναι μεγαλύτερο του του 9_{10} → **λάθος** παράσταση BCD
  - ▶ πρέπει να προστεθεί ο αριθμός 6_{10} στο δυαδικό άθροισμα
- ❖ προσπαθούμε να βρούμε **κανόνες**, ώστε το άθροισμα που παράγει ο δυαδικός αρθοιστής να μπορεί να **μετατραπεί** στη σωστή **παράσταση BCD**
  - ✗ K = **1** (όπου K είναι το κρατούμενο εξόδου του δυαδικού αθροιστή) → **λάθος** παράσταση BCD
  - ✗ Z_8 = **1** και Z_4 = **1** → **λάθος** παράσταση BCD
  - ✗ Z_8 = **1** και Z_2 = **1** → **λάθος** παράσταση BCD
- ❖ οπότε, η **συνθήκη** για να γίνει η διόρθωση (+6_{10}) και να δημιουργηθεί κρατούμενο εξόδου εκφράζεται από τη συνάρτηση: $C = K + Z_8Z_4 + Z_8Z_2$

	δυαδικό άθροισμα					άθροισμα BCD				
Δεκαδικός	K	Z_8	Z_4	Z_2	Z_1	C	S_4	S_3	S_2	S_1
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

Αθροιστής BCD
Σχεδίαση (ΙΙΙ)

- ισχύει ότι: **C** = **K** + Z_8Z_4 + Z_8Z_2
- όταν **C = 0** → ο κάτω αθροιστής προσθέτει στο δυαδικό άρθοισμα που παράγει ο πάνω αθροιστής ($Z_8Z_4Z_2Z_1$) την τιμή **0000** → άρα, δεν το μεταβάλει
- όταν **C = 1** → ο κάτω αθροιστής προσθέτει στο δυαδικό άρθοισμα που παράγει ο πάνω αθροιστής ($Z_8Z_4Z_2Z_1$) την τιμή **0110** (= 6_{10})→ άρα, μετατρέπει το άθροισμα στο **σωστό BCD ψηφίο**

✍ το κρατούμενο που παράγεται από τον κάτω αθροιστή μπορεί να αγνοηθεί
  - είναι ίδιο με το τελικό κρατούμενο εξόδου (C)



λογικό διάγραμμα ενός αθροιστή BCD

Δεκαδικός αθροιστής
Σχεδίαση

- για να κατασκευαστεί ένας **δεκαδικός αθροιστής**,

    ο οποίος προσθέτει **n** δεκαδικά ψηφία,

  χρειάζονται **n** στάδια **αρθοιστή BCD**

  - το κρατούμενο εξόδου που παράγεται στο κάθε στάδιο → τροφοδοτείται ως κρατούμενο εισόδου του επόμενου, υψηλότερου επιπέδου, σταδίου

Δυαδικός πολλαπλασιαστής

Σχεδίαση

Πολλαπλασιασμός

- ο πολλαπλασιαμός δυαδικών αριθμών είναι όμοιος με τον πολλαπλασιασμό δεκαδικών αριθμών

  π.χ. 11_{10} * 3_{10}

			1	0	1	1
*				1	0	1
			1	0	1	1
		0	0	0	0	
+	1	0	1	1		
γινόμενο	1	1	0	1	1	1

Δυαδικός Πολλαπλασιαστής

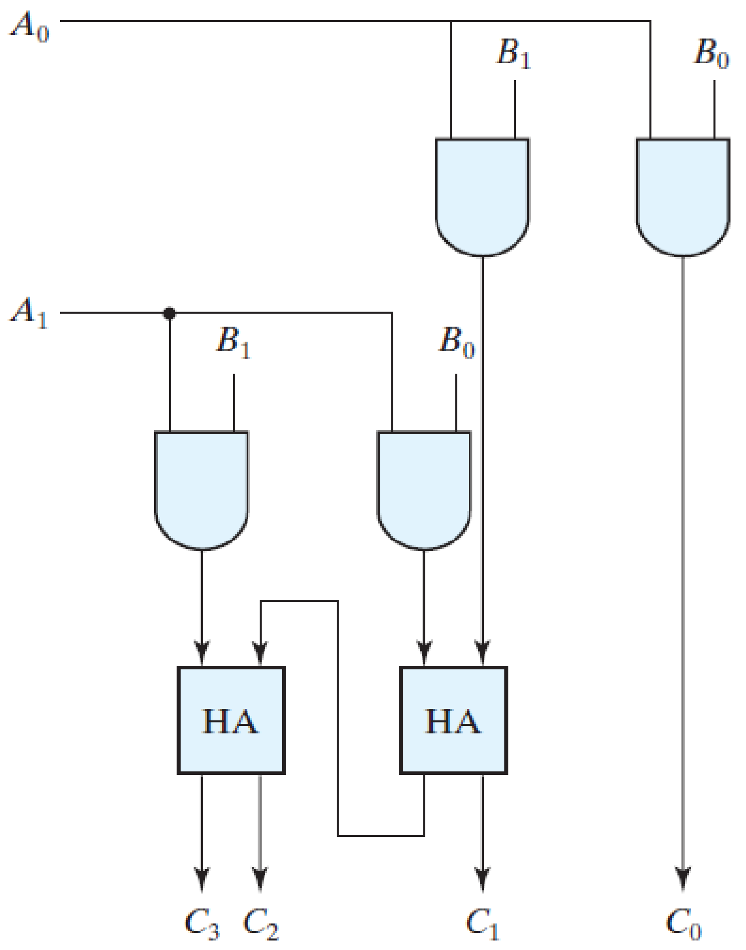
Σχεδιασμός - 2*2 bit

- Θέλουμε να πολλαπλασιάσουμε το διψήφιο δυαδικό αριθμό **B** (B_1B_0) με το διψήφιο δυαδικό αριθμό **A** (A_1A_0) και έστω **C** ($C_3C_2C_1C_0$) το γινόμενό τους, οπότε:
  - $C_0 = A_0B_0$
  - $C_1 = A_0B_1 + A_1B_0$
  - $C_2 = A_1B_1$ + **κρατούμενο εξόδου** από προηγούμενο άρθοισμα
  - C_3 = **κρατούμενο εξόδου** από το προηγούμενο άρθοισμα

πύλες AND

ημιαθροιστές (HA)

			B_1	B_0
*			A_1	A_0
			A_0B_1	A_0B_0
+		A_1B_1	A_1B_0	
γινόμενο	C_3	C_2	C_1	C_0



λογικό διάγραμμα ενός πολλαπλασιαστή δύο αριθμών των 2 bit

Δυαδικός Πολλαπλασιαστής

Σχεδιασμός - πολλαπλών bit

- ένας δυαδικός πολλαπλασιαστής δύο αριθμών που έχουν περισσότερα από δύο bit μπορεί να κατασκευαστεί με ανάλογο τρόπο
  1. κάθε bit του πολλαπλασιαστή
     - περνάει από πύλες **AND** με όλα τα bit του πολλαπλασιαστέου
     - σε τόσα επίπεδα όσα και τα bit του πολλαπλασιαστή
  2. η δυαδική έξοδος κάθε επιπέδου πυλών **AND** προστίθεται με το μερικό γινόμενο του προηγούμενου επιπέδου → οπότε, προκύπτει ένα νέο μερικό γινόμενο
  3. στο τελευταίο επίπεδο παράγεται το ζητούμενο γινόμενο

- για **J** bit πολλαπλασιαστή και **K** bit πολλαπλασιαστέο χρειαζόμαστε
  - **J*K** πύλες **AND**
  - **J-1** αθροιστές των **K** bit

  ώστε να παραχθεί ένα γινόμενο των **J+K** bit

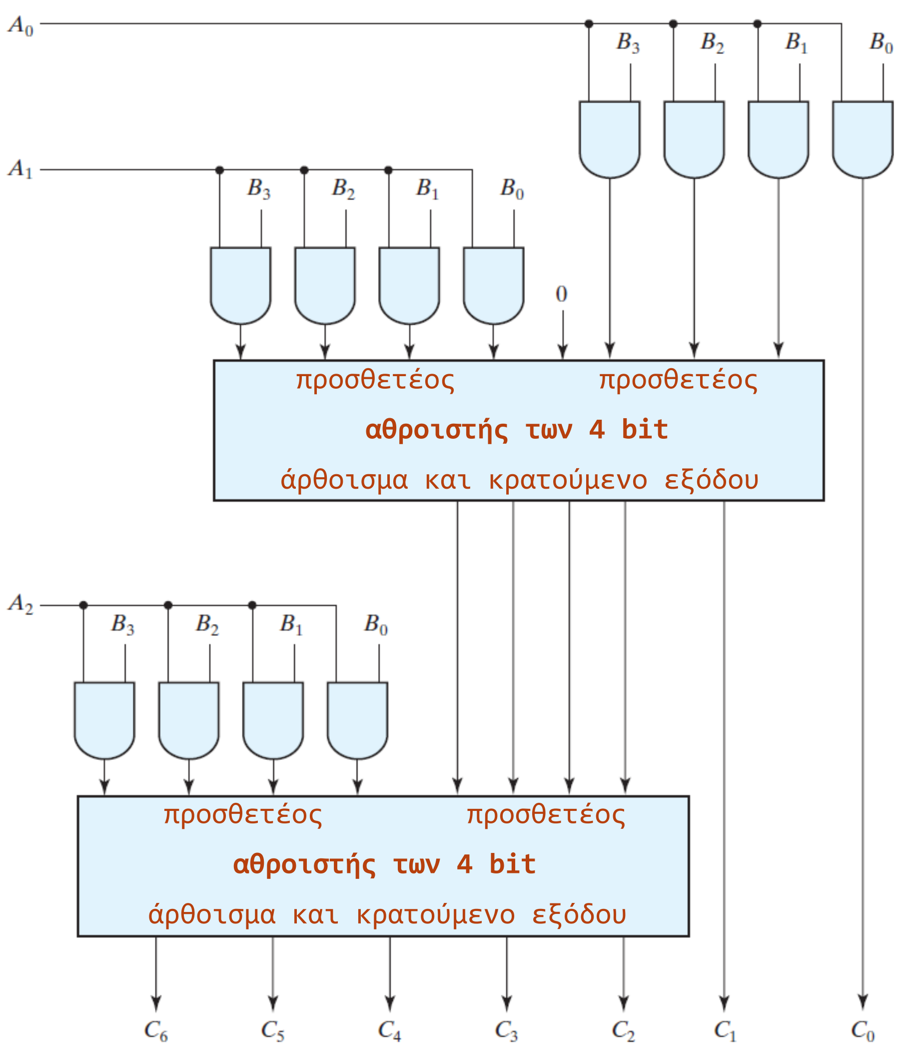
Δυαδικός πολλαπλασιαστής
Σχεδιασμός - 4*3 bit

- πολλαπλασιαστέος **B** (**$B_3B_2B_1B_0$**)
- πολλαπλασιαστής **A** (**$A_2A_1A_0$**)
- καθώς **J=3** και **K=4** χρειαζόμαστε
  - (J*K=) **12** πύλες **AND**
  - (J-1=) **2** αθροιστές των τεσσάρων bit

ώστε να παράγουμε το γινόμενο των (J+K=) **7** bit

				B_3	B_2	B_1	B_0
*					A_2	A_1	A_0
				A_0B_3	A_0B_2	A_0B_1	A_0B_0
			A_1B_3	A_1B_2	A_1B_1	A_1B_0	
+		A_2B_3	A_2B_2	A_2B_1	A_2B_0		
γινόμενο	C_6	C_5	C_4	C_3	C_2	C_1	C_0

λογικό διάγραμμα ενός πολλαπλασιαστή δύο αριθμών με 4 και 3 bit



Συγκριτής μεγέθους

Σχεδίαση

Συγκριτής μεγέθους
Σχεδίαση

- η σύγκριση δύο αριθμών προσδιορίζει εάν ο ένας από αυτούς είναι μεγαλύτερος, ή μικρότερος από το δεύτερο, ή ίσος με το δεύτερο
- ο συγκριτής μεγέθους είναι ένα συνδυαστικό κύκλωμα που συγκρίνει δύο αριθμούς **A** και **B**
  - το αποτέλεσμα της σύγκρισης δίνεται από τρεις παραγόμενες δυαδικές μεταβλητές
    1. **(A=B)**, η οποία έχει την τιμή **1** όταν ο **A** είναι ίσος με τον **B**, αλλιώς έχει την τιμή **0**
    2. **(A>B)**, η οποία έχει την τιμή **1** όταν ο **A** είναι μεγαλύτερος του **B**, αλλιώς έχει την τιμή **0**
    3. **(A<B)**, η οποία έχει την τιμή **1** όταν ο **A** είναι μικρότερος του **B**, αλλιώς έχει την τιμή **0**
  - έστω ότι **A** και **B** έχουν **n** ψηφία → ο πίνακας αληθείας του κυκλώματος έχει 2^{2n} γραμμές!
  - θα κατασκευάσουμε έναν αλγόριθμο για τη σύγκριση των αριθμών

Συγκριτής μεγέθους

Σχεδίαση - Αλγόριθμος για αριθμούς τεσσάρων bit

έστω οι αριθμοί **A** ($A_3A_2A_1A_0$) και **B** ($B_3B_2B_1B_0$)

- οι αριθμοί **A** και **B** είναι ίσοι όταν όλα τα ψηφία τους είναι ίσα, δηλαδή όταν ισχύουν όλα τα παρακάτω:
  - $A_3 = B_3 \rightarrow x_3 = A_3B_3 + A_3'B_3'$
  - $A_2 = B_2 \rightarrow x_2 = A_2B_2 + A_2'B_2'$
  - $A_1 = B_1 \rightarrow x_1 = A_1B_1 + A_1'B_1'$
  - $A_0 = B_0 \rightarrow x_0 = A_0B_0 + A_0'B_0'$

☞ οπότε ισχύει ότι:

$$(A=B) = x_3x_2x_1x_0$$
$$= (A_3B_3 + A_3'B_3')(A_2B_2 + A_2'B_2')(A_1B_1 + A_1'B_1')(A_0B_0 + A_0'B_0')$$

Συγκριτής μεγέθους
Σχεδίαση - Αλγόριθμος για αριθμούς τεσσάρων bit (II)

έστω οι αριθμοί **A** ($A_3A_2A_1A_0$) και **B** ($B_3B_2B_1B_0$)

- ο αριθμός **A** είναι μεγαλύτερος του **B** όταν ισχύει τουλάχιστον ένα από τα παρακάτω:
  1. $A_3 > B_3 \rightarrow y_3 = A_3B_3'$ ή
  2. $A_3 = B_3$ (δηλαδή, $x_3 = 1$) και $A_2 > B_2 \rightarrow y_2 = x_3A_2B_2'$ ή
  3. $A_3 = B_3$ (δηλαδή, $x_3 = 1$) και $A_2 = B_2$ (δηλαδή, $x_2 = 1$) και $A_1 > B_1 \rightarrow y_1 = x_3x_2A_1B_1'$ ή
  4. $A_3 = B_3$ (δηλαδή, $x_3 = 1$) και $A_2 = B_2$ (δηλαδή, $x_2 = 1$) και $A_1 = B_1$ (δηλαδή, $x_1 = 1$) και $A_0 > B_0 \rightarrow y_0 = x_3x_2x_1A_0B_0'$

☞ οπότε ισχύει ότι:
$$(A>B) = y_3 + y_2 + y_1 + y_0$$
$$= A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

Συγκριτής μεγέθους

Σχεδίαση - Αλγόριθμος για αριθμούς τεσσάρων bit (III)

έστω οι αριθμοί A ($A_3A_2A_1A_0$) και B ($B_3B_2B_1B_0$)

- ο αριθμός A είναι μικρότερος του B όταν ισχύει τουλάχιστον ένα από τα παρακάτω:
  1. $A_3 < B_3 \rightarrow z_3 = A_3'B_3$ ή
  2. $A_3 = B_3$ (δηλαδή, $x_3 = 1$) και $A_2 > B_2 \rightarrow z_2 = x_3A_2'B_2$ ή
  3. $A_3 = B_3$ (δηλαδή, $x_3 = 1$) και $A_2 = B_2$ (δηλαδή, $x_2 = 1$) και $A_1 > B_1 \rightarrow z_1 = x_3x_2A_1'B_1$ ή
  4. $A_3 = B_3$ (δηλαδή, $x_3 = 1$) και $A_2 = B_2$ (δηλαδή, $x_2 = 1$) και $A_1 = B_1$ (δηλαδή, $x_1 = 1$) και $A_0 > B_0 \rightarrow z_0 = x_3x_2x_1A_0'B_0$

☞ οπότε ισχύει ότι: $(A<B) = z_3 + z_2 + z_1 + z_0$

$$= A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

Συγκριτής μεγέθους

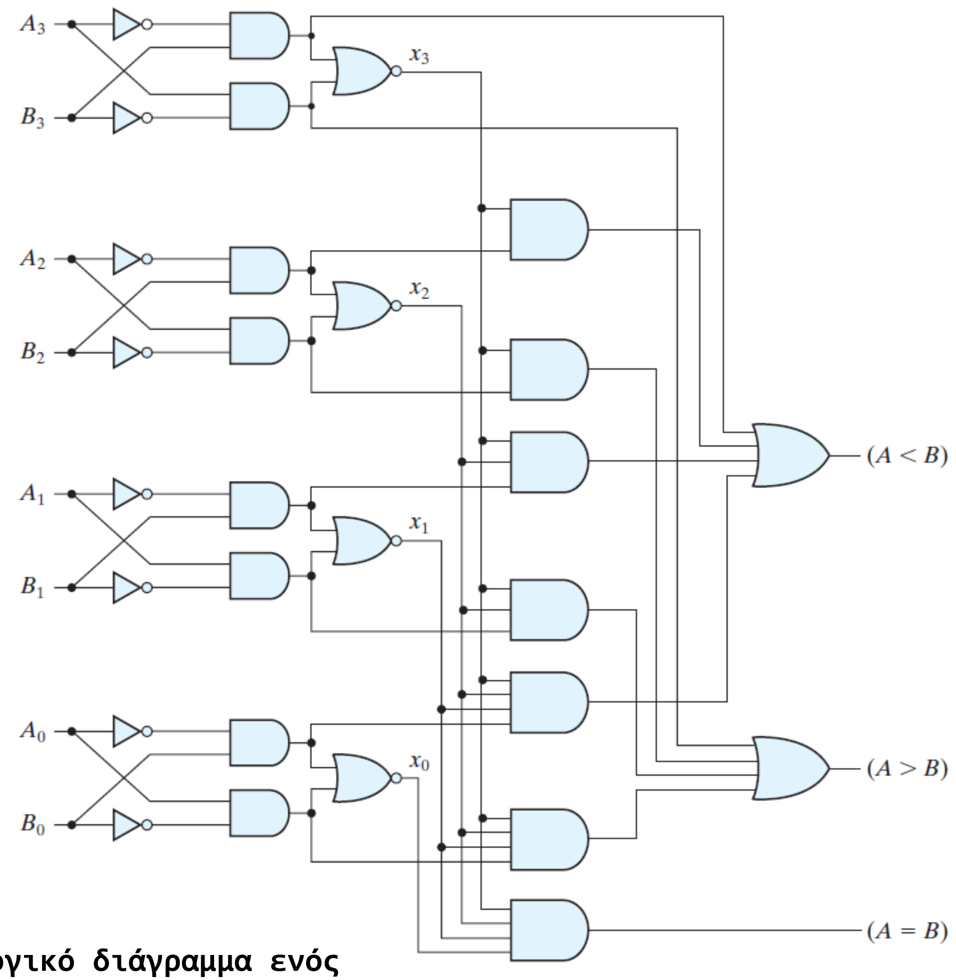
Σχεδίαση - Ψηφιακό κύκλωμα για αριθμούς τεσσάρων bit

<u>ισχύουν</u>:

- $x_i = A_iB_i + A_i'B_i'$, για $i = 0, 1, 2, 3$
- $(A=B) = x_3x_2x_1x_0$
- $(A>B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$
- $(A<B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$

✍ το λογικό διάγραμμα δεν είναι αρκετά πολύπλοκο, καθώς οι συναρτήσεις Boole επαναχρησιμοποιούν αρκετούς όρους

✍ η σχεδίαση κυκλωμάτων συγκριτών δυαδικών αριθμών που έχουν περισσότερα από τέσσερα bit γίνεται με άμεση επέκταση του αλγορίθμου που σχεδιάσαμε



λογικό διάγραμμα ενός συγκριτή μεγέθους τεσσάρων bit

Αποκωδικοποιητής

Σχεδίαση

Αποκωδικοποιητής (decoder)

- τα ψηφιακά συστήματα χρησιμοποιούν δυαδικούς κώδικες για την παράσταση διακριτής πληροφορίας
  - ένας δυαδικός κώδικας των **n** bit → παριστά έως **2^n** διακριτά στοιχεία κωδικοποιημένης πληροφορίας

- ο <u>αποκωδικοποιητής</u> είναι ένα συνδυαστικό κύκλωμα που μετατρέπει
  - κωδικοποιημένη δυαδική πληροφορία, η οποία έρχεται σε **n** γραμμές εισόδου

  σε

  - ισοδύναμη πληροφορία που τοποθετείται σε διακριτές γραμμές εξόδου
    - οι οποίες μπορεί να είναι το πολύ **2^n**

- εάν στην κωδικοποιημένη πληροφορία δε χρησιμοποιούνται κάποιοι από τους δυνατούς συνδυασμούς → ο αποκωδικοποιητής μπορεί να έχει λιγότερες από **2^n** γραμμές εξόδου

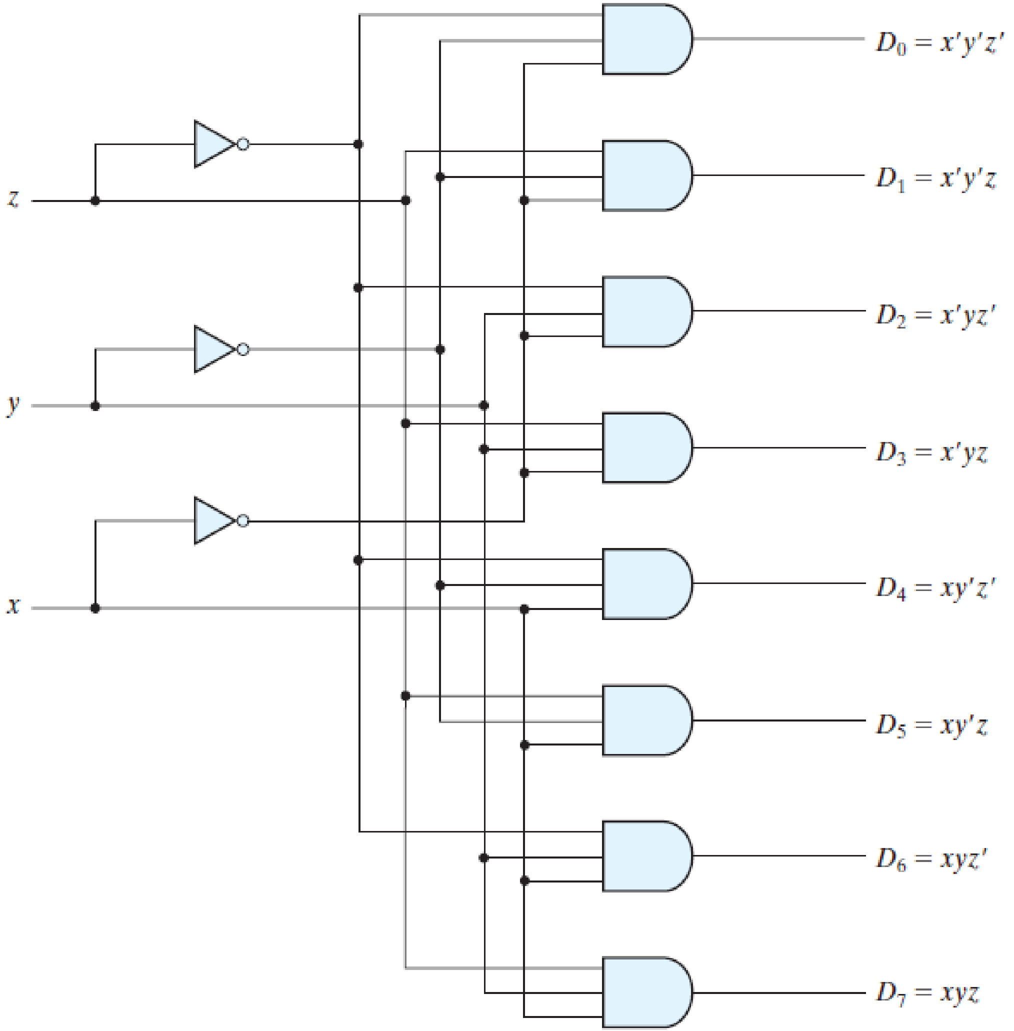
Αποκωδικοποιητής

Σχεδίαση

- σκοπός: η παραγωγή **2**^{n} (ή λιγότερων) ελαχιστόρων των **n** μεταβλητών
  - δίνουμε το όνομα αποκωδικοποιητής **n** γραμμών σε **m** γραμμές (ή απλά αποκωδικοποιητής **n**-σε-**m**), όπου **m** ≤ **2**^{n}

π.χ. αποκωδικοποιητής **3**-σε-**8**

- μία εφαρμογή του είναι η μετατροπή ενός δυαδικού αριθμού σε οκταδικό
  - οι μεταβλητές εισόδου: παριστάνουν ένα δυαδικό αριθμό
  - κάθε έξοδος: παριστάνει ένα οκταδικό ψηφίο



λογικό διάγραμμα ενός αποκωδικοποιητή 3-σε-8

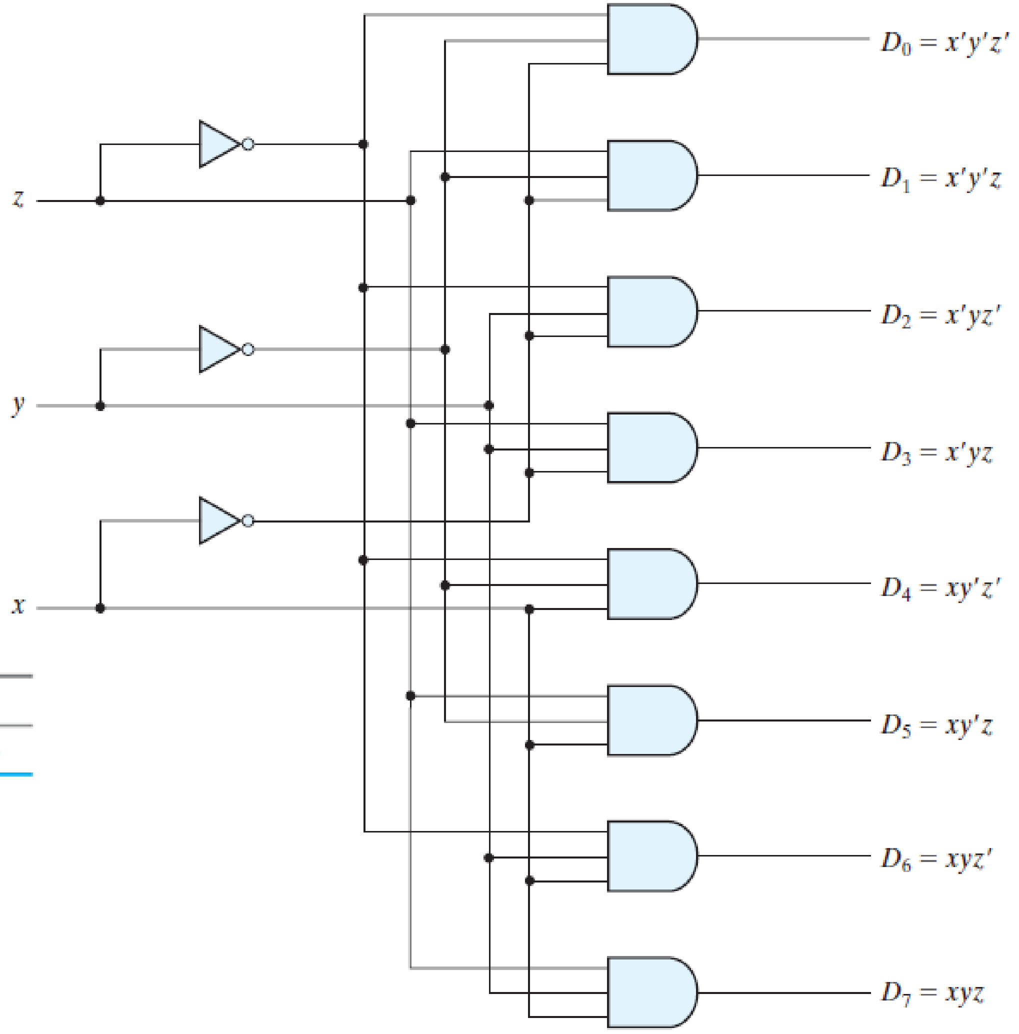
Αποκωδικοποιητής

Σχεδίαση - Πίνακας αληθείας

π.χ. αποκωδικοποιητής 3-σε-8

- μία εφαρμογή του είναι η μετατροπή ενός δυαδικού αριθμού σε οκταδικό
  - οι μεταβλητές εισόδου: παριστάνουν ένα δυαδικό αριθμό
  - κάθε έξοδος: παριστάνει ένα οκταδικό ψηφίο
- εξετάζουμε τον πίνακα αληθείας, ώστε να κατανοήσουμε καλύτερα τη λειτουργία του

είσοδοι			έξοδοι							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_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

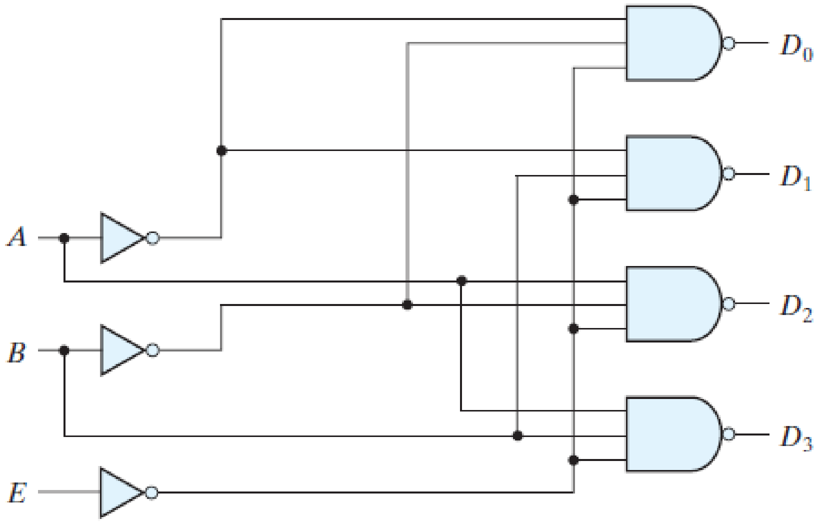


λογικό διάγραμμα ενός αποκωδικοποιητή 3-σε-8

Αποκωδικοποιητής
Υλοποίηση με πύλες NAND & είσοδο επίτρεψης (enable)

- αρκετοί αποκωδικοποιητές κατασκευάζονται με πύλες **NAND**
  - η πύλη **NAND** εκτελεί τη λογική πράξη **AND** και αντιστρέφει την προκύπτουσα έξοδο
  - οπότε, είναι πιο οικονομικό οι ελαχιστόροι που προκύπτουν από τον αποκωδικοποιητή να παράγονται στη συμπληρωμένη τους μορφή
- συνήθως, οι αποκωδικοποιητές περιλαμβάνουν μία ή περισσότερες εισόδους επίτρεψης (enable) που ελέγχουν τη λειτουργία του κυκλώματος
- π.χ. αποκωδικοποιητής **2**-σε-**4** με είσοδο επίτρεψης (**E**)
  - λειτουργεί όταν **E = 0**
  - το κύκλωμα απενεργοποιείται όταν **E = 1**, ανεξάρτητα από τις τιμές των άλλων δύο εισόδων

είσοδοι			έξοδοι			
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



λογικό διάγραμμα ενός αποκωδικοποιητή 2-σε-4 με είσοδο επίτρεψης

Αποκωδικοποιητής

Παρατηρήσεις

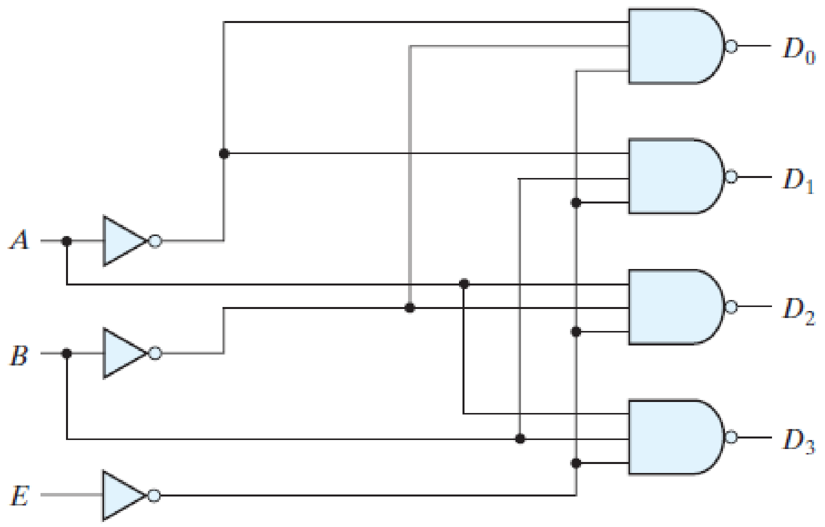
γενικά, ένας αποκωδικοποιητής μπορεί να έχει:

- εξόδους σε συμπληρωμένη ή κανονική μορφή
- είσοδο επίτρεψης που ενεργοποιείται είτε με την τιμή **0** είτε με την τιμή **1**
- δύο ή περισσότερες εισόδους επίτρεψης
  - για να ενεργοποιηθεί το κύκλωμα → οι τιμές που τίθενται στις εισόδους επίτρεψης πρέπει να ικανοποιούν μια συγκεκριμένη λογική συνθήκη

Αποκωδικοποιητής-Αποπλέκτης

- ο αποπλέκτης (demultiplexer) είναι ένα συνδυαστικό κύκλωμα που λαμβάνει πληροφορία από μία γραμμή και την προωθεί σε μία από **2^n** πιθανές γραμμές
  - η επιλογή μιας συγκεκριμένης εξόδου γίνεται με βάση το συνδυασμό **n** τιμών εισόδων → οι οποίες ονομάζονται γραμμές επιλογής ή είσοδοι επιλογής

- ένας αποκωδικοποιητής με είσοδο επίτρεψης μπορεί να λειτουργήσει και ως αποπλέκτης
  - π.χ. ο αποκωδικοποιητής **2**-σε-**4** με είσοδο επίτρεψης (**E**) μπορεί να λειτουργήσει και ως αποπλέκτης **1**-σε-**4** , αν:
    1. το **E** θεωρηθεί ως γραμμή εισόδου των δεδομένων
    2. και τα **A**, **B** θεωρηθούν ως είσοδοι επιλογής



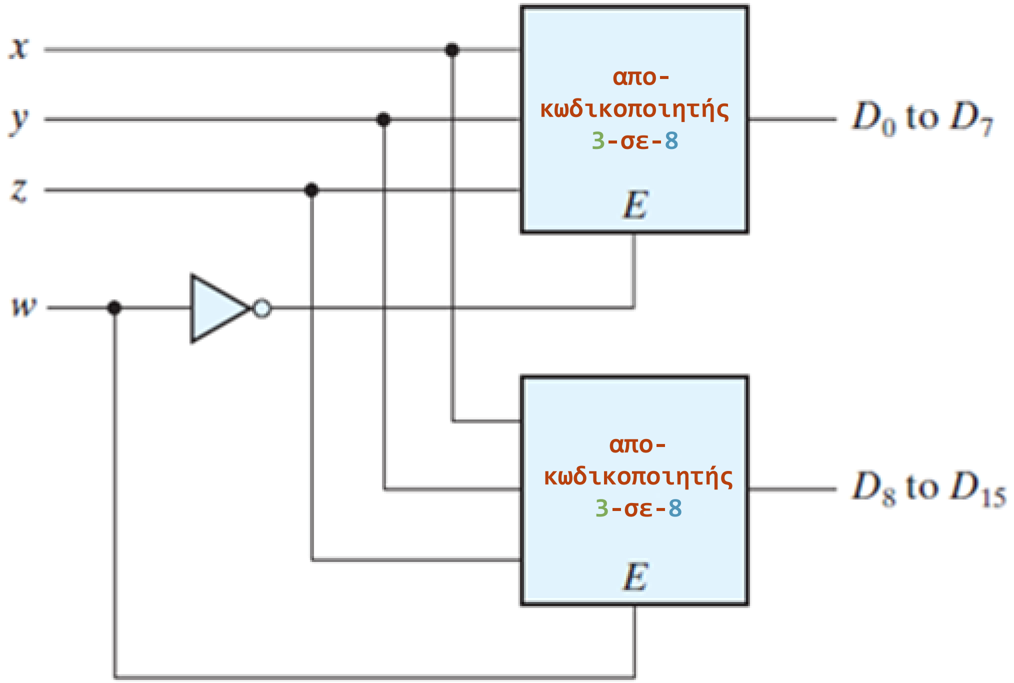
λογικό διάγραμμα ενός αποκωδικοποιητή 2-σε-4 με είσοδο επίτρεψης ή αποπλέκτη 1-σε-4 ή αποκωδικοποιητή-αποπλέκτη

- ✍ καθώς οι λειτουργίες του αποκωδικοποιητή και του αποπλέκτη εκτελούνται από το ίδιο κύκλωμα → ένας αποκωδικοποιητής με είσοδο επίτρεψης ονομάζεται: **αποκωδικοποιητής-αποπλέκτης**

Αποκωδικοποιητής
Διασύνδεση αποκωδικοποιητών με εισόδους επίτρεψης

οι αποκωδικοποιητές με εισόδους επίτρεψης μπορούν να διασυνδεθούν ώστε να δημιουργήσουν ένα μεγαλύτερο κύκλωμα αποκωδικοποιητή

- π.χ. δύο αποκωδικοποιητές 3-σε-8 με είσοδο επίτρεψης συνδέονται κατάλληλα ώστε να δημιουργήσουν έναν αποκωδικοποιητές 4-σε-16
  - όταν ισχύει `w = 0` → ενεργοποιείται ο πάνω αποκωδικοποιητής, ενώ ο κάτω απενεργοποιείται
    - όλες οι έξοδοι του κάτω αποκωδικοποιητή είναι `0`
    - οι οκτώ έξοδοι του πάνω αποκωδικοποιητή παράγουν έναν από τους ελαχιστόρους: `0000` έως `0111`
  - όταν ισχύει `w = 1` → οι συνθήκες αντιστρέφονται
    - οι οκτώ έξοδοι του κάτω αποκωδικοποιητή παράγουν έναν από τους ελαχιστόρους: `1000` έως `1111`
    - όλες οι έξοδοι του πάνω αποκωδικοποιητή είναι `0`



λογικό διάγραμμα ενός αποκωδικοποιητή 4-σε-16
(που υλοποιείται από δύο αποκωδικοποιητές 3-σε-8 με είσοδο επίτρεψης)

Αποκωδικοποιητής
Υλοποίηση συνδυαστικής λογικής

ισχύουν τα εξής:

- ο αποκωδικοποιητής παράγει τους **2^n** ελαχιστόρους των **n** μεταβλητών εισόδου
- κάθε ενεργή έξοδος του αποκωδικοποιητή σχετίζεται με ένα μοναδικό συνδυασμό των εισόδων
- οποιαδήποτε συνάρτηση Boole μπορεί να εκφραστεί σε μορφή αθροίσματος ελαχιστόρων

επομένως, για την υλοποίηση μιας συνάρτησης Boole **F** μπορούμε:

1. να χρησιμοποιήσουμε έναν αποκωδικοποιητή που παράγει τους ελαχιστόρους των μεταβλητών της **F** και
2. να αθροίσουμε λογικά τους ελαχιστόρους που ανήκουν στην **F**
   - εάν ο αποκωδικοποιητής είναι υλοποιημένος με πύλες **AND** και **OR** → χρησιμοποιούμε μία πύλη **OR** για το εν λόγω λογικό άθροισμα
   - εάν ο αποκωδικοποιητής είναι υλοποιημένος με πύλες **NAND** → χρησιμοποιούμε μία πύλη **NAND** για το εν λόγω λογικό άθροισμα

Αποκωδικοποιητής

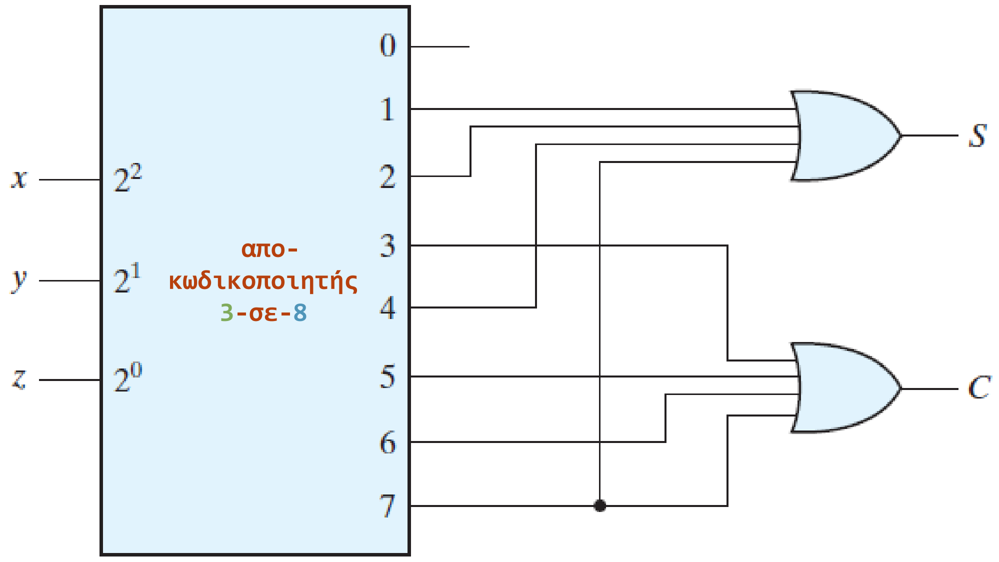
Υλοποίηση συνδυαστικής λογικής - Πλήρης αθροιστής

x	y	z	C	S
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

ο πλήρης αθροιστής

- δέχεται τρεις δυαδικές εισόδους **x**, **y** και **z**
  - οπότε χρειαζόμαστε έναν αποκωδικοποιητή 3-σε-8
- παράγει δύο δυαδικές εξόδους **S** και **C**
  - για την τιμή αθροίσματος (**S**) και κρατουμένου (**C**)
- έχει το διπλανό πίνακα αληθείας και ισχύει:
  - `S(x,y,z) = Σ(1,2,4,7)`
  - `C(x,y,z) = Σ(3,5,6,7)`

- **υποθέτουμε ότι ο αποκωδικοποιητής υλοποιείται με πύλες AND και OR**
- **έτσι, χρησιμοποιούμε πύλες OR για να υλοποιήσουμε το λογικό άθροισμα των ελαχιστόρων των εξαρτημένων μεταβλητών S και C**



λογικό διάγραμμα ενός πλήρη αθροθστή
(που υλοποιείται με έναν αποκωδικοποιητή 3-σε-8)

Αποκωδικοποιητής

Υλοποίηση συνδυαστικής λογικής - Παρατήρηση

- για την υλοποίηση μιας συνάρτησης με πολλούς ελαχιστόρους απαιτείται μία πύλη **OR** (ή μία πύλη **NAND**) με μεγάλο αριθμό εισόδων

- μια συνάρτηση **F** που έχει **k** ελαχιστόρους → μπορεί να εκφραστεί στη συμπληρωμένη μορφή της (**F '**) me 2^n-**k** ελαχιστόρους

- εάν ο αριθμός ελαχιστόρων της **F** είναι μεγαλύτερος από $2^n/2$
  - η **F '** έχει λιγότερους ελαχιστόρους από την **F**
  - επιλέγουμε να χρησιμοποιήσουμε μία πύλη **NOR** (ή **AND**, αντίστοιχα) για να αθροίσουμε τους ελαχιστόρους της **F '**
  - η έξοδος της πύλης **NOR** (ή **AND**, αντίστοιχα) συμπληρώνει αυτό το άθροισμα → επομένως προκύπτει η έξοδος **F**

Κωδικοποιητής

Σχεδίαση

Κωδικοποιητής (encoder)

- εκτελεί την ανάστροφη λειτουργία από αυτή του αποκωδικοποιητή
- έχει **2**^{n} (ή λιγότερες) γραμμές εισόδου και **n** γραμμές εξόδου
  - όλες μαζί οι γραμμές εξόδου → παράγουν την κατάλληλη λέξη ενός δυαδικού κώδικα που αντιστοιχεί στην ενεργή γραμμή εισόδου

- π.χ. κωδικοποιητής οκταδικού ψηφίου σε δυαδική αναπαράσταση
  - έχει οκτώ εισόδους και τρεις εξόδους
  - υποθέτουμε ότι μόνο μία είσοδος έχει τιμή **1** σε κάθε χρονική στιγμή
  - ισχύουν:

1. $x = D_4 + D_5 + D_6 + D_7$
2. $y = D_2 + D_3 + D_6 + D_7$
3. $z = D_1 + D_3 + D_5 + D_7$

είσοδοι								έξοδοι		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

αυτός ο κωδικοποιητής μπορεί να υλοποιηθεί με τρεις πύλες OR των τεσσάρων εισόδων

Κωδικοποιητής
Ασάφειες

εἰσοδοι								έξοδοι		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

a) μόνο μία είσοδος έχει τιμή **1** σε κάθε χρονική στιγμή

- εάν δύο ή περισσότερες είσοδοι είναι ενεργές ταυτόχρονα → η έξοδος είναι (συνήθως) μία δυαδική λέξη που δίνει λανθασμένη πληροφορία
  - π.χ. εάν τα D_3 και D_6 γίνουν ταυτόχρονα **1** → η έξοδος του κωδικοποιητή θα είναι **111**
    - όμως η έξοδος **111** δεν παριστάνει ούτε το δυαδικό **3** ούτε το δυαδικό **6**
- λύση: πρέπει να ορίζεται μια σειρά προτεραιότητας των εισόδων → ώστε να εξασφαλιστεί ότι ανά πάσα στιγμή μόνο μία είσοδος κωδικοποιείται
  - π.χ. ορίζουμε ότι οι είσοδοι με μεγαλύτερους δείκτες έχουν μεγαλύτερη προτεραιότητα
    - τότε, ακόμη και αν τα D_3 και D_6 γίνουν ταυτόχρονα **1** → η έξοδος θα γίνει **110**, καθώς το D_6 έχει μεγαλύτερη προτεραιότητα από το D_3

b) όταν όλες οι είσοδοι είναι **0** → η έξοδος γίνεται **000**

- όμως η έξοδος **000** είναι αυτή που προκύπτει στην περίπτωση που το D_0 είναι **1**
- λύση: χρησιμοποιούμε μία ακόμη έξοδο → η οποία δείχνει ότι τουλάχιστον μία είσοδος είναι **1**

Κωδικοποιητής προτεραιότητας

- ένα είδος κωδικοποιητή, η λειτουργία του οποίου περιλαμβάνει και την έννοια της προτεραιότητας των εισόδων
  - εάν δύο ή περισσότεροι είσοδοι γίνουν ταυτόχρονα **1** → θα κωδικοποιηθεί η είσοδος που έχει τη μεγαλύτερη προτεραιότητα

π.χ.

  - έστω ο διπλανός πίνακας αληθείας ενός κωδικοποιητή τεσσάρων εισόδων
  - εκτός από τις δύο εξόδους (**x** και **y**), το κύκλωμα έχει και μία ακόμη έξοδο (**V**) που λειτουργεί ως ενδείκτης εγκυρότητας
    - εάν όλες οι είσοδοι είναι **0** → **V = 0** → έχουμε ένδειξη ότι η είσοδος δεν είναι έγκυρη
      - στην περίπτωση αυτή δε μας ενδιαφέρουν οι άλλες δύο έξοδοι → για το λόγο αυτό έχουν τιμή **X**
  - όσο μεγαλύτερη είναι η τιμή του δείκτη εισόδου → τόσο μεγαλύτερη είναι η προτεραιότητα της εισόδου

είσοδοι				έξοδοι		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Κωδικοποιητής προτεραιότητας
Παράδειγμα - Σχεδίαση

- τα **X** στον πίνακα αληθείας χρησιμεύουν ώστε να παρουσιαστεί ο πίνακας σε συμπυκνωμένη μορφή
  - π.χ. το `X100` → αντιστοιχεί στα `0100` και `1100`
- οπότε, ισχύουν:
  - $x = \Sigma(1,2,3,5,6,7,9,10,11,13,14,15)$
  - $y = \Sigma(1,3,4,5,7,9,11,12,13,15)$

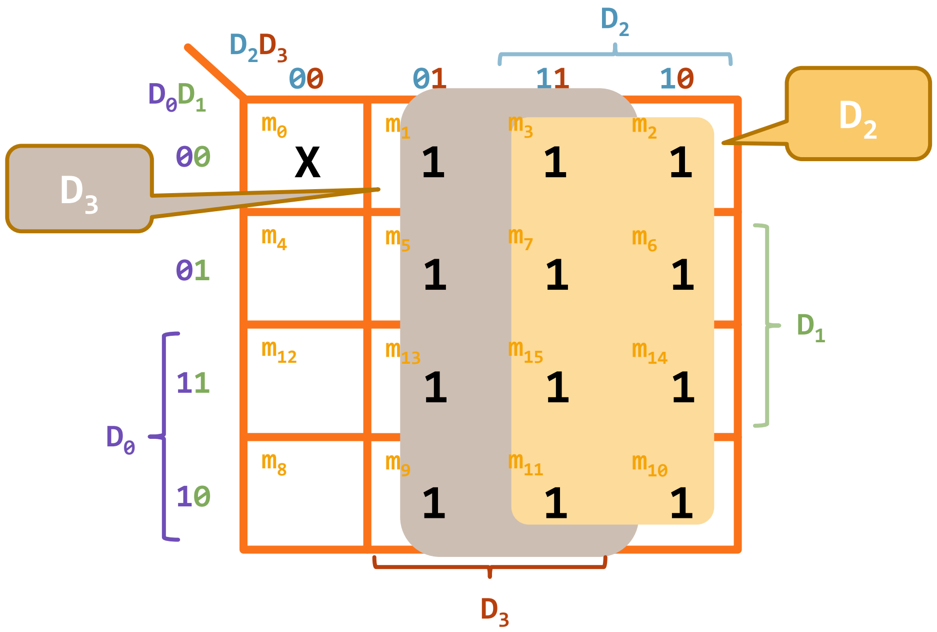
  με κοινή συνθήκη αδιαφορίας $d = \Sigma(0)$

είσοδοι				έξοδοι		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

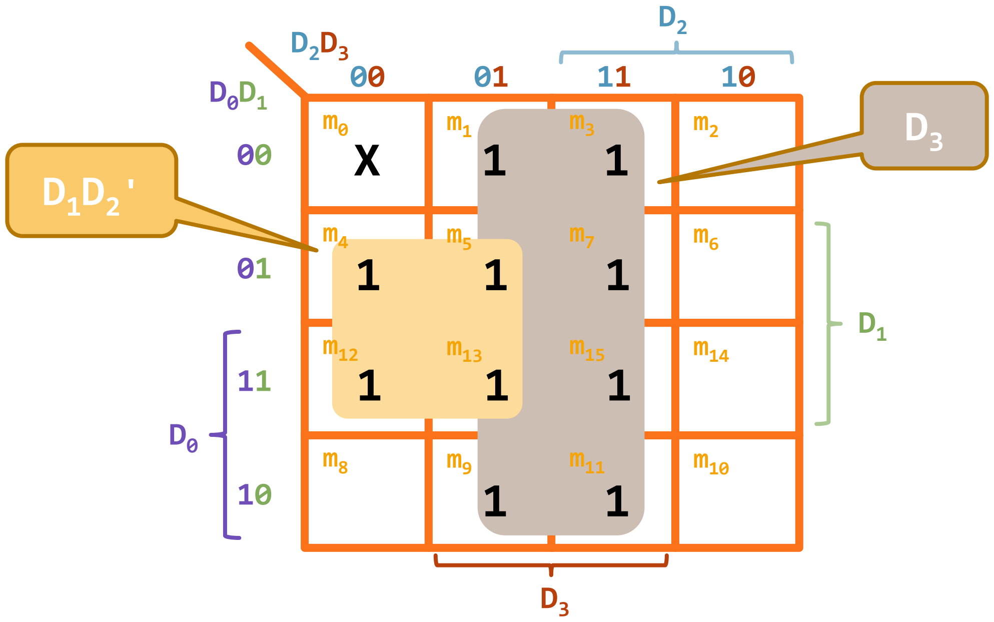
Κωδικοποιητής προτεραιότητας
Παράδειγμα - Σχεδίαση - Απλοποίηση συναρτήσεων εξόδου

- εύρεση απλοποιημένης συνάρτησης Boole για κάθε έξοδο

$$d(D_0,D_1,D_2,D_3) = \Sigma(0)$$
$$x(D_0,D_1,D_2,D_3) = \Sigma(1,2,3,5,6,7,9,10,11,13,14,15) = D_2 + D_3$$



$$d(D_0,D_1,D_2,D_3) = \Sigma(0)$$
$$y(D_0,D_1,D_2,D_3) = \Sigma(1,3,4,5,7,9,11,12,13,15) = D_1D_2' + D_3$$



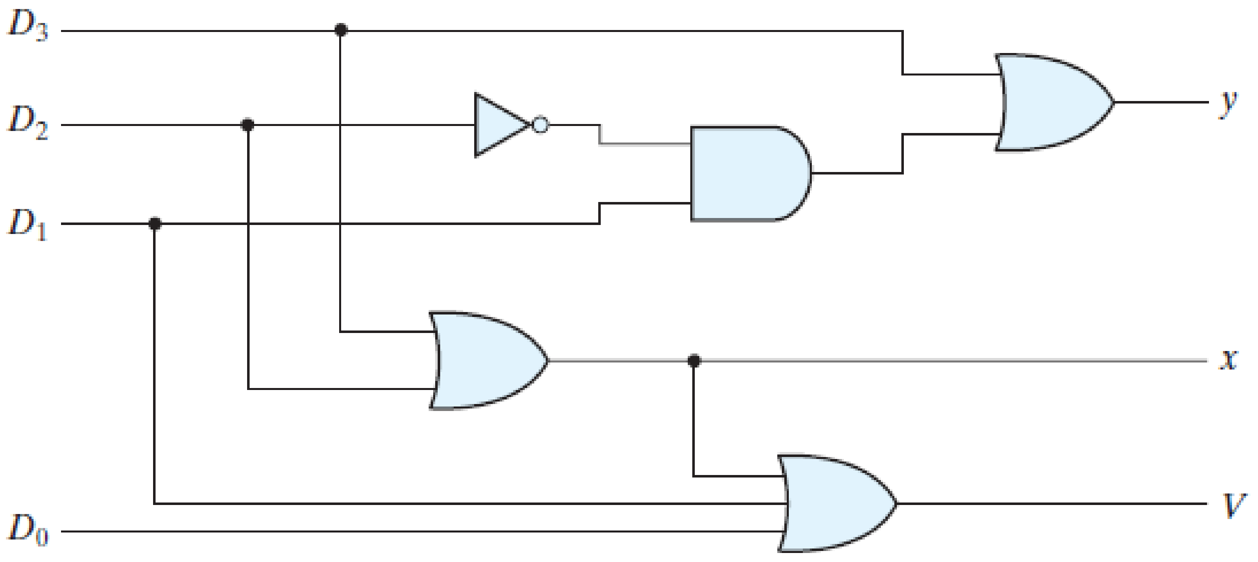
Κωδικοποιητής προτεραιότητας

Παράδειγμα - Σχεδίαση - Υλοποίηση λογικού διαγράμματος

οπότε, ισχύουν:

- $x = D_2 + D_3$
- $y = D_1D_2' + D_3$
- $V = D_0 + D_1 + D_2 + D_3$

και παράγεται το διπλανό λογικό διάγραμμα



λογικό διάγραμμα του κωδικοποιητή προτεραιότητας του παραδείγματος

Πολυπλέκτης

Σχεδίαση

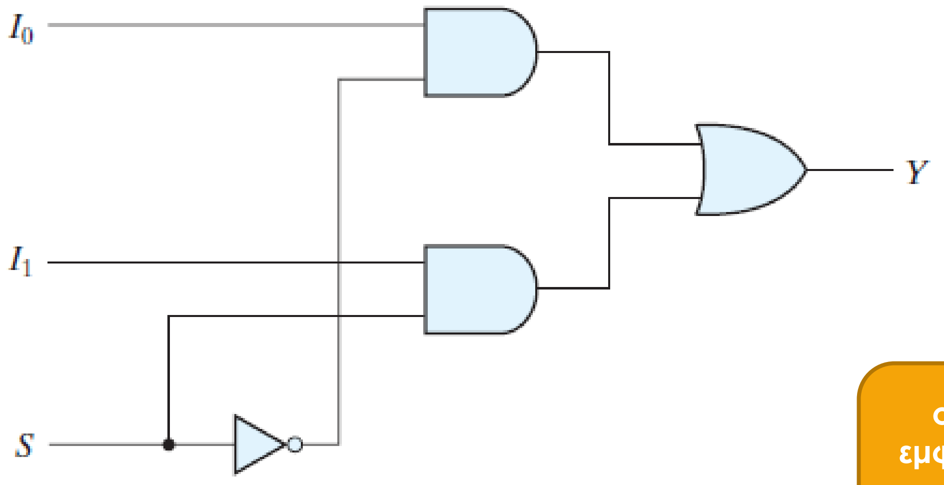
Πολυπλέκτης

- ένα συνδυαστικό κύκλωμα που
  - επιπλέγει δυαδική πληροφορία που έρχεται σε μία από τις πολλές γραμμές εισόδου και
  - την κατευθύνει σε μία γραμμή εξόδου

- η επιλογη μιας συγκεκριμένης γραμμής εισόδου ελέγχεται από ένα σύνολο από γραμμές επιλογής
  - υπάρχουν **2^n** γραμμές εισόδου και **n** γραμμές επιλογής

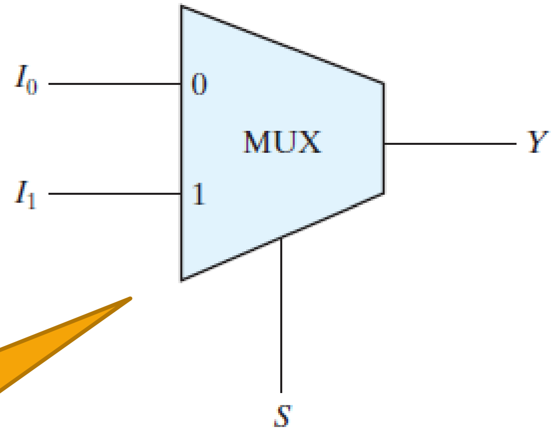
Πολυπλέκτης
2-σε-1

- ο πολυπλέκτης δύο γραμμών σε μία (ή απλούστερα **2**-σε-**1**) συνδέει λογικά μία από τις γραμμές εισόδου (I_0, I_1) σε μία μοναδική γραμμή εξόδου (**Y**)
  - όταν **S = 0** → ενεργοποιείται η πάνω πύλη **AND** και η τιμή του I_0 μεταφέρεται στην έξοδο
  - όταν **S = 1** → ενεργοποιείται η κάτω πύλη **AND** και η τιμή του I_1 μεταφέρεται στην έξοδο



λογικό διάγραμμα
πολυπλέκτη 2-σε-1

ο πολυπλέκτης συχνά εμφανίζεται με την αγγλική ονομασία (MUX) στα διαγράμματα



σχηματικό σύμβολο
πολυπλέκτη 2-σε-1

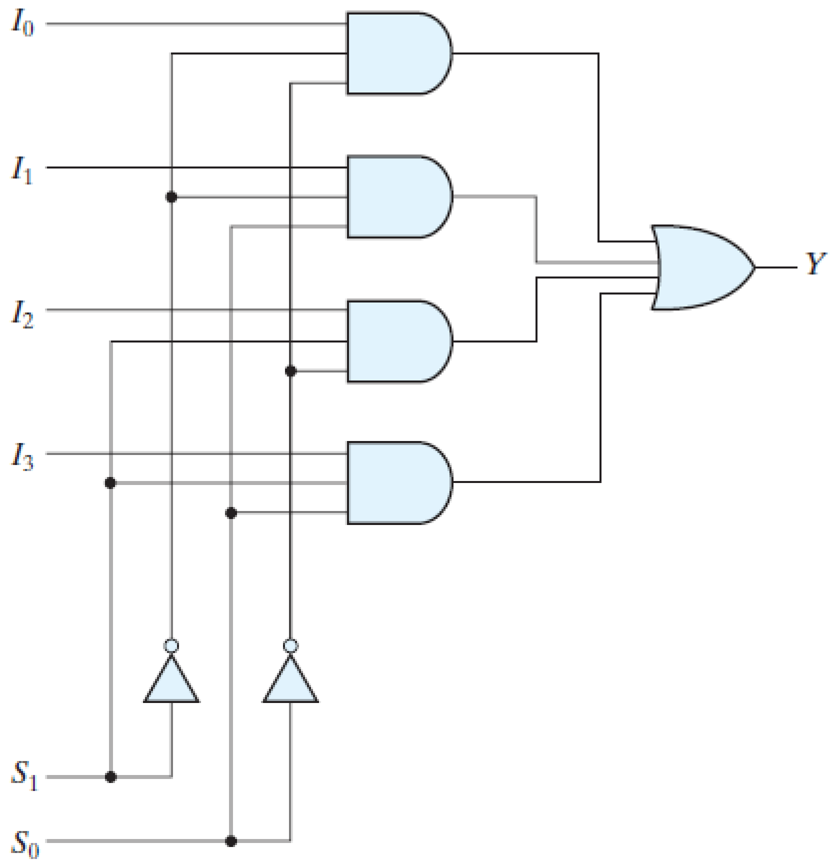
Πολυπλέκτης

4-σε-1

- ο πολυπλέκτης τεσσάρων γραμμών σε μία (ή απλούστερα 4-σε-1) συνδέει λογικά μία από τις γραμμές εισόδου (I_0, I_1, I_2 , I_3) σε μία μοναδική γραμμή εξόδου (Y)

S_1	S_2	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

πίνακας λειτουργίας πολυπλέκτη 4-σε-1

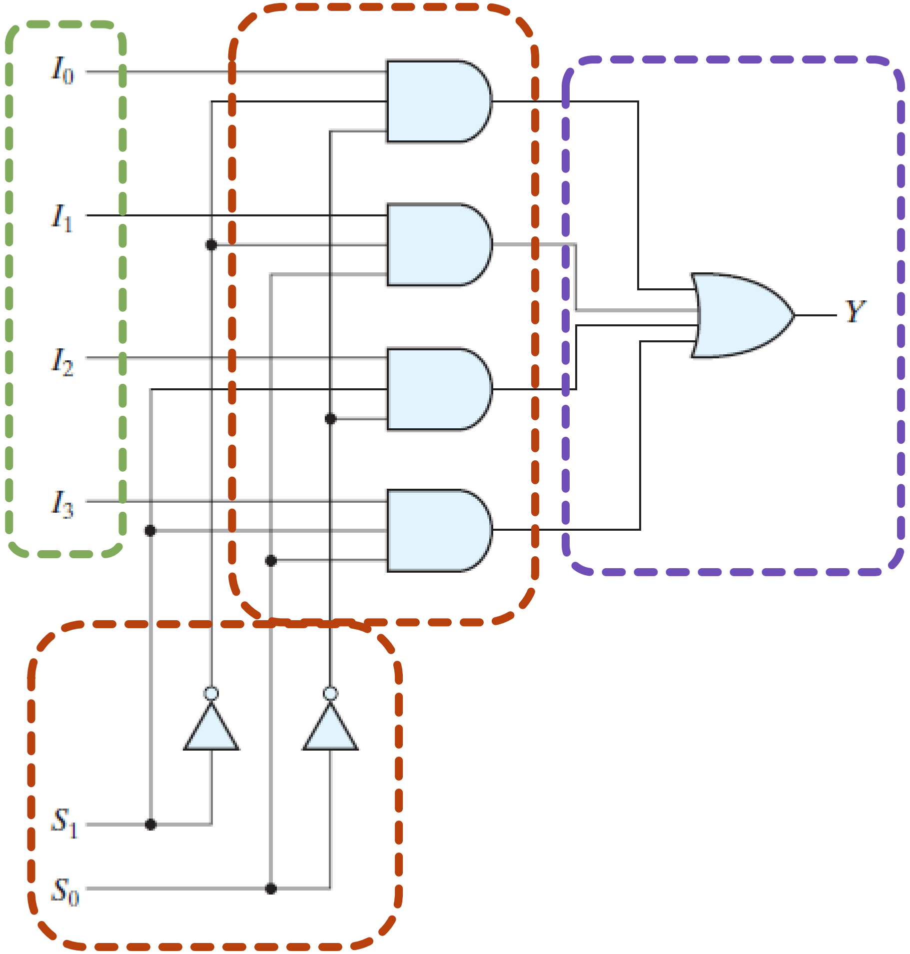


λογικό διάγραμμα πολυπλέκτη 4-σε-1

Πολυπλέκτης

4-σε-1 (ΙΙ)

- η δομή των αντιστροφέων και των πυλών **AND** του πολυπλέκτη είναι ένα κύκλωμα αποκωδικοποιητή
- γενικά, ένας πολυπλέκτης **2^n-σε-1** κατασκευάζεται εάν:
  1. προσθέσουμε **2^n** γραμμές εισόδου σε έναν απόκωδικοποιητή **n-σε-2^n**
     - συγκεκριμένα από μία σε κάθε πύλη **AND**
  2. και οδηγήσουμε τις εξόδους των πυλών **AND** σε μία και μοναδική πύλη **OR**



λογικό διάγραμμα πολυπλέκτη 4-σε-1

Πολυπλέκτης
Παρατηρήσεις

- το μέγεθος ενός πολυπλέκτης καθορίζεται από τον αριθμό των γραμμών εισόδου δεδομένων του (μέχρι **2^n**) και από τη μία γραμμή εξόδου
  - ο αριθμός των γραμμών επιλογής (**n**) υπονοείται, επειδή υπολογίζεται εύκολα από τον αριθμό των επιθυμητών γραμμών δεδομένων

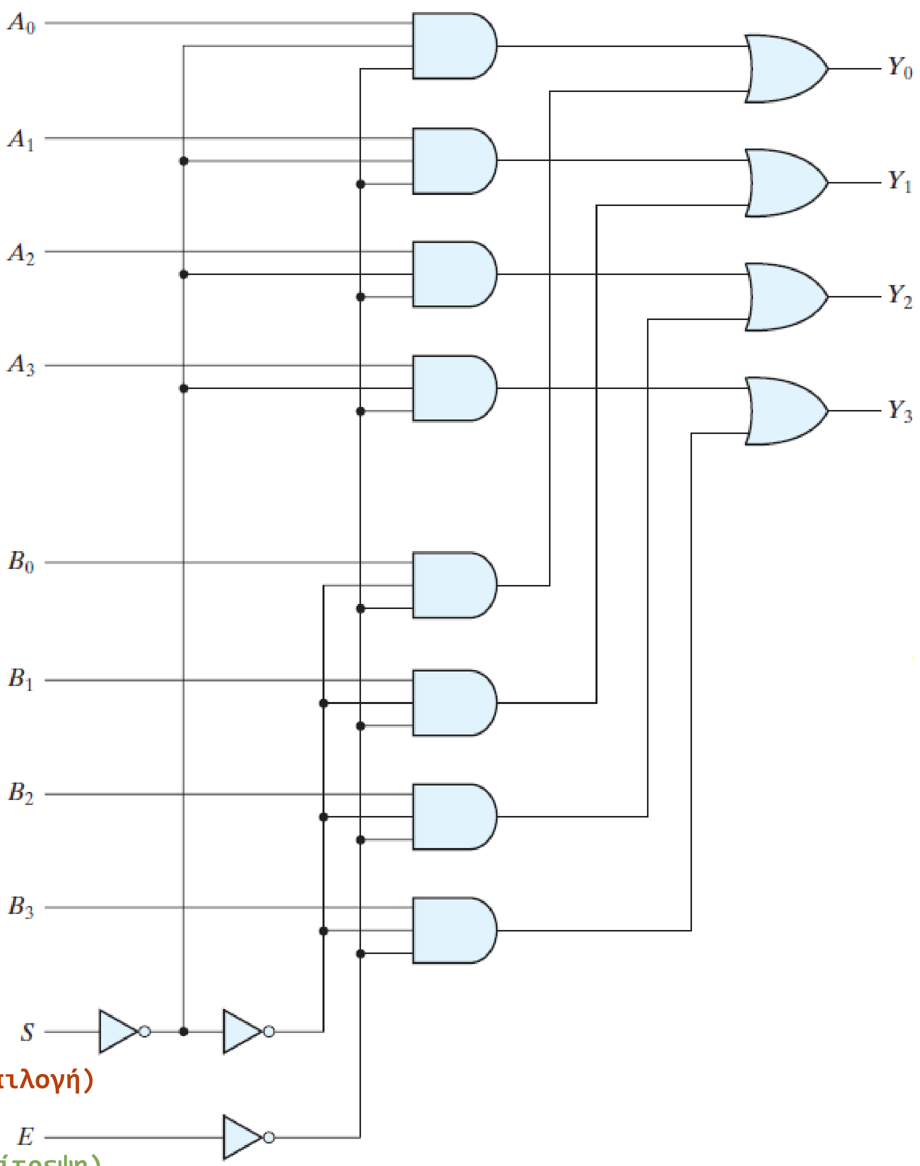
- ένας πολυπλέκτης μπορεί να έχει είσοδο επίτρεψης (enable) για τον έλεγχο της λειτουργίας του
  - όταν η είσοδος επίτρεψης είναι σε μη ενεργή κατάσταση → οι έξοδοι απενεργοποιούνται επίσης
  - όταν η είσοδος επίτρεψης είναι σε ενεργή κατάσταση → το κύκλωμα λειτουργεί ως κανονικός πολυπλέκτης

Πολυπλέκτης
Συνδυασμός

- οι πολυπλέκτες μπορούν να διασυνδεθούν μεταξύ τους → ώστε με τη χρήση κοινών εισόδων επιλογής να κατασκευαστούν μεγαλύτεροι πολυπλέκτες, με περισσότερες
  - γραμμές εισόδου ή/και
  - γραμμές επιλογής

E	S	έξοδος Y
1	X	όλα 0
0	0	επιλέγεται A
0	1	επιλέγεται B

πίνακας λειτουργίας τετραπλού πολυπλέκτη 2-σε-1 με είσοδο επίτρεψης (E)



λογικό διάγραμμα ενός τετραπλού πολυπλέκτη 2-σε-1 με είσοδο επίτρεψης (E)
(που υλοποιείται από τέσσερις πολυπλέκτες 2-σε-1)

Πολυπλέκτης

Υλοποίηση συναρτήσεων Boole - 1η μέθοδος

- σε έναν πολυπλέκτη το κύκλωμα που σχετίζεται με τις εισόδους επιλογής → παράγει τους ελαχιστόρους που αντιστοιχούν στις μεταβλητές επιλογής
  - είναι δηλαδή ένας αποκωδικοποιητής
- μπορούμε να διαλέξουμε οποιουσδήποτε από τους ελαχιστόρους αυτούς, αν τροφοδοτήσουμε τις εισόδους δεδομένων με τις κατάλληλες τιμές δεδομένων

☞ έτσι, μπορούμε να υλοποιήσουμε μία συνάρτηση Boole των **n** μεταβλητών, χρησιμοποιώντας έναν πολυπλέκτη που έχει **n** εισόδους επιλογής και 2^n εισόδους δεδομένων, ως εξής:

1. στις εισόδους επιλογής συνδέουμε τις μεταβλητές της συνάρτησης
2. σε κάθε είσοδο δεδομένων
   - που αντιστοιχεί σε ελαχιστόρο της συνάρτησης → θέτουμε **1**
   - που αντιστοιχεί σε ελαχιστόρο που δεν εμφανίζεται στη συνάρτηση → θέτουμε **0**

Πολυπλέκτης

Υλοποίηση συναρτήσεων Boole - 2η μέθοδος

- υλοποιούμε μία συνάρτηση Boole των **n** μεταβλητών, χρησιμοποιώντας έναν πολυπλέκτη που έχει **n-1** εισόδους επιλογής και 2^{n-1} εισόδους δεδομένων, ως εξής:
  1. σχεδιάζουμε τον πίνακα αληθείας της συνάρτησης
  2. στις εισόδους επιλογής συνδέουμε τις μεταβλητές της συνάρτησης, εκτός από την τελευταία (π.χ. **L**)
     - με τη σειρά που εμφανίζονται στον πίνακα αληθείας
  3. για κάθε συνδυασμό των μεταβλητών επιλογής:
     - υπολογίζουμε την επιθυμητή έξοδο του κυκλώματος ως αλγεβρική έκφραση της εναπομείνασας μεταβλητής (**L**)
     - η έκφραση αυτή θα είναι μία εκ των: **1**, **0**, **L**, **L'**
  4. σε κάθε είσοδο δεδομένων του πολυπλέκτη τοποθετούμε τις αντίστοιχες εκφράσεις που υπολογίσαμε

Πολυπλέκτης

Υλοποίηση συναρτήσεων Boole - 2η μέθοδος - 1ο Παράδειγμα

- υλοποιήστε το λογικό διάγραμμα της συνάρτησης: χρησιμοποιώντας έναν πολυπλέκτη
  - καθώς η συνάρτηση έχει τρεις μεταβλητές → $n = 3$
  - ο πολυπλέκτης πρέπει να έχει 2^{n-1} (= 2^2 = 4) εισόδους
  - άρα, θα χρησιμοποιήσουμε έναν πολυπλέκτη 4-σε-1
- διαδικασία υλοποίησης:
  1. σχεδιάζουμε τον πίνακα αληθείας της συνάρτησης
  2. στις εισόδους επιλογής συνδέουμε τις μεταβλητές της συνάρτησης (x και y), εκτός από την τελευταία (z)
  3. για κάθε συνδυασμό των μεταβλητών επιλογής υπολογίζουμε την επιθυμητή έξοδο του κυκλώματος ως αλγεβρική έκφραση της z

$$F(x,y,z) = \Sigma(1,2,6,7)$$

x	y	z	F	
0	0	0	0	F = z
0	0	1	1	
0	1	0	1	F = z'
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

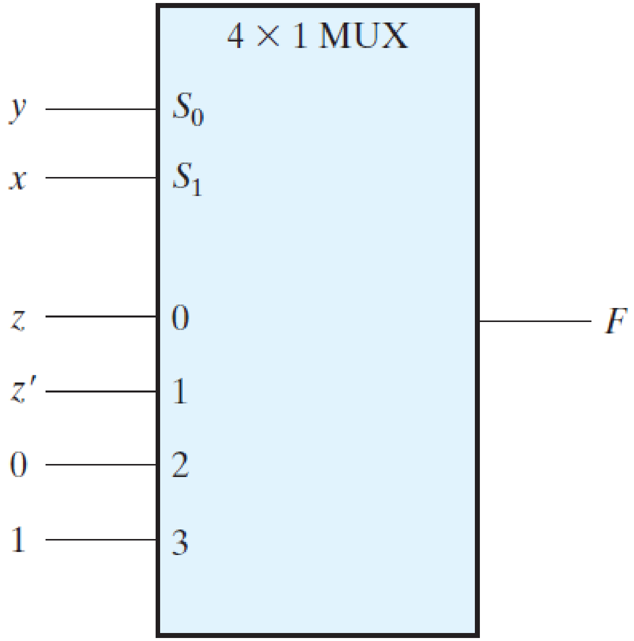
Πολυπλέκτης

Υλοποίηση συναρτήσεων Boole - <u>2η μέθοδος</u> - 1ο Παράδειγμα (II)

- υλοποιήστε το λογικό διάγραμμα της συνάρτησης: **F(x,y,z) = Σ(1,2,6,7)** χρησιμοποιώντας έναν πολυπλέκτη

2. στις εισόδους επιλογής συνδέουμε τις μεταβλητές της συνάρτησης (**x** και **y**), εκτός από την τελευταία (**z**)

4. σε κάθε είσοδο δεδομένων του πολυπλέκτη τοποθετούμε τις αντίστοιχες εκφράσεις που υπολογίσαμε



<table>
<tr><th>x</th><th>y</th><th>z</th><th colspan="2">F</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="2">F = z</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td rowspan="2">F = z'</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td rowspan="2">F = 0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td rowspan="2">F = 1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>

Πολυπλέκτης
Υλοποίηση συναρτήσεων Boole - 2η μέθοδος - 2ο Παράδειγμα

- υλοποιήστε το λογικό διάγραμμα της συνάρτησης:

$$F(A,B,C,D) = \Sigma(1,3,4,11,12,13,14,15)$$

  χρησιμοποιώντας έναν πολυπλέκτη
  - καθώς η συνάρτηση έχει τέσσερις μεταβλητές → **n = 4**
  - ο πολυπλέκτης πρέπει να έχει 2^{n-1} (= 2^3 = 8) εισόδους
  - άρα, θα χρησιμοποιήσουμε έναν πολυπλέκτη **8-σε-1**
- διαδικασία υλοποίησης:
  1. σχεδιάζουμε τον πίνακα αληθείας της συνάρτησης
  2. στις εισόδους επιλογής συνδέουμε τις μεταβλητές της συνάρτησης (**A**, **B** και **C**), εκτός από την τελευταία (**D**)
  3. για κάθε συνδυασμό των μεταβλητών επιλογής υπολογίζουμε την επιθυμητή έξοδο του κυκλώματος ως αλγεβρική έκφραση της **D**

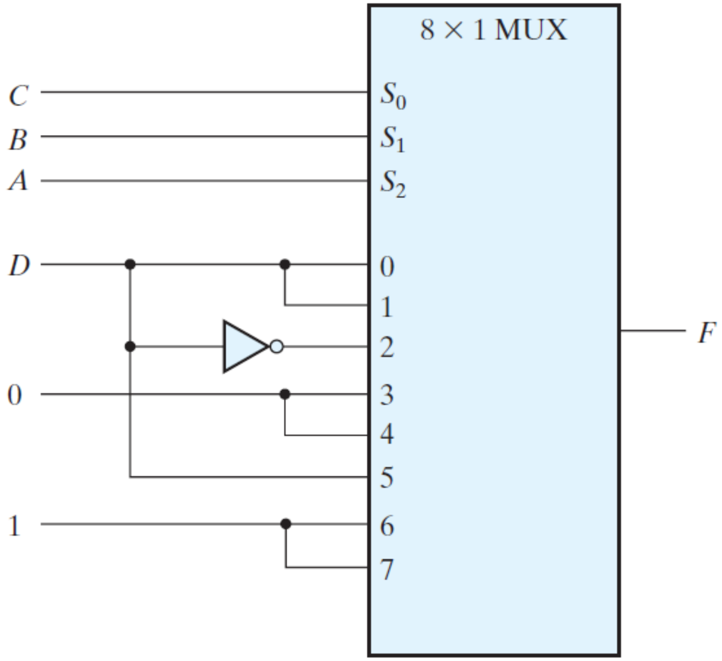
A	B	C	D	F	
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	F = D'
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = 1
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	

Πολυπλέκτης

Υλοποίηση συναρτήσεων Boole - 2η μέθοδος - 2ο Παράδειγμα (II)

- υλοποιήστε το λογικό διάγραμμα της συνάρτησης:
  F(A,B,C,D) = Σ(1,3,4,11,12,13,14,15)

2. στις εισόδους επιλογής συνδέουμε τις μεταβλητές της συνάρτησης (**A**, **B** και **C**), εκτός από την τελευταία (**D**)
4. σε κάθε είσοδο δεδομένων του πολυπλέκτη τοποθετούμε τις αντίστοιχες εκφράσεις που υπολογίσαμε



A	B	C	D	F	
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	F = D'
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = 1
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	

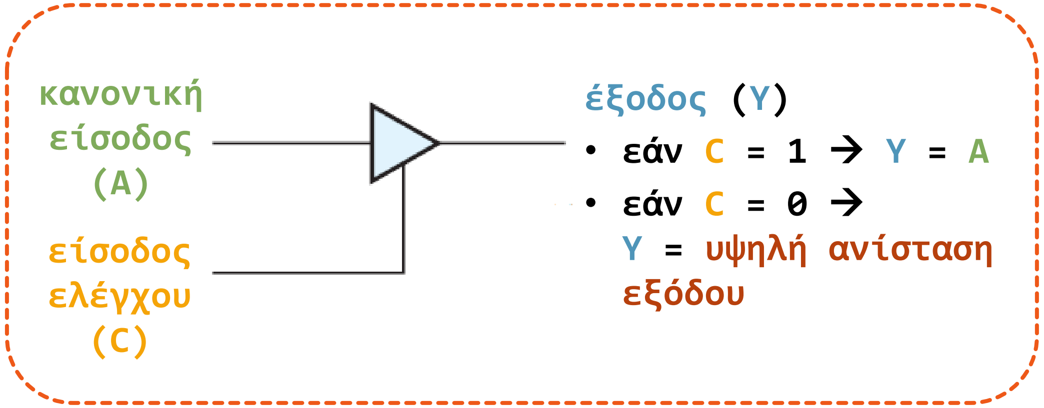
Πολυπλέκτης

Σχεδίαση με πύλες τριών καταστάσεων

Πύλες τριών καταστάσεων (ή τρισταθείς)

- έχουν τις δύο γνωστές καταστάσεις: λογικό **0** και λογικό **1**
- έχουν μία επιπλέον κατάσταση υψηλής αντίστασης εξόδου
  1. η έξοδος του λογικού κυκλκώματος συμπεριφέρεται σαν ένα ανοικτό κύκλωμα
     - μπορεί να αποσυνδεθεί από το επόμενο κύκλωμα
  2. το τρισταθές κύκλωμα δε συμμετέχει στον καθορισμό της λογικής τιμής της εξόδου
     - (ή όπως συνηθίζεται να λέγεται) δεν έχει καμία λογική σημασία
  3. το κύκλωμα που συνδέεται στην έξοδο της τρισταθούς πύλης δεν επηρεάζεται καθόλου από τις εισόδους της πύλης
- μπορούν να χρησιμοποιηθούν για να υλοποιήσουν οποιαδήποτε συμβατική λογική πράξη (π.χ. **AND** ή **NAND**)
- χρησιμοποιούνται στην πράξη για την υλοποίηση της πύλης του τρισταθούς απομονωτή

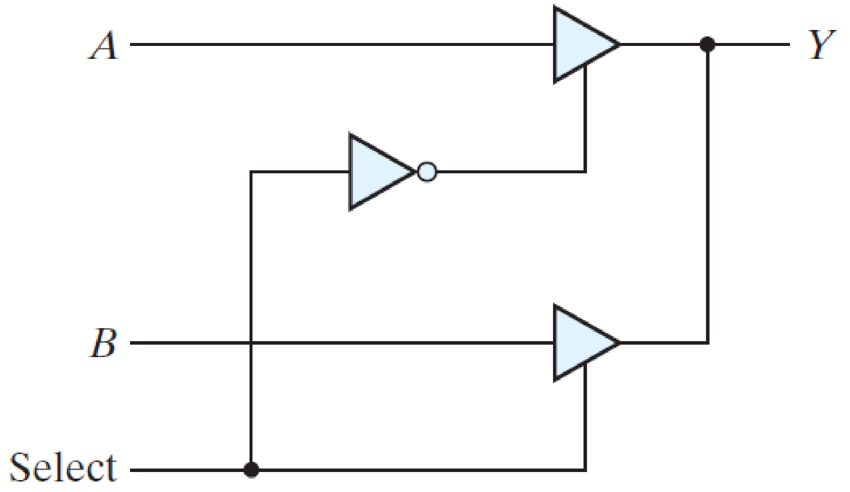
Τρισταθής απομονωτής



- έχει μία κανονική λογική είσοδο (**A**)
- έχει μία έξοδο (**Y**)
- επιπρόσθετα έχει μία **είσοδο ελέγχου** (**C**) → καθορίζει την κατάσταση της εξόδου
  - όταν είναι **C = 1** → η έξοδος ενεργοποιείται και η πύλη συμπεριφέρεται ως συμβατικός απομονωτής
    - η έξοδος είναι λογικά ίδια με την κανονική είσοδο
  - όταν είναι **C = 0** → η έξοδος απενεργοποιείται και η πύλη μεταβαίνει σε **κατάσταση υψηλής αντίστασης εξόδου**, ανεξάρτητα από την τιμή της κανονικής εισόδου
    - ✍ λόγω αυτού του χαρακτηριστικού, οι έξοδοι μεγάλου αριθμού τρισταθών πυλών μπορούν να **βραχυκυκλωθούν** → ώστε να σχηματιστεί ένας **κοινός** κόμβος εξόδου
      - 👍 χωρίς να υπάρχει κίνδυνος από υπερφόρτωση των ηλεκτρονικών σταδίων εξόδου των αντίστοιχων πυλών
      - ✍ θα υπήρχε κίνδυνος υπερφόρτωσης εάν βραχυκυκλώνονταν έξοδοι συμβατικών πυλών

Πολυπλέκτης
2-σε-1 - Υλοποίηση με τρισταθής απομονωτές

- χρησιμοποιούνται δύο τρισταθείς απομονωτές και ένας αντιστροφέας
- οι έξοδοι των τρισταθών απομονωτών βραχυκυκλώνονται → ώστε να σχηματιστεί ένας κόμβος εξόδου
  - δηλαδή μία γραμμή εξόδου (**Y**)
- όταν η **είσοδος επιλογής** (**Select**) είναι **0**:
  - ενεργοποιείται ο πάνω απομονωτής και απενεργοποιείται ο κάτω απομονωτής
  - η έξοδος ισούται με **A**
- όταν η **είσοδος επιλογής** (**Select**) είναι **1**:
  - ενεργοποιείται ο κάτω απομονωτής και απενεργοποιείται ο πάνω απομονωτής
  - η έξοδος ισούται με **B**

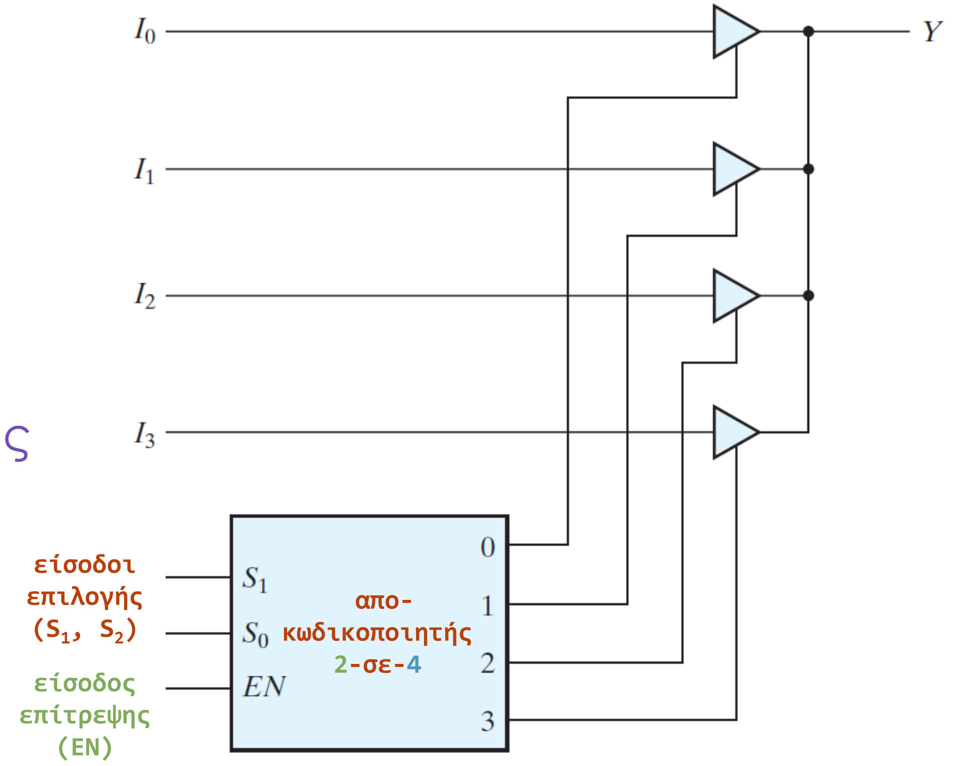


λογικό διάγραμμα πολυπλέκτη 2-σε-1
(που υλοποιείται με χρήση δύο τρισταθών απομονωτών και ενός αντιστροφέα)

Πολυπλέκτης

4-σε-1 - Υλοποίηση με τρισταθής απομονωτές

- χρησιμοποιούνται τέσσερις τρισταθείς απομονωτές και ένας αποκωδικοποιητής **2-σε-4**
- οι έξοδοι των τρισταθών απομονωτών βραχυκυκλώνονται → ώστε να σχηματιστεί ένας κόμβος εξόδου, δηλαδή μία γραμμή εξόδου (**Y**)
- σε κάθε χρονική στιγμή μόνο ένας από τους τρισταθείς απομονωτές επιστρέπεται να είναι ενεργός
  - όταν η **είσοδος επίτρεψης** (**EN**) είναι **0**:
    - οι τέσσερις έξοδοι του αποκωδικοποιητή είναι 0
    - και οι τεσσερις τρισταθείς απομονωτές είναι ανενεργοί
    - η γραμμή εξόδου του κυκλώματος βρίσκεται σε κατάσταση υψηλής αντίστασης → ακαθόριστη τιμή
  - όταν η **είσοδος επίτρεψης** (**EN**) είναι **1**:
    - ενεργοποιείται ένας μόνο από τους τρισταθείς απομονωτές, ανάλογα με το συνδυασμό τιμών των εισόδων επιλογής του αποκωδικοποιητή



λογικό διάγραμμα πολυπλέκτη 4-σε-1
(που υλοποιείται με χρήση τεσσάρων τρισταθών απομονωτών και ενός αποκωδικοποιητή 2-σε-4)

Σύνοψη

- Συνδυαστικά κυκλώματα
  - Διαδικασία ανάλυσης
  - Διαδικασία σχεδίασης
- Δυαδικός αθροιστής-αφαιρέτης
  - Ημιαθροιστής & Πλήρης αθροιστής
  - Δυαδικός αθροιστής ριπής κρατούμενου & πρόβλεψης κρατούμενου
    - γεννήτρια πρόβλεψης κρατούμενου
  - Υπερχείλιση - Εντοπισμός
- Δεκαδικός αθροιστής (αθροιστής BCD)
- Δυαδικός πολλαπλασιαστής & Συγκριτής μεγέθους
- Αποκωδικοποιητής & Αποπλέκτης
- Κωδικοποιητής
- Πολυπλέκτης