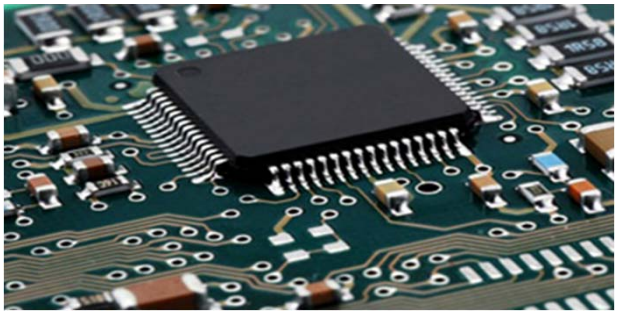
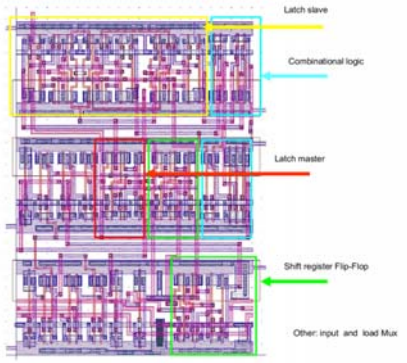
SCHOOL OF INFORMATICS





Ψηφιακή Σχεδίαση

Διάλεξη 4 – Βασικά Συνδυαστικά Κυκλώματα

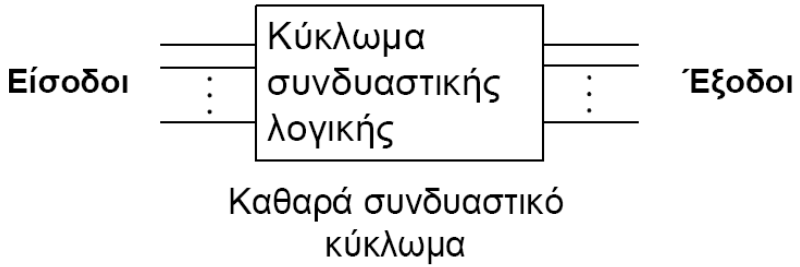
Γεώργιος Κεραμίδας, Επίκουρος Καθηγητής

2ο Εξάμηνο, Τμήμα Πληροφορικής

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

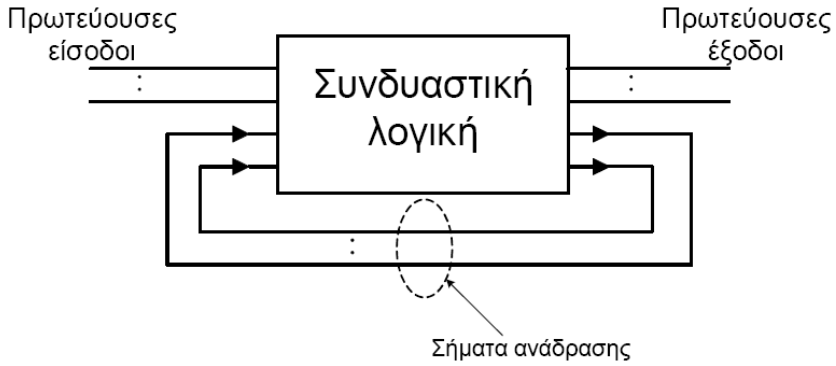
Γενικές κατηγορίες κυκλωμάτων

- **Συνδυαστικά**



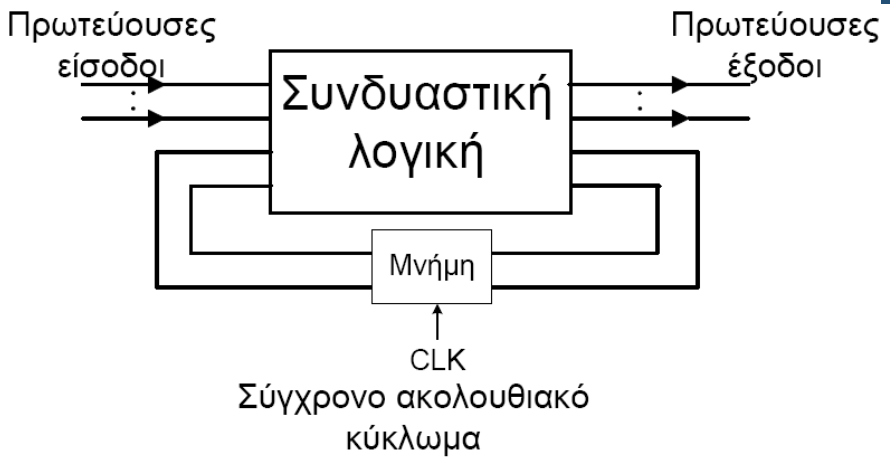
Καθαρά συνδυαστικό κύκλωμα

- **Ακολουθιακά**



Ασύγχρονο ακολουθιακό κύκλωμα

Ακολουθιακά



- **Τα ακολουθιακά κυκλώματα «θυμούνται» μέσω της σύνδεσης της ανάδρασης. Δύο είναι οι κύριες κατηγορίες των ακολουθιακών κυκλωμάτων:**
  - **Ασύγχρονα: Αλλάζουν κατάσταση σύμφωνα με τις αλλαγές των εισόδων τους. Απαιτούνται ειδικές τεχνικές σχεδιασμού.**
  - **Σύγχρονα: Τα σήματα ανάδρασης διακόπτονται από καταχωρητές που σκανδαλίζονται από παλμούς ρολογιού. Συνεπώς η κατάστασή του κυκλώματος αλλάζει σύμφωνα με τους παλμούς του ρολογιού. Η κατάσταση του κυκλώματος ορίζεται από το περιεχόμενο των στοιχείων της μνήμης.**

Συνδυαστικά κυκλώματα που θα μας απασχολήσουν

- **Αριθμητικά κυκλώματα :**
  - **Ημιαθροιστής**
  - **Πλήρης αθροιστής**
  - **Παράλληλος Αθροιστής / Αφαιρέτης**
    - **Με διάδοση κρατουμένου**
    - **Με πρόβλεψη κρατουμένου**
  - **Συγκριτής**
  - **Πολλαπλασιαστής**

- **Κωδικοποίησης / Αποκωδικοποίησης / Πολυπλεξίας / Αποπλεξίας**
  - **Κωδικοποιητής**
  - **Κωδικοποιητής προτεραιότητας**
  - **Αποκωδικοποιητής / Αποπλέκτης**
  - **Πολυπλέκτης**

Ημι-Αθροιστής (Half Adder)

1.Καθορισμός προβλήματος: **κύκλωμα που να προσθέτει δύο δυαδικά ψηφία.**

2.Πλήθος εισόδων/εξόδων: **2 είσοδοι – 2 έξοδοι.**

3.Ονομασία εισόδων/εξόδων: **έστω x, y οι δύο είσοδοι (προσθετέοι) και C (κρατούμενο), S (άθροισμα) οι δύο έξοδοι.**

4.Πίνακας αλήθειας:

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

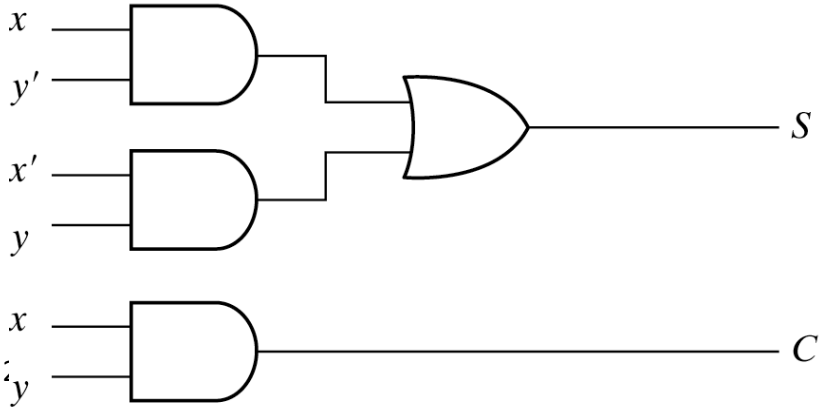
Ημι-Αθροιστής

(α)

$S = x'y + xy'$

$C = xy$

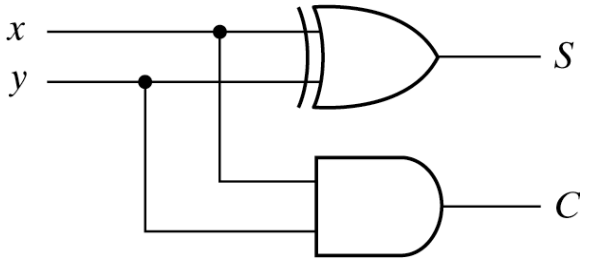
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



(β)

$S = x \oplus y$

$C = xy$



Πλήρης Αθροιστής (Full Adder)

1.Καθορισμός προβλήματος: **κύκλωμα που να προσθέτει τρία δυαδικά ψηφία.**

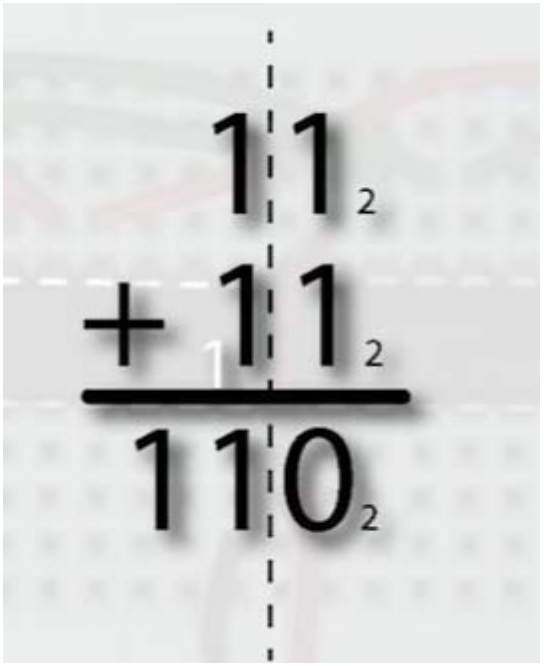
2.Πλήθος εισόδων/εξόδων: **3 είσοδοι – 2 έξοδοι.**

3.Ονομασία εισόδων/εξόδων: **έστω x, y οι δύο είσοδοι (προσθετέοι), z το κρατούμενο της προηγούμενης βαθμίδας και C (κρατούμενο), S (άθροισμα) οι δύο έξοδοι.**

4.Πίνακας αλήθειας:

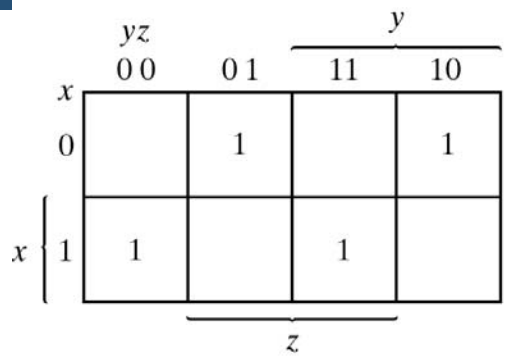
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

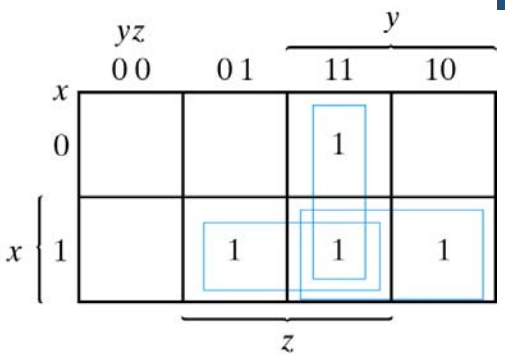
Πλήρης Αθροιστής (Full Adder)



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

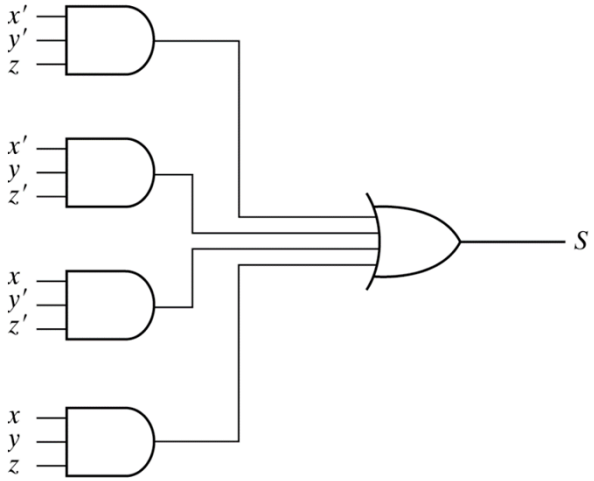
Πλήρης-Αθροιστής

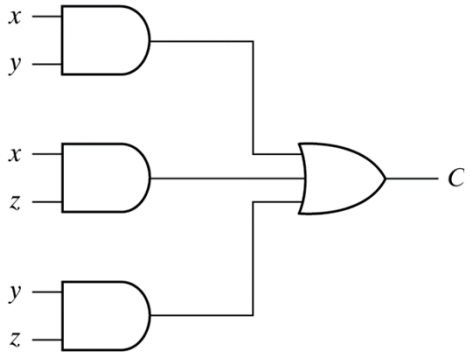




S = x′y′z+x′yz′+xy′z′+xyz

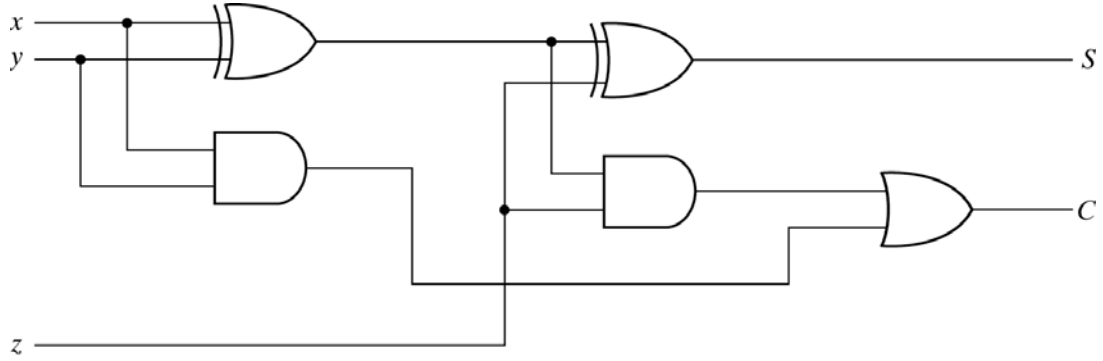
C = xy + yz + xz





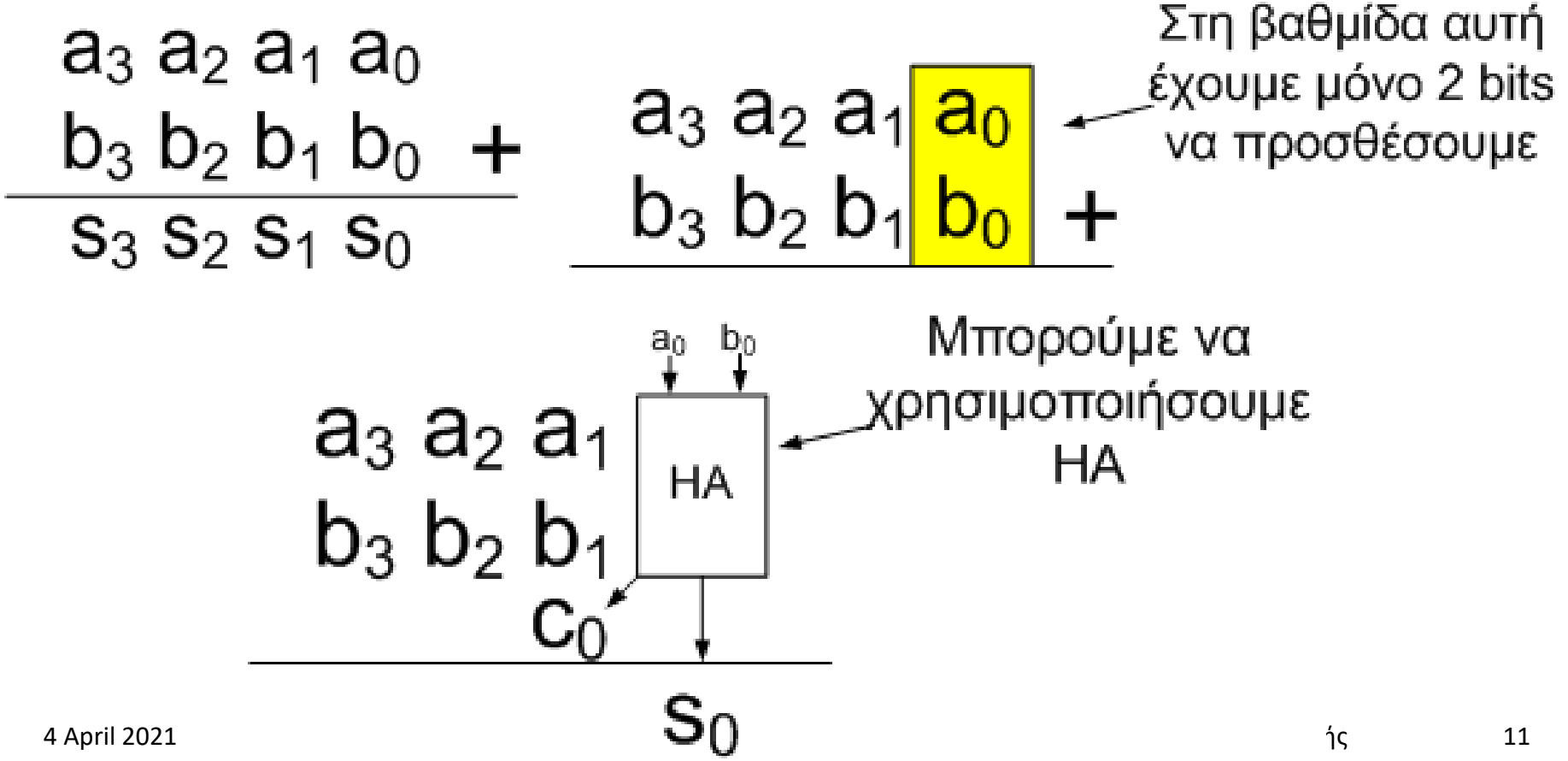
Πλήρης-Αθροιστής

- Είναι S = (x ⊕ y) ⊕ z
- Επίσης ισχύει
- C = xy + yz + xz = xy + z (x + y) = xy + z (x'y + xy' + xy) =
  xy + zxy + z (x'y + xy') = xy + z (x ⊕ y)

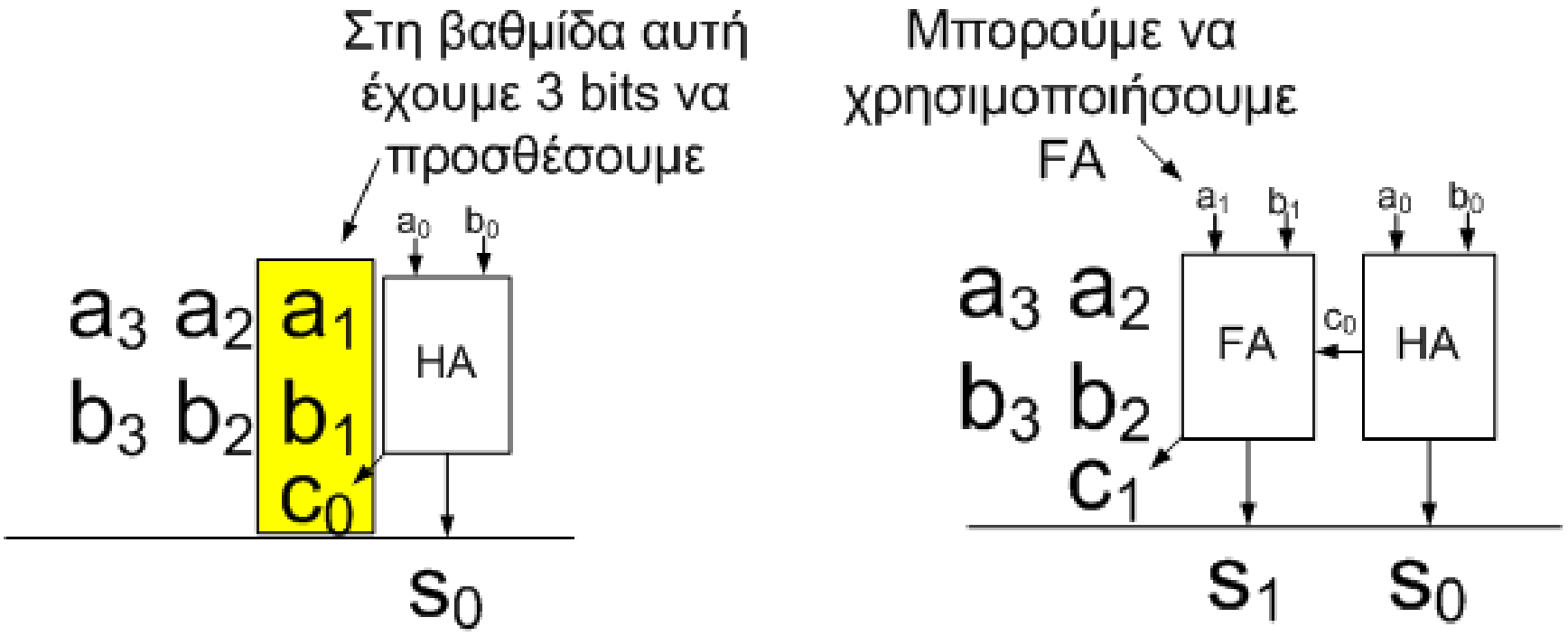


- Μπορώ συνεπώς να κατασκευάσω το πλήρη αθροιστή από 2 ημιαθροιστές και μια πύλη OR

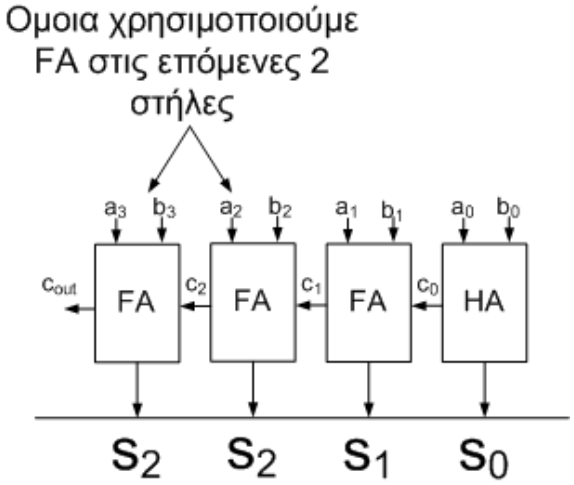
Πρόσθεση στο δυαδικό με διάδοση κρατουμένου (1/4)

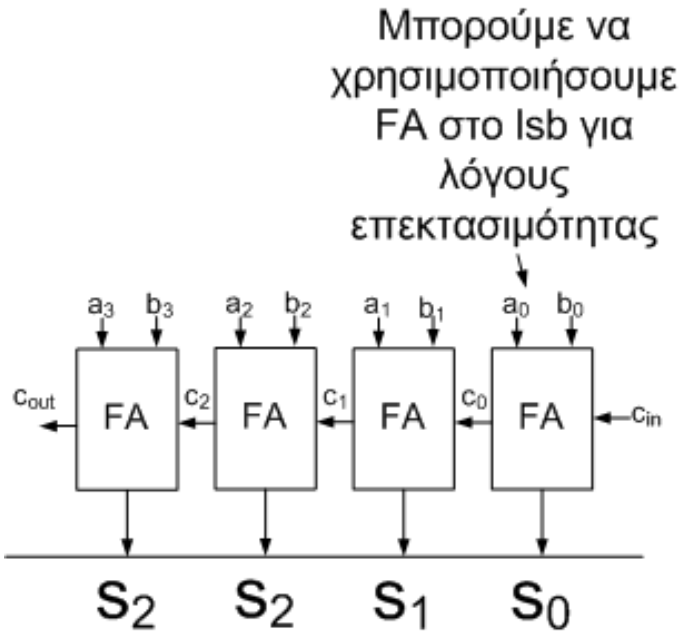


Πρόσθεση στο δυαδικό με διάδοση κρατουμένου (2/4)



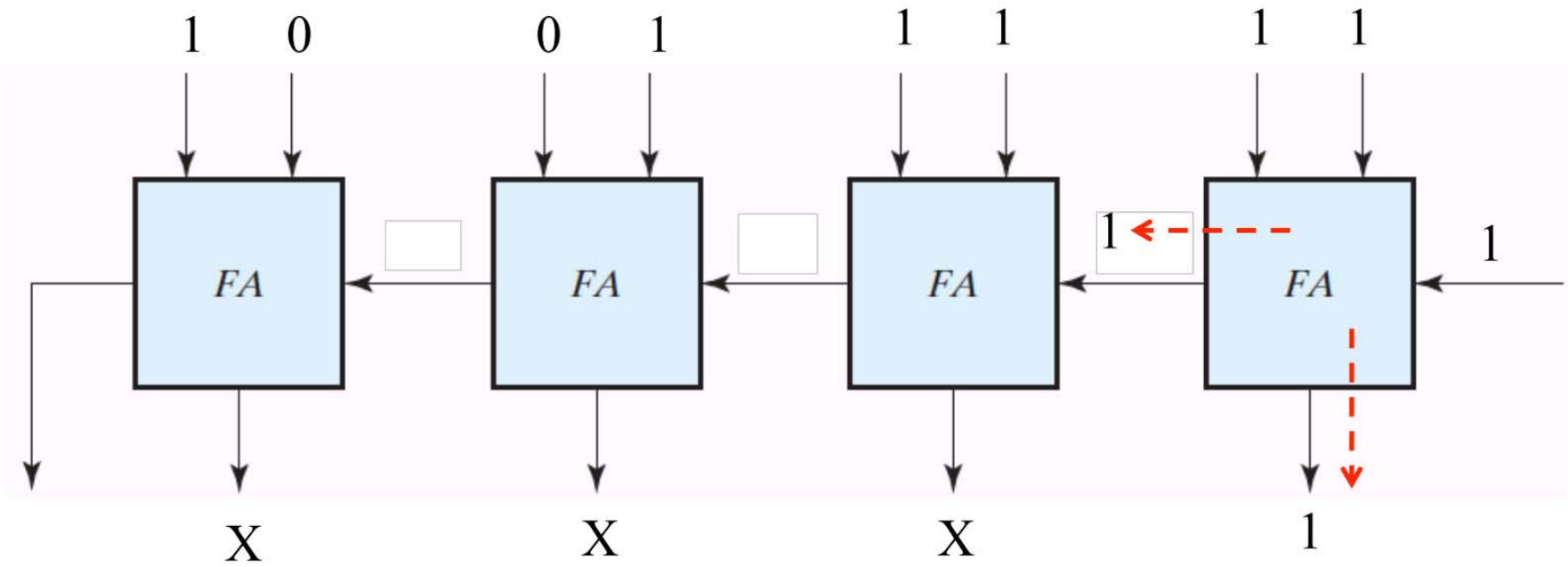
Πρόσθεση στο δυαδικό με διάδοση κρατουμένου (3/4)



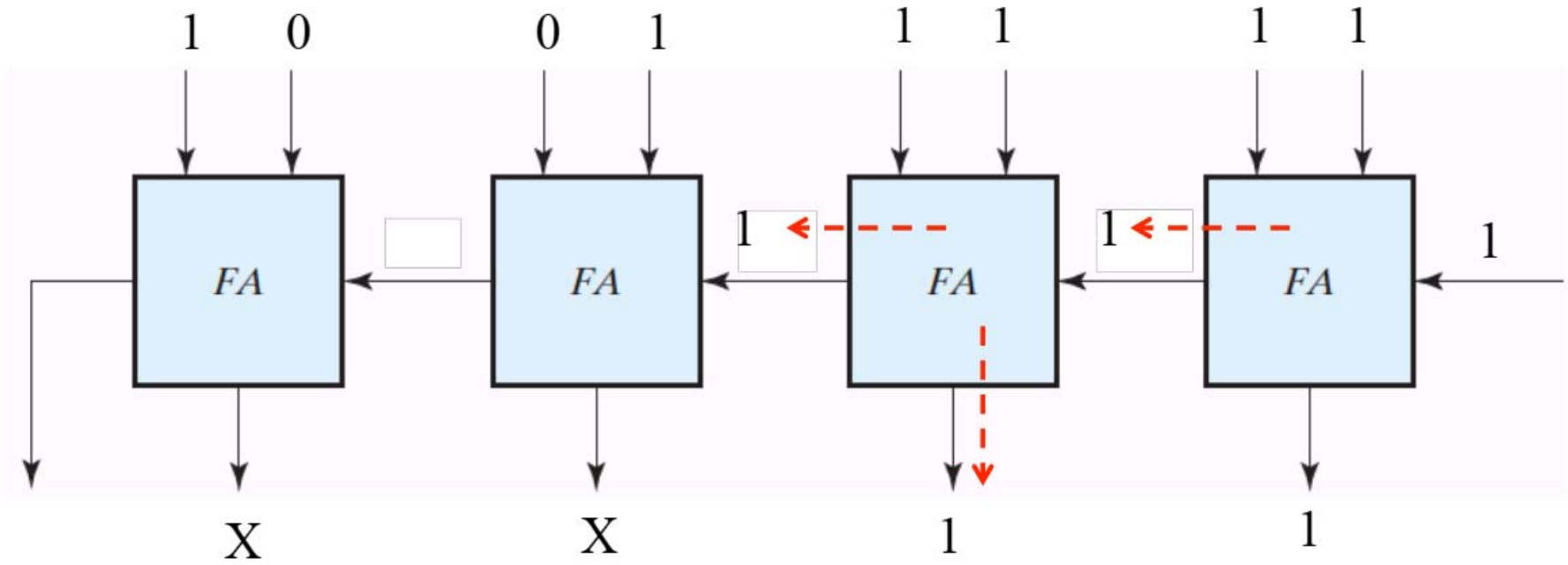


- **Φτιάξαμε συνεπώς ένα κύκλωμα που μπορεί να προσθέτει έντελα των 4 δυαδικών ψηφίων.**
- **Το κύκλωμα αυτό θα ονομάζεται παράλληλος αθροιστής των 4 δυαδικών ψηφίων.**
- **Με την ίδια λογική μπορούμε να φτιάξουμε παράλληλους αθροιστές των κ δυαδικών ψηφίων**
- **Χρησιμοποιώντας δηλαδή MSI, φτιάχνουμε LSI (για ικανοποιητικά μεγάλο).**

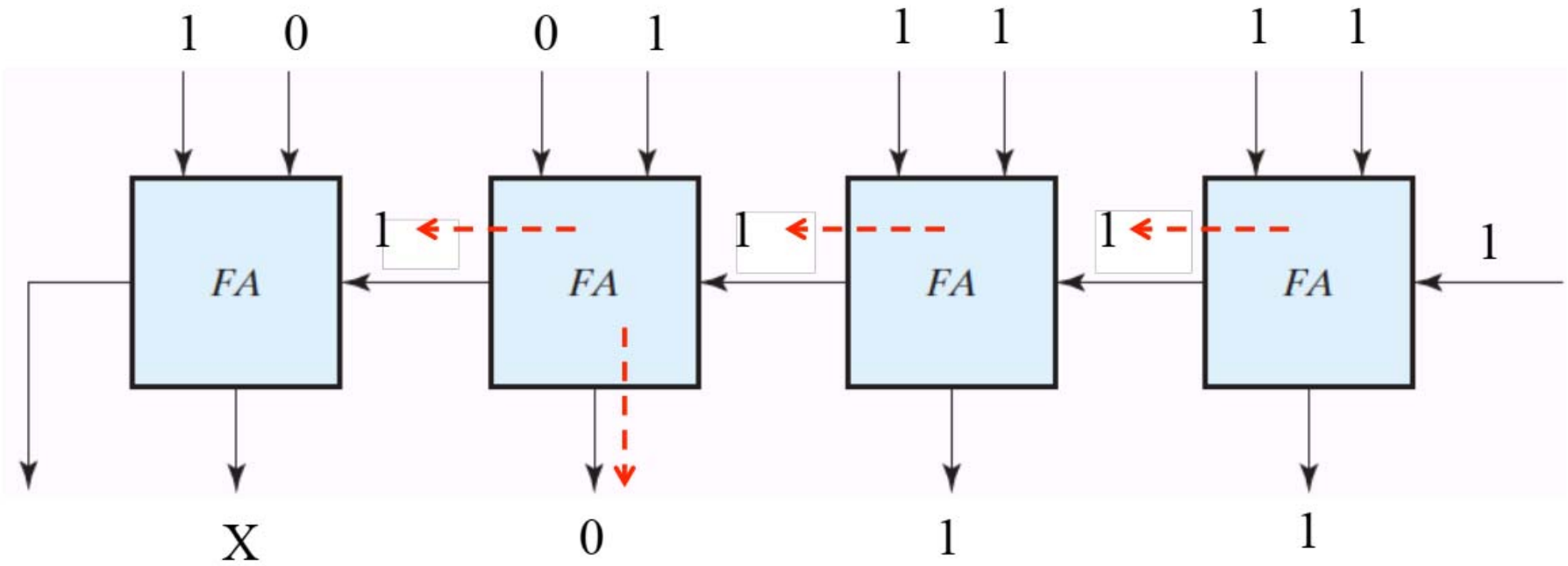
Παράδειγμα



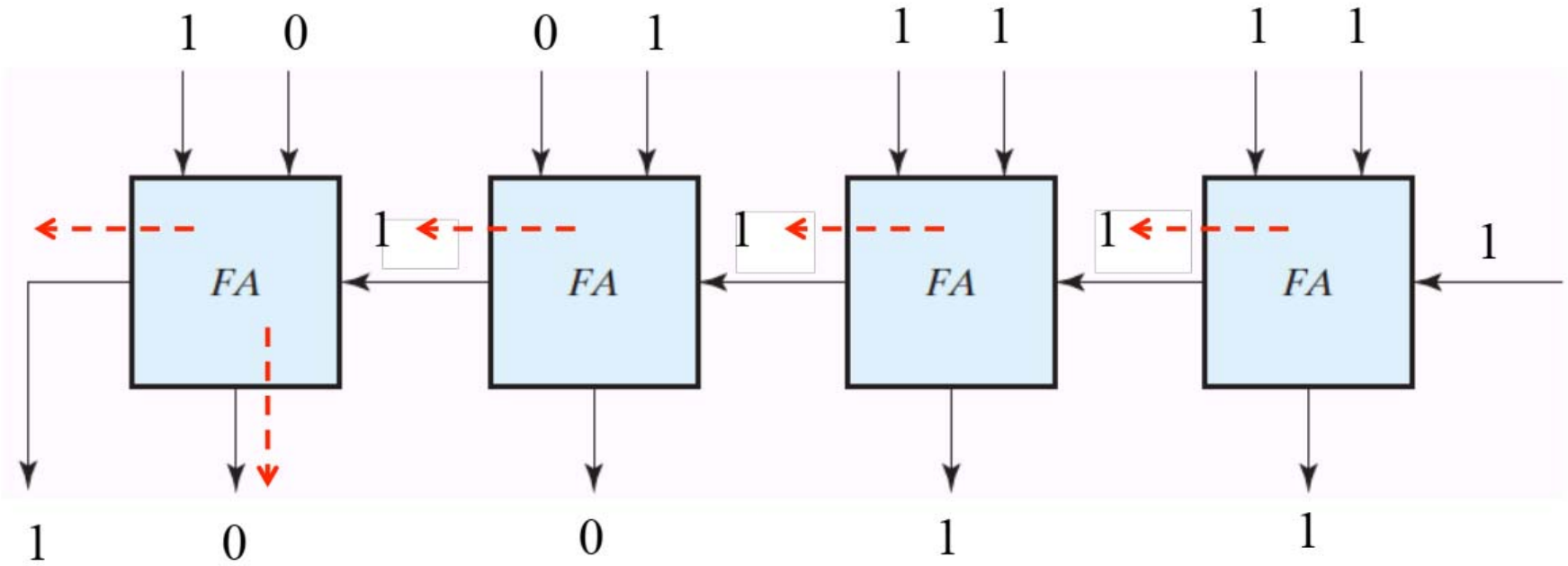
Παράδειγμα



Παράδειγμα



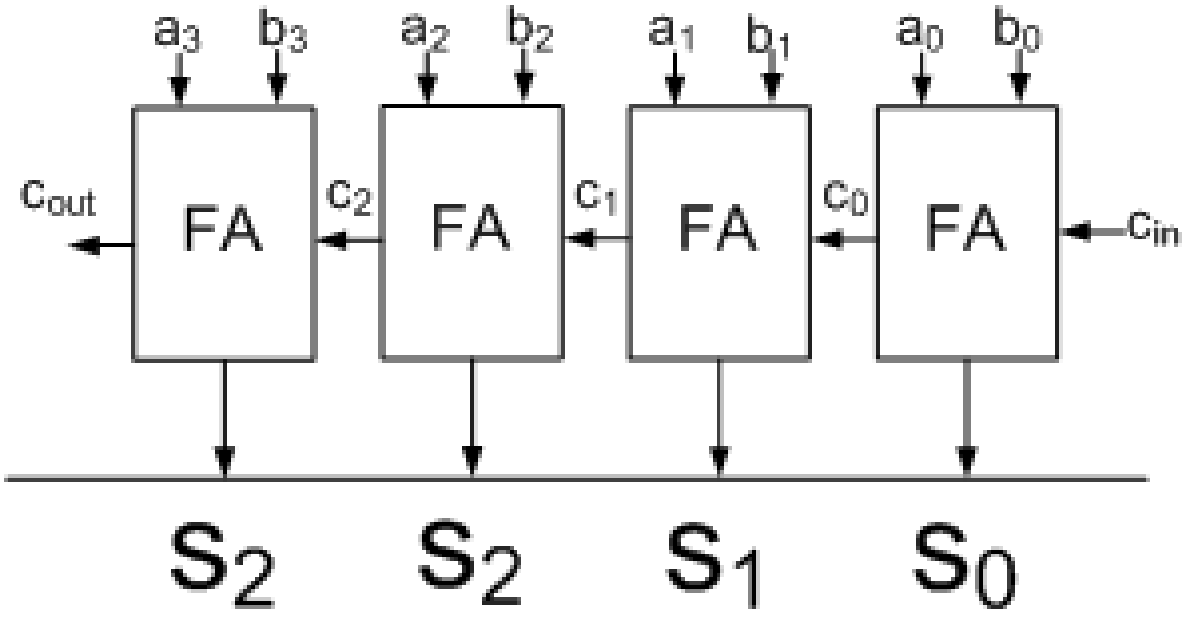
Παράδειγμα

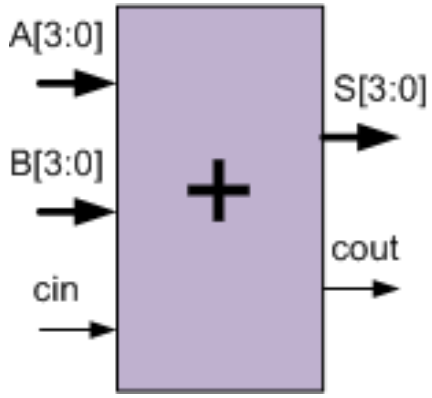


Πρόσθεση στο δυαδικό με διάδοση κρατουμένου (4/4)

- Στην ουσία στις προηγούμενες διαφάνειες δε περιγράψαμε ένα κύκλωμα, αλλά μια οικογένεια κυκλωμάτων.
- Όλα τα μέλη της οικογένειας ακολουθούν το ίδιο μοτίβο :
  - Αποτελούνται από πλήρεις αθροιστές.
  - Κάθε πλήρης αθροιστής προσθέτει ένα ζεύγος από αντίστοιχα δυαδικά ψηφία των εντέλων και το κρατούμενο της προηγούμενης βαθμίδας, παρέχει 1 δυαδικό ψηφίο του αποτελέσματος και κρατούμενο προς τον επόμενο αθροιστή.
- Κάθε τέτοιο καλώς οριζόμενο μοτίβο, στην επιστήμη μας είναι μια αρχιτεκτονική
- Περιγράψαμε συνεπώς παράλληλους αθροιστές με αρχιτεκτονική διάδοσης κρατουμένου (κάθε βαθμίδα διαδίδει τυχόν κρατούμενο στην επόμενη) → n-bit ripple carry adder (αθροιστής ριπής)

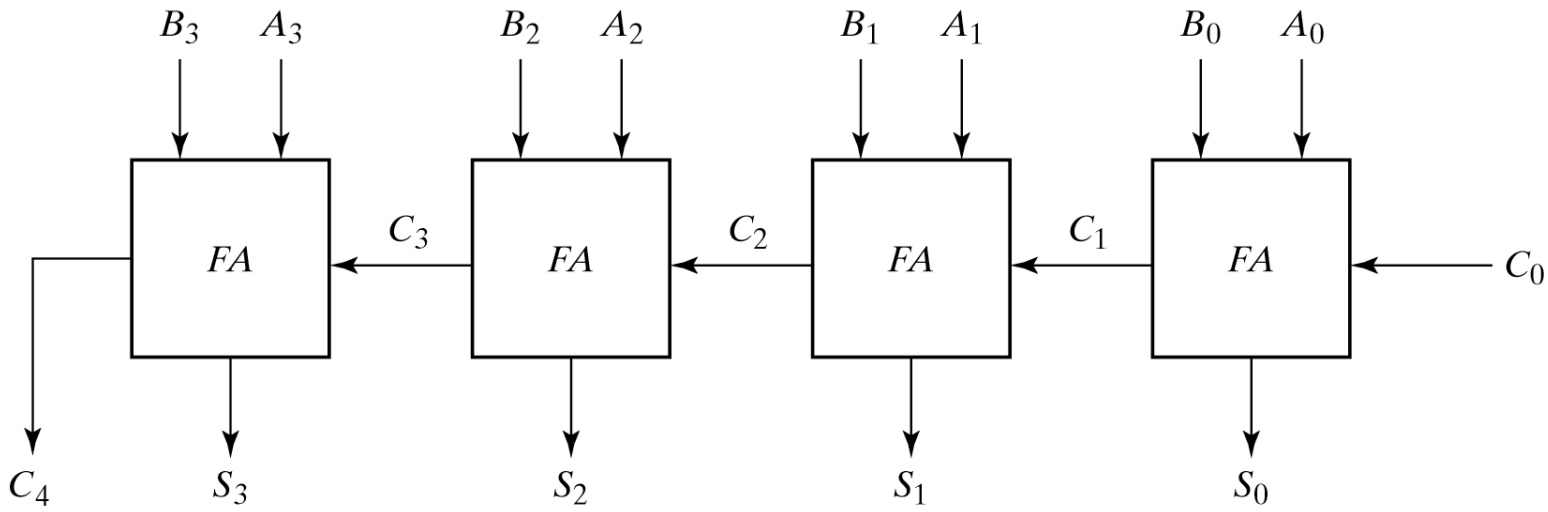
Παράλληλος δυαδικός αθροιστής με διάδοση κρατουμένου





- **Υλοποίηση με συναρτήσεις: Πίνακας αλήθειας με 9 εισόδους και 2^9=512 καταστάσεις.**

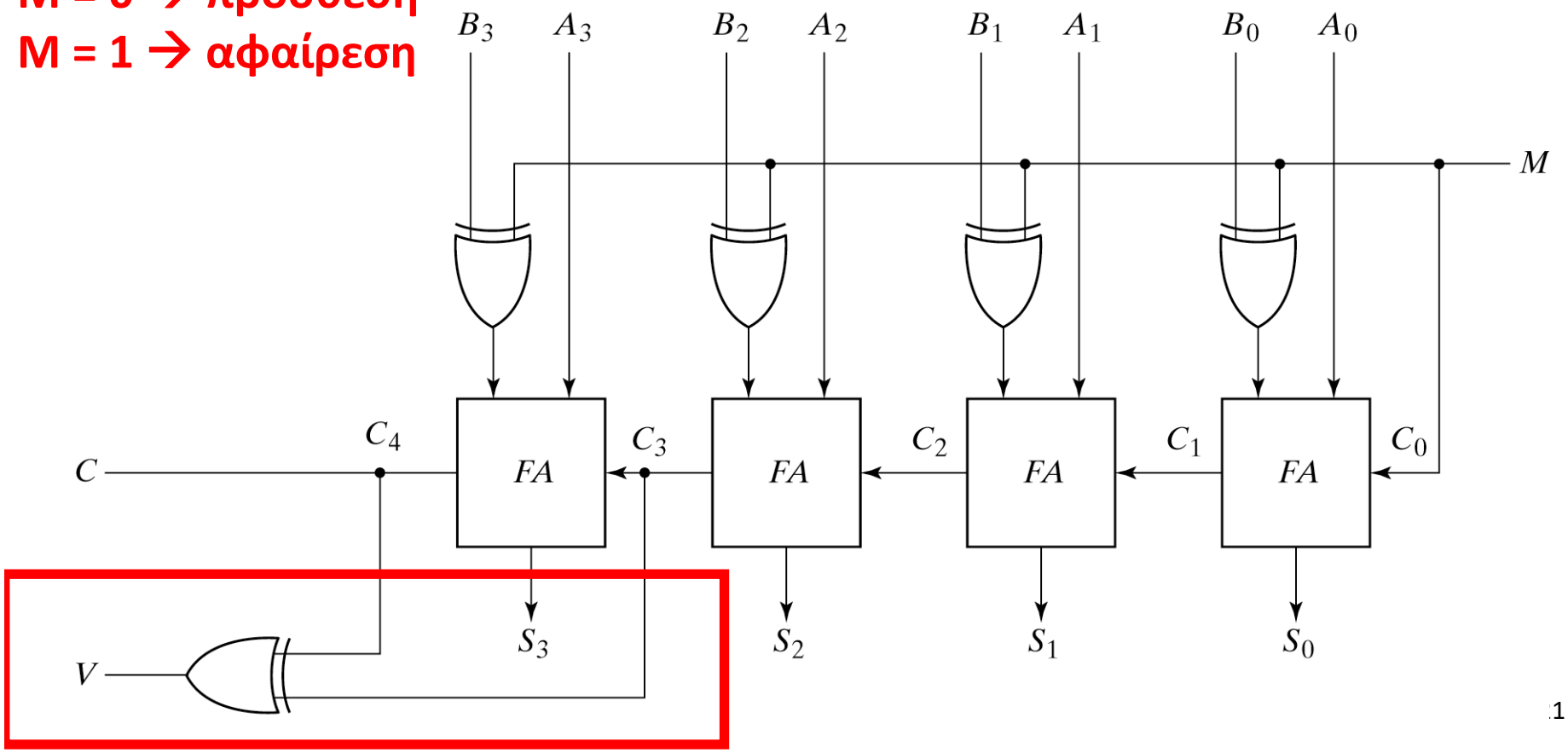
Παράλληλος Δυαδικός Αφαιρέτης



- **A – B = A + συμπλήρωμα(B) = A + B΄ + 1**

Παράλληλος Αθροιστής/Αφαιρέτης

M = 0 → πρόσθεση
M = 1 → αφαίρεση



Το πρόβλημα της Υπερχείλισης στο Σύστημα Signed-2's complement

- **Να θυμάστε ότι το MSB είναι το πρόσημο. Αλλά προστίθεται και το πρόσημο! Το carry "δείχνει" υπερχείλιση μόνο όταν προσθέτω μη-προσημασμένους αριθμούς**

Θετικός – Αρνητικός →
δεν μπορεί να βγάλει overflow

	1	X	X	X
+	0	X	X	X
	1	**Αρνητικός → Σωστό**		

Cout = 0 Cin = 0

	1	X	X	X
+	0	X	X	X
	0	**Θετικός → Σωστό**		

Cout = 1 Cin = 1

Το πρόβλημα της Υπερχείλισης στο Σύστημα Signed-2's complement

- **Να θυμάστε ότι το MSB είναι το πρόσημο. Αλλά προστίθεται και το πρόσημο! Το carry "δείχνει" υπερχείλιση μόνο όταν προσθέτω μη-προσημασμένους αριθμούς**

Θετικός – Θετικός →
μπορεί να βγάλει overflow

+	0	X	X	X
	0	X	X	X
	0	**Θετικός → Σωστό**		

Cout = 0 Cin = 0

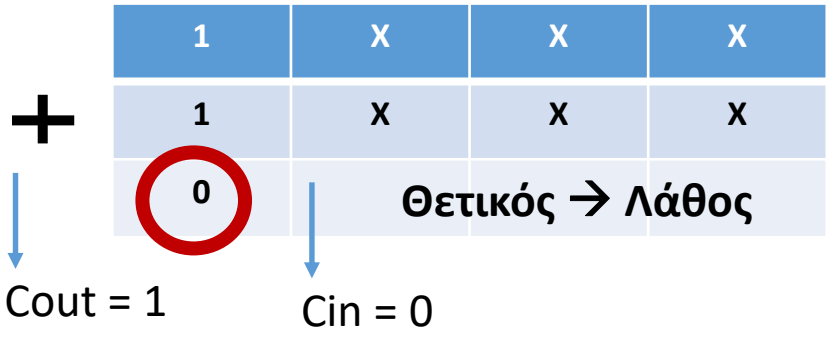
+	0	X	X	X
	0	X	X	X
	1	**Αρνητικός → Λάθος**		

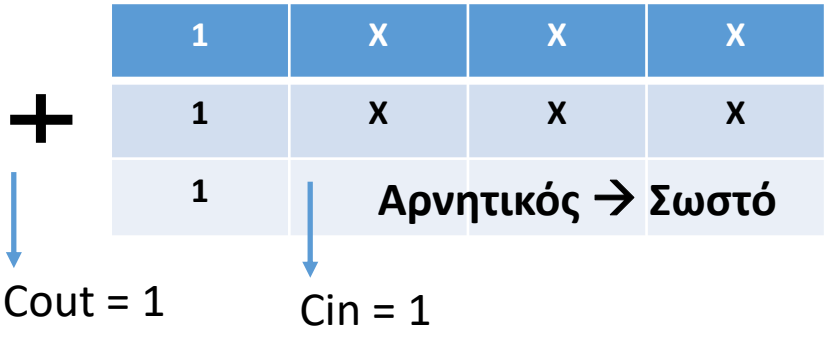
Cout = 0 Cin = 1

Το πρόβλημα της Υπερχείλισης στο Σύστημα Signed-2's complement

- **Να θυμάστε ότι το MSB είναι το πρόσημο. Αλλά προστίθεται και το πρόσημο! Το carry "δείχνει" υπερχείλιση μόνο όταν προσθέτω μη-προσημασμένους αριθμούς**

Αρνητικός – Αρνητικός →
μπορεί να βγάλει overflow



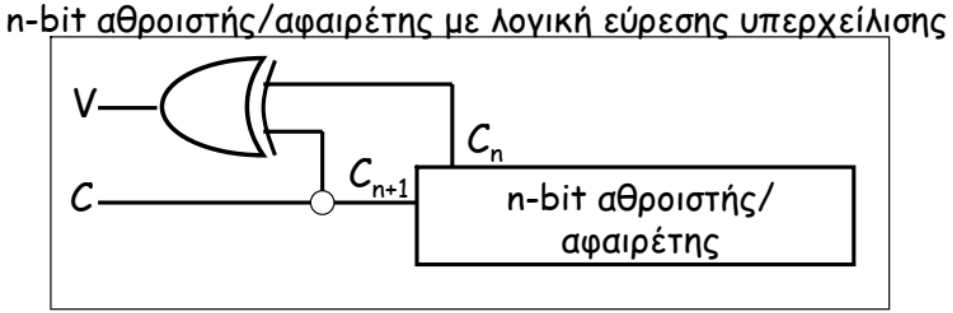


Το πρόβλημα της Υπερχείλισης στο Σύστημα Signed-2's complement

- Να θυμάστε ότι το MSB είναι το πρόσημο. Αλλά προστίθεται και το πρόσημο! Το carry "δείχνει" υπερχείλιση μόνο όταν προσθέτω μη-προσημασμένους αριθμούς
- Άρα, ένα carry_out == 1 δεν σημαίνει πάντα υπερχείλιση!
- Υπερχείλιση παρατηρείται ΜΟΝΟ όταν και οι 2 αριθμοί έχουν το ίδιο πρόσημο. Αυτή η κατάσταση μπορεί να βρεθεί όταν το τελικό carry out (Cn) είναι διαφορετικό από το carry της προηγούμενης θέσης (Cn-1)
- ΆΡΑ ΜΙΑ ΠΎΛΗ XOR

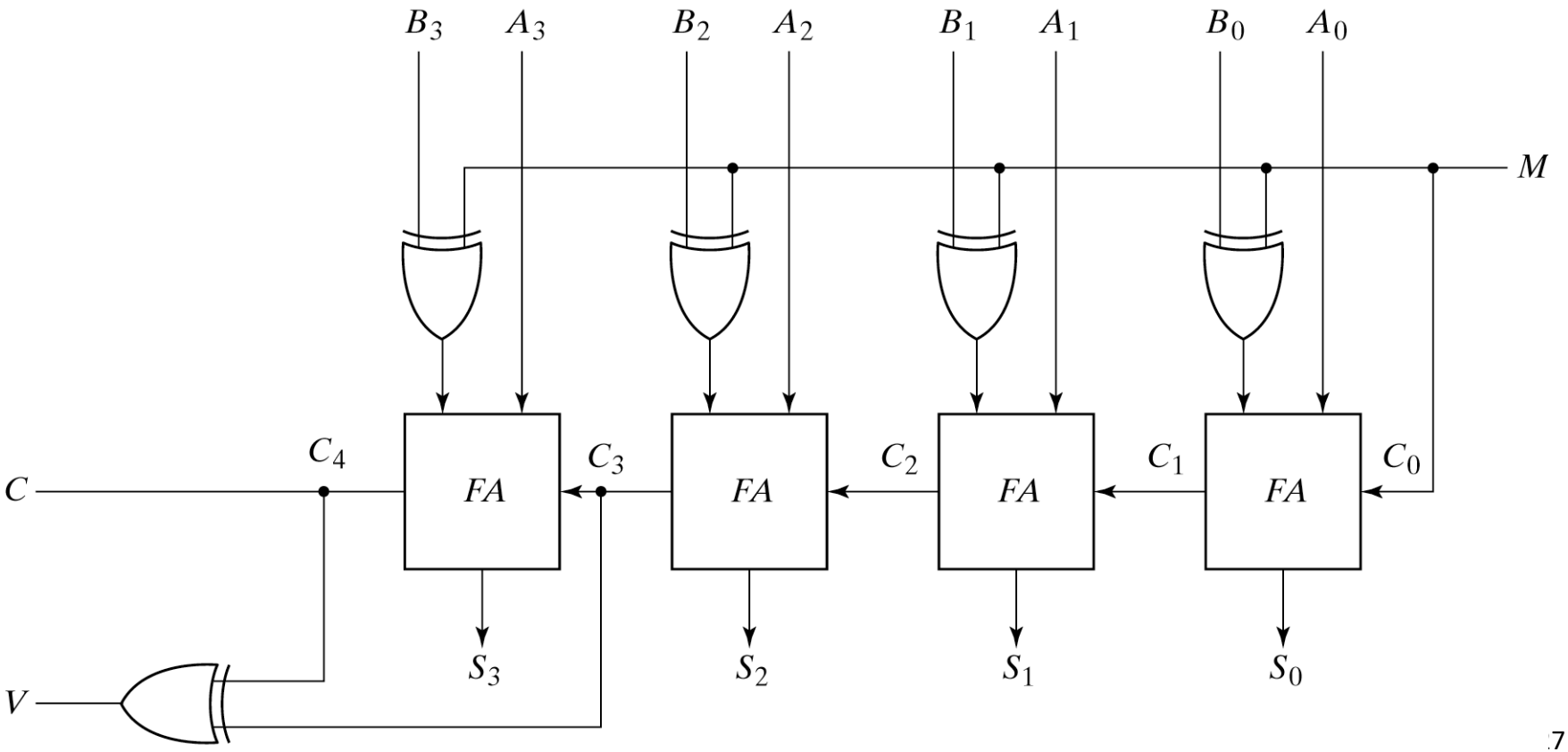
Συμπερασματικά

- **Οι καταστάσεις υπερχείλισης εντοπίζονται συγκρίνοντας τις τιμές στο carry in και curry out του sign bit (Cn-1 και Cn)**



- **το C =1 δείχνει υπερχείλιση όταν προσθέτουμε/αφαιρ. unsigned αριθμούς.**
- **το V=1 δείχνει υπερχείλιση όταν προσθέτουμε/αφαιρ. αριθμούς σε signed-2's complement**

Παράλληλος Αθροιστής/Αφαιρέτης



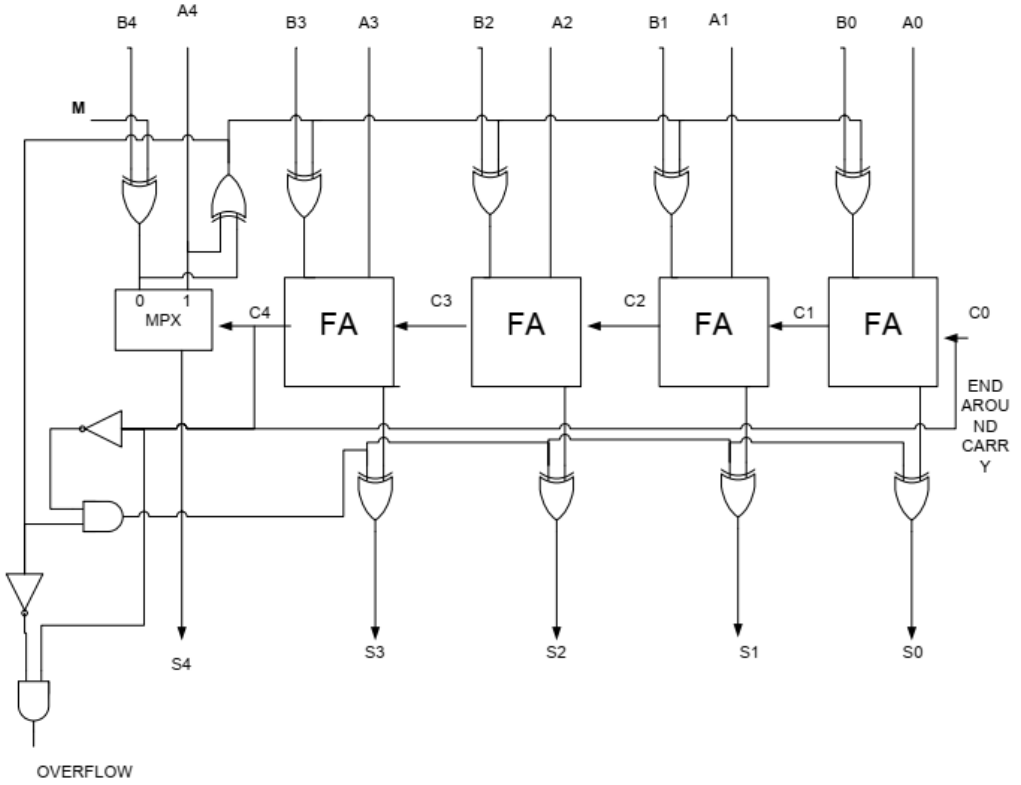
Παράλληλος αθροιστής για αριθμητική πρόσημου-μέτρου

- **Αριθμητική πρόσημου-μέτρου**
  - **πιο απλό**
  - **αλλά χάνουμε σε εύρος αριθμών,**
    - **Το bit προσήμου δεν παίζει ρόλο στις πράξεις ή στην εσωτερική αναπαράσταση του αριθμού αλλά μόνο στο πρόσημο του.**
      - **Εύρος: $[-(2^{N-1}-1), 2^{N-1}-1]$.**
  - **Δύο αναπαραστάσεις του 0 (0...00 και 1...00).**

	Signed Magnitude	1's Complement	2's Complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
0	0000 1000	0000 1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

Παράλληλος αθροιστής για αριθμητική πρόσημου-μέτρου

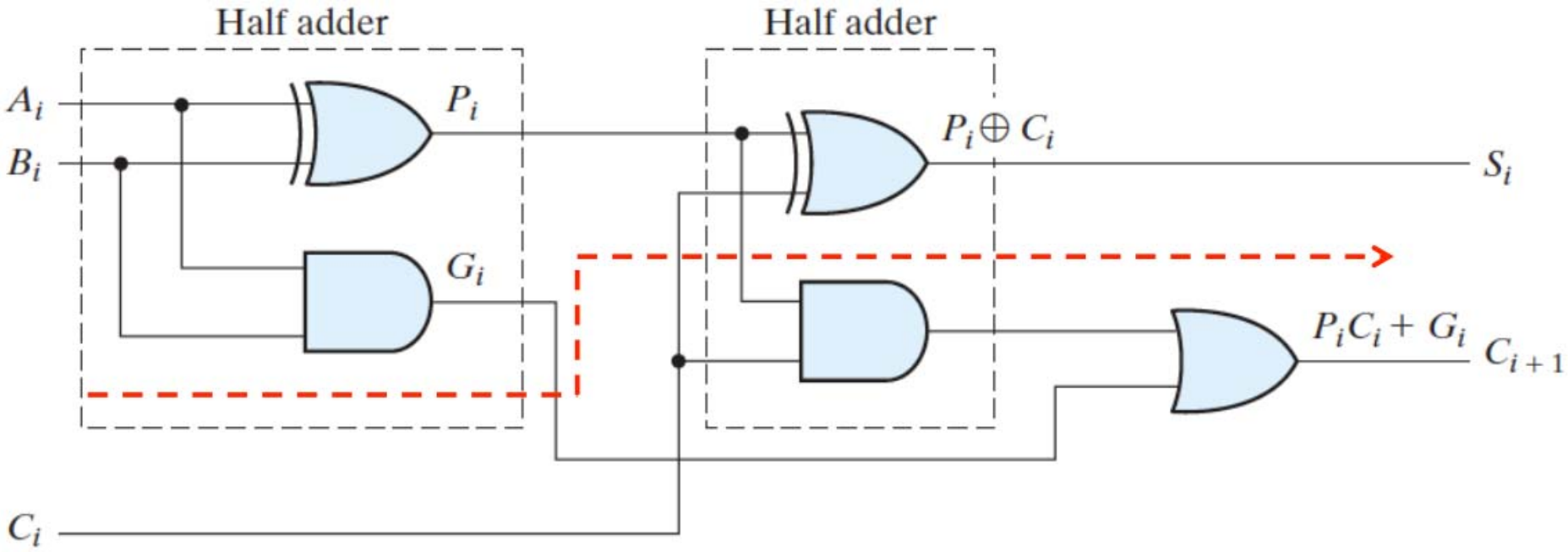
- Χρησιμοποιώντας αριθμητική προσήμου-μέτρου οδηγούμαστε σε πιο πολύπλοκα κυκλώματα → ο δεύτερος λόγος που χρησιμοποιούμε συμπληρώματα ως προς 2



Χρονική ανάλυση του παράλληλου αθροιστή

- Έστω:
  - κάθε carry μεταδίδεται στον επόμενο full adder σε χρόνο t_{rc}
  - ο πλήρης αθροιστής χρειάζεται t_d χρόνο για να υπολογίσει το S
- Ο δεύτερος εν σειρά αθροιστής θα χρειαστεί
  - $t_d + t_{rc}$ για να υπολογίσει το S_1
  - $2t_{rc}$ για να υπολογίσει το C_1
- Αν έχουμε N ψηφία, τότε τα αποτελέσματα θα ολοκληρωθούν σε χρόνο $t_d + (N-1)t_{rc}$
- Το τελευταίο carry θα εξαχθεί σε χρόνο Nt_{rc}.
- Αρκετά «αργό» σε περιπτώσεις που θέλουμε γρήγορη εξαγωγή αποτελεσμάτων.
  - Η διάδοση του κρατουμένου μας τρώει αρκετό από τον διαθέσιμο κύκλο ρολογιού καθώς εξαρτάται γραμμικά από το μήκος του adder

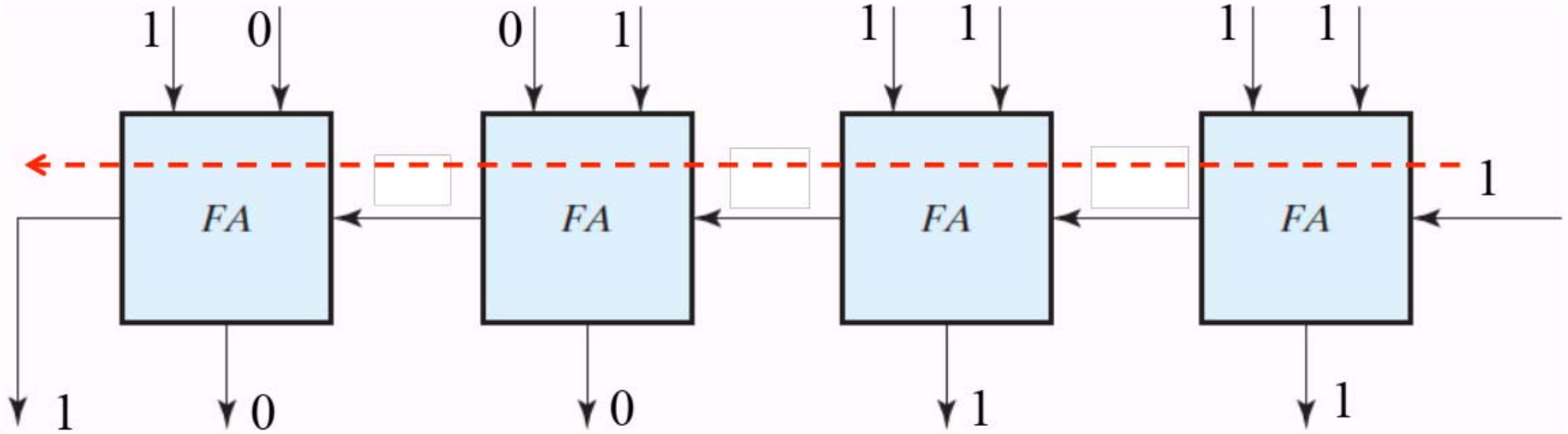
Διάδοση Κρατουμένου



- 2 επίπεδα πυλών για κάθε διάδοση κρατουμένου.
- $2 \times n$ επίπεδα για τον παράλληλο αθροιστή n bits.

Διάδοση Κρατουμένου

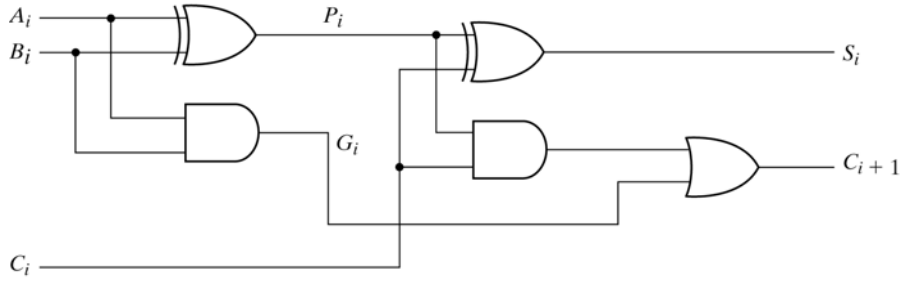
- **Η διάδοση του κρατουμένου έχει καθυστέρηση η οποία αυξάνει με την αύξηση των βαθμίδων**



Χρόνος Διάδοσης): Επίπεδα Πυλών × Καθυστέρηση Πύλης

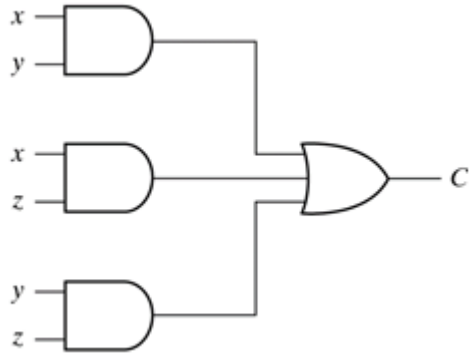
Διάδοση Κρατουμένου Δυαδικού Αθροιστή

- **Χρόνος Διάδοσης (Καθυστέρηση): Επίπεδα Πυλών x Καθυστέρηση Πύλης**
- **Παράλληλος Αθροιστής με διάδοση κρατουμένου : Η μεγαλύτερη καθυστέρηση οφείλεται στη διάδοση του κρατουμένου.**



- **2 επίπεδα πυλών για κάθε διάδοση κρατουμένου.**
- **2 x n επίπεδα για τον παράλληλο αθροιστή n bits.**

- **Η πρόσθεση είναι η πλέον συχνά χρησιμοποιούμενη πράξη.**
- **Η καθυστέρηση μπορεί να μειωθεί με τη χρήση γρηγορότερων πυλών. Ομως αυτή η λύση έχει σαφές άνω όριο οριζόμενο από την ισχύουσα τεχνολογία.**
- **Εναλλακτική λύση :**
  - **Η χρήση μιας πιο "παράλληλης" αρχιτεκτονικής**
  - **Πιο ακριβή σε υλικό**



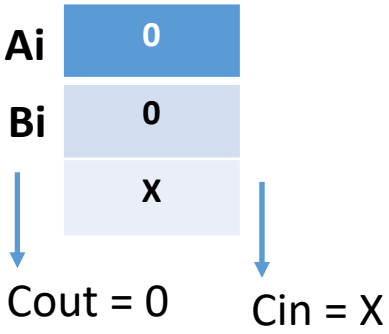
Αθροιστής Πρόβλεψης Κρατουμένου (Carry Lookahead Adder -- CLA)

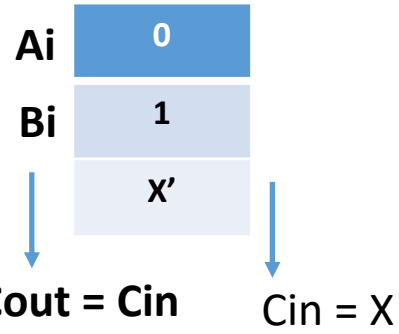
- **Εναλλακτικός σχεδιασμός για ένα συνδυαστικό αθροιστή με n-bit**
- **Πρακτικός σχεδιασμός με μειωμένη καθυστέρηση, αλλά απαιτεί πιο πολύπλοκο σχεδιασμό**
- **Παράγεται από ένα μετασχηματισμό του σχεδιασμού αθροιστή ριπής**

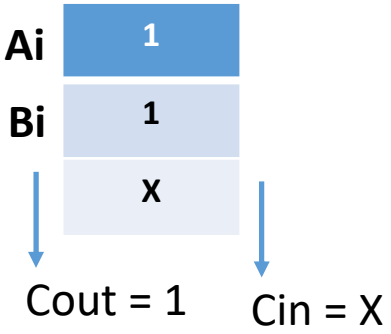
Σχεδιασμός CLA (Carry Look-Ahead)

- Από ένα FA, διαχωρίζουμε μεταξύ της παραγωγής (generation) του κρατουμένου (όταν ένα νέο κρατούμενο παράγεται, C_{out}=1) και της μετάδοσης (propagation) του κρατουμένου (όταν ένα υπάρχον C_{in} μεταδίδεται στο C_{out})
  - Παραγωγή: $G_i = A_iB_i$: if 1, C_{i+1}=1
  - Μετάδοση: $P_i = A_i \oplus B_i$: εάν 1 τότε $C_{i+1} = C_i$
- Ένα bit από λογική G/P μόνο δεν βοηθά, αλλά…
- Διαδοχική λογική G/P μπορεί να παράγει το κρατούμενο εξόδου ενός μπλοκ

Διάδοση (Όχι Υπολογισμός) Κρατουμένου

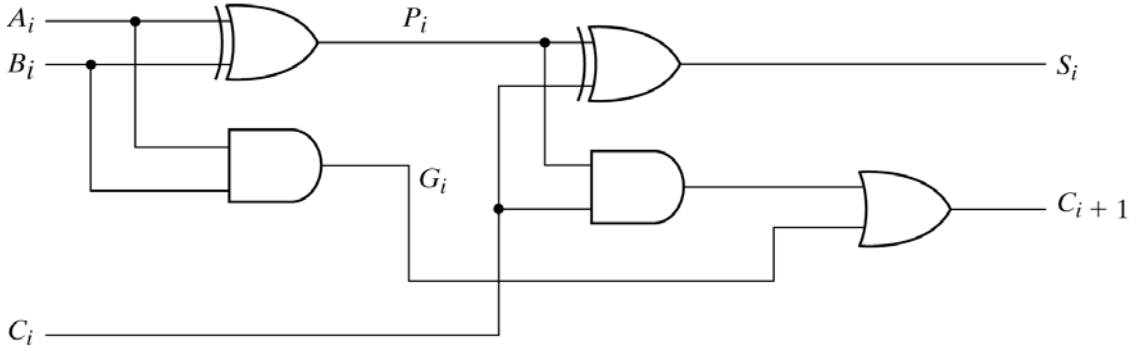






- **Μετάδοση: Pi = $A_i \oplus B_i$: εάν 1 τότε $C_{i+1} = C_i$**

Πρόβλεψη Κρατουμένου (Carry Look-Ahead)



$P_i = A_i \oplus B_i$ $S_i = P_i \oplus C_i$

$G_i = A_i B_i$ $C_{i+1} = G_i + P_i C_i$

- Η σχέση $C_{i+1} = G_i + P_i C_i$ είναι αναδρομική
- Αναπτύσσοντας την αναδρομή, μπορούμε να πάρουμε τις εξής εξισώσεις :

  C_0 = κρατούμενο εισόδου

  $C_1 = G_0 + P_0 C_0$

  $C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0 = A_1 B_1 + (A_1 \oplus B_1) A_0 + (A_1 \oplus B_1) (A_0 \oplus B_0) C_0$

  $C_3 = G_2 + P_2 C_2 = \ldots = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$
- Μπορούμε αυτές να τις υπολογίσουμε παράλληλα !

Γεννήτρια Πρόβλεψης Κρατουμένου

$C_1 = G_0 + P_0C_0$

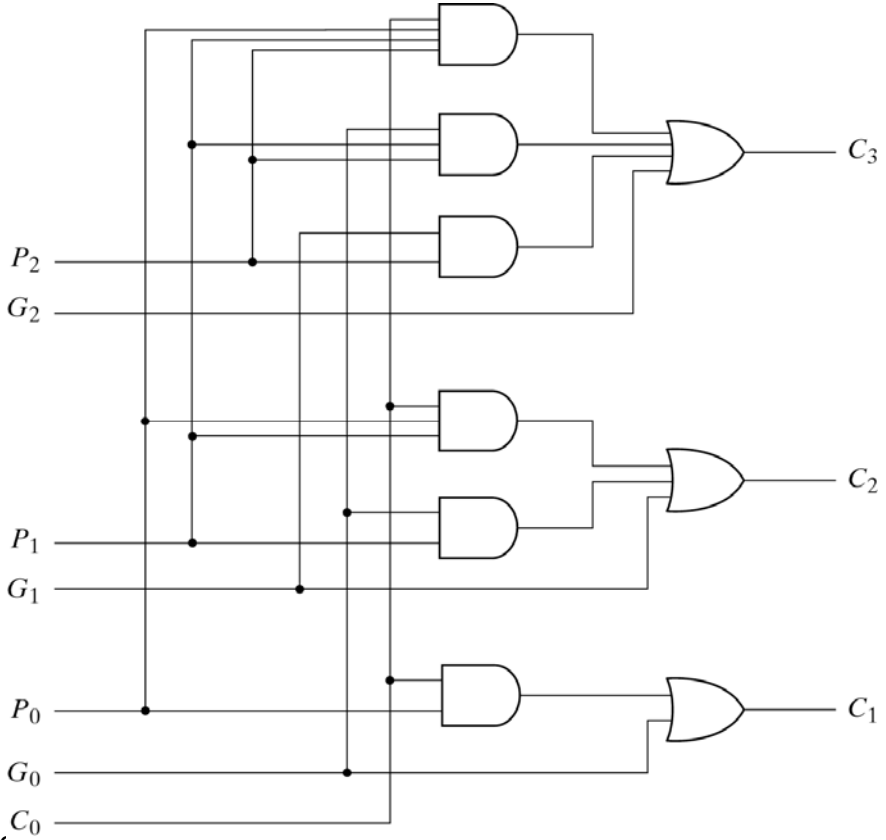
$C_2 = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

$C_4 = \ldots$

$C_5 = \ldots$

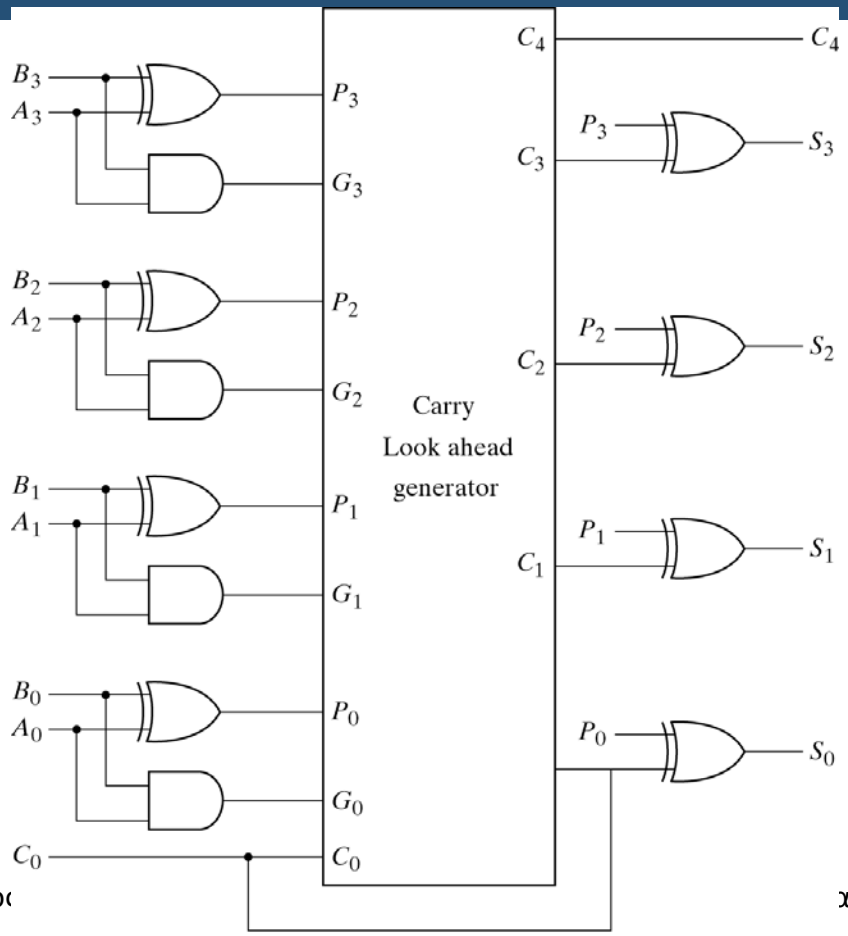
Εξαρτώνται από τα: Ai, Bi και C_0



Αθροιστής Πρόβλεψης Κρατουμένου (CLA)

$P_i = A_i \oplus B_i$

$G_i = A_i B_i$



$S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_i C_i$

Συγκριτής (Comparator)

Συγκριτής Μεγέθους: συγκρίνει δύο αριθμούς και βρίσκει τη σχέση τους (<,>,=).

Για δύο αριθμούς των *n* bits έχουμε 2^{2n} συνδυασμούς.

Το κύκλωμα του συγκριτή έχει αρκετή κανονικότητα.

- Έστω $A = A_3A_2A_1A_0$ και $B = B_3B_2B_1B_0$ οι δύο αριθμοί.

- Ισχύει A = B όταν όλα τα ζευγάρια (A_i, B_i) είναι ίσα, δηλαδή $A_3=B_3$ και $A_2=B_2$ και $A_1=B_1$ και $A_0=B_0$.

$$(A=B) = x_3x_2x_1x_0 \qquad x_i = A_iB_i + A_i'B_i'$$

Συγκριτής Μεγέθους

- **Για να βρούμε εάν A<B ή A>B εξετάζουμε τα σχετικά μεγέθη των ζευγαριών ψηφίων ξεκινώντας από την πιο σημαντική θέση. Εάν τα δύο ψηφία είναι ίσα τότε συγκρίνουμε το επόμενο λιγότερο σημαντικό ζευγάρι ψηφίων.**
- **Εάν $A_i = 1$ και $B_i = 0$ τότε A > B, εάν $A_i = 0$ και $B_i = 1$ τότε A < B**

A = 0, B=0, x=1 --- AB’ = 0

A = 0, B=1, x=0 --- AB’ = 0

A = 1, B=0, x=0 --- AB’ = 1

A = 1, B=1, x=1 --- AB’ = 0

Αν AB’ = 1 → A > B

Συγκριτής Μεγέθους

- **Για να βρούμε εάν A<B ή A>B εξετάζουμε τα σχετικά μεγέθη των ζευγαριών ψηφίων ξεκινώντας από την πιο σημαντική θέση. Εάν τα δύο ψηφία είναι ίσα τότε συγκρίνουμε το επόμενο λιγότερο σημαντικό ζευγάρι ψηφίων.**
- **Εάν $A_i = 1$ και $B_i = 0$ τότε A > B, εάν $A_i = 0$ και $B_i = 1$ τότε A < B**

$$(A>B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

$$(A<B) = B_3A_3' + x_3B_2A_2' + x_3x_2B_1A_1' + x_3x_2x_1B_0A_0'$$

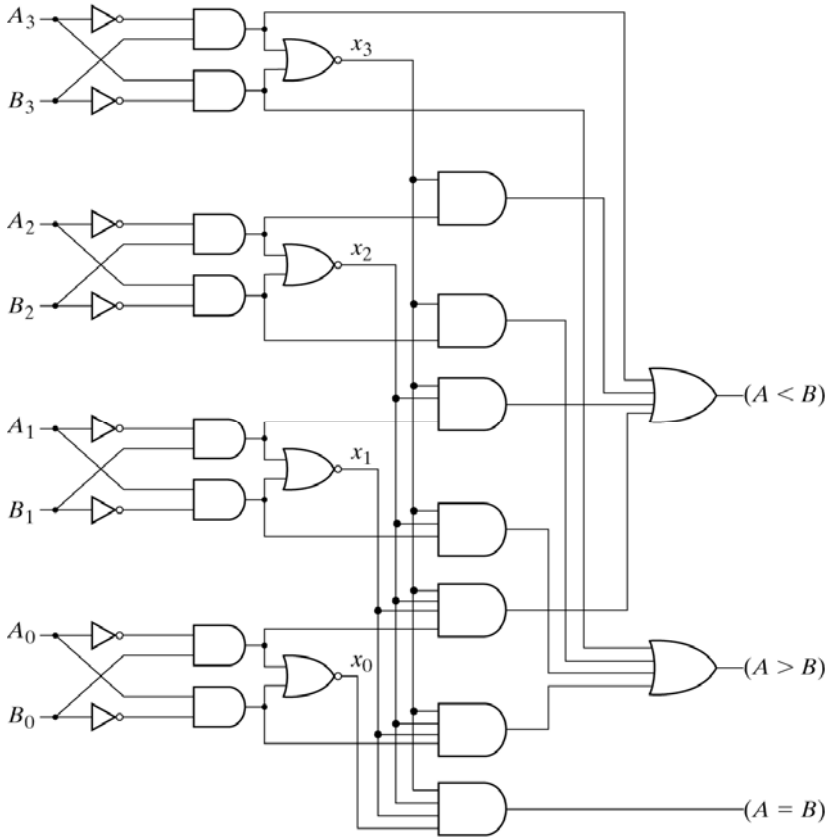
$$x_i = A_iB_i + A_i'B_i'$$

Συγκριτής Μεγέθους

$(A=B) = x_3x_2x_1x_0 \qquad x_i = A_iB_i + A_i'B_i'$

$(A>B)=A_3B_3'+x_3A_2B_2'+x_3x_2A_1B_1'+x_3x_2x_1A_0B_0'$

$(A<B)=B_3A_3'+x_3B_2A_2'+x_3x_2B_1A_1'+x_3x_2x_1B_0A_0'$



Πολλαπλασιαστής

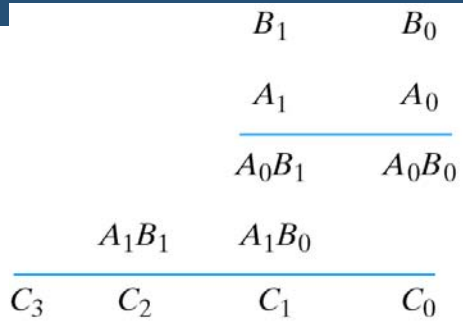
- Αποτελεί το 10% κατά μέγιστο του συνολικού ημιαγωγικού υλικού σε ένα ψηφιακό επεξεργαστή
- Συνήθως μοιράζεται το hardware και με άλλες πράξεις όπως αυτές της διαίρεσης και της στρογγυλοποίησης προκειμένου να επιτευχθεί μεγαλύτερη ταχύτητα
- Κατά μέσο όρο η πράξη του πολλαπλασιασμού διαρκεί περίπου τέσσερις φορές από ότι η πρόσθεση
- Το 1/3 των αριθμητικών πράξεων είναι πολλαπλασιασμοί

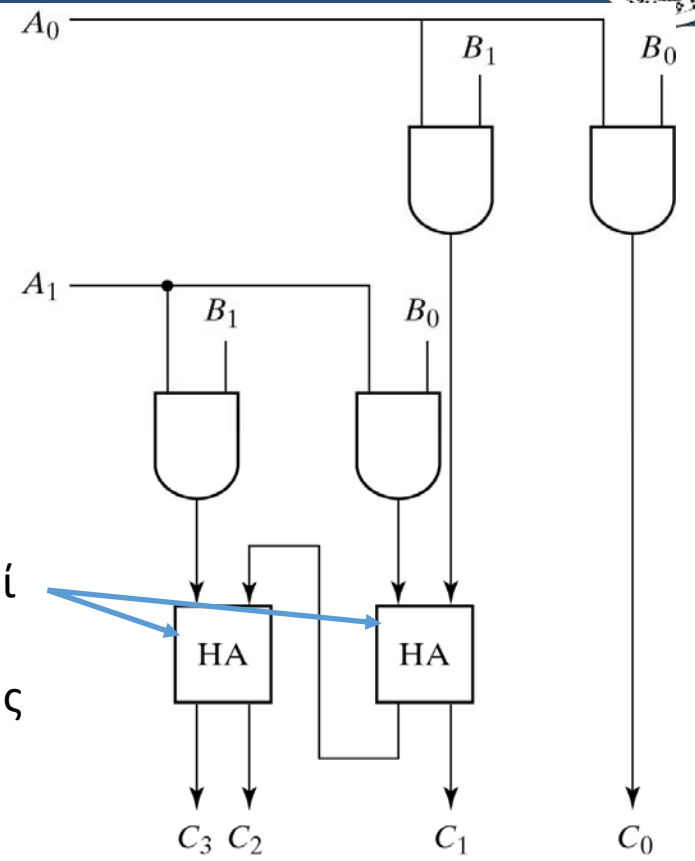
Πολλαπλασιασμός: Ολίσθηση και Πρόσθεση

```
1011001 x 1101 :
      1011001        Πολλαπλασιαστέος
         1101 x      Πολλαπλασιαστής
-------------------
1 :    1011001
0 :   0000000
1 :  1011001
1 : 1011001     +
-------------------
  10010000101
```

Προσθέτουμε από δεξιά προς αριστερά

Δυαδικός πολλαπλασιαστής





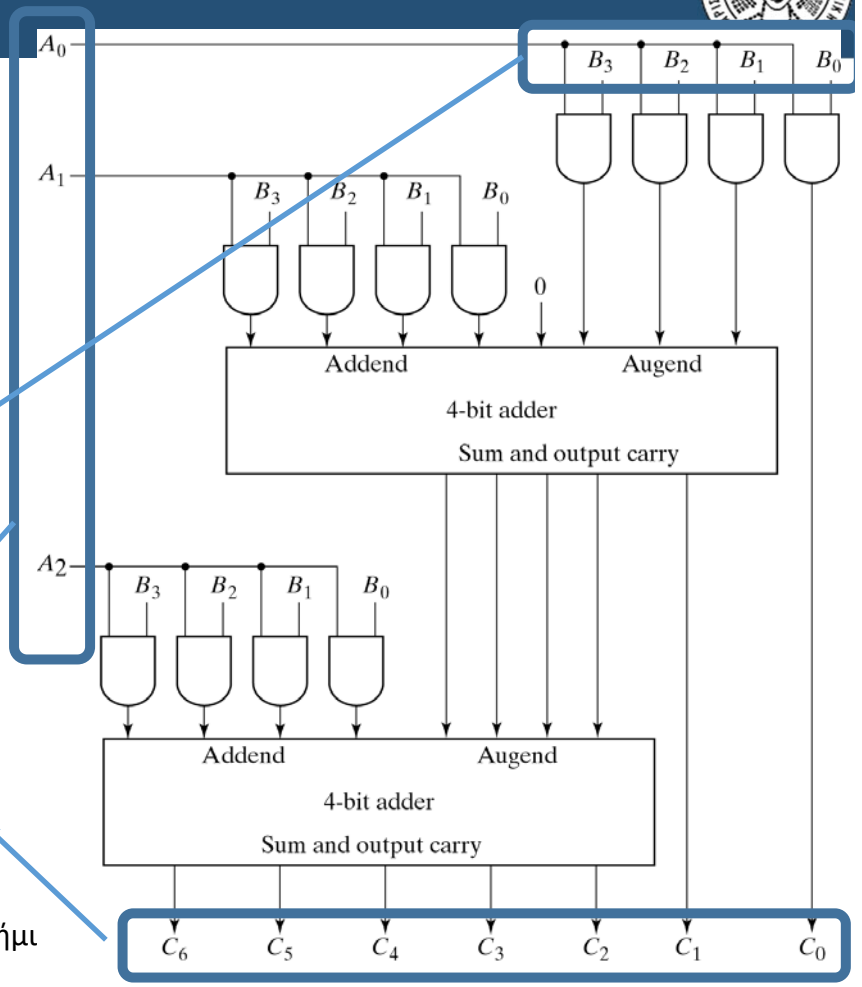
AND gate == επί (x)

Οι Half Adders είναι αρκετοί αφού δεν υπάρχει Carry-in μαζί με τις δύο εισόδους της πρόσθεσης.

Δυαδικός πολλαπλασιαστής

- **Κύκλωμα πολ/στη 4x3 bit**

$$
\begin{array}{rl}
 & B_3B_2B_1B_0 \\
\text{x} & A_2A_1A_0 \\
\hline
+ & 0 \quad A_0B_3 \quad A_0B_2 \quad A_0B_1 \quad A_0B_0 \\
+ & A_1B_3 \quad A_1B_2 \quad A_1B_1 \quad A_1B_0 \\
 & A_2B_3 \quad A_2B_2 \quad A_2B_1 \quad A_2B_0 \\
\hline
 & \text{***Τελικό Γινόμενο***}
\end{array}
$$



Πολλαπλασιαστέος

Πολλαπλασιαστής

Αποτέλεσμα 7 bits

Πολλαπλασιασμός Χρησιμοποιώντας Αποθηκευμένες Τιμές

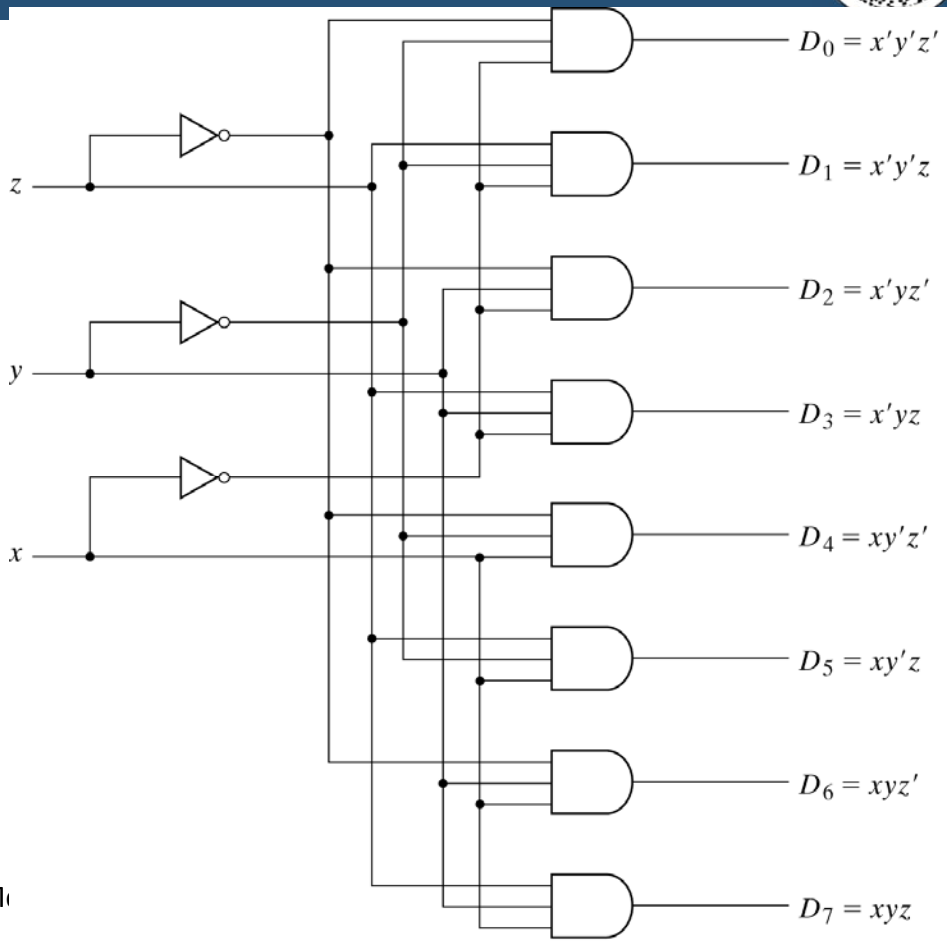
- **Ονομάζονται Look up tables (Μικρή καθυστέρηση, Εκθετικό μέγεθος)**

	000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
000	000000	000000	000000	000000	000000	000000	000000	000000
001	000000	000001	000010	000011	000100	000101	000110	000111
010	000000	000010	000100	000110	001000	001010	001100	001110
011	000000	000011	000110	001001	001100	001111	010010	010101
100	000000	000100	001000	001100	010000	010100	011000	011100
101	000000	000101	001010	001111	010100	011001	011110	100011
110	000000	000110	001100	010010	011000	011110	100100	101010
111	000000	000111	001110	010101	011100	100011	101010	110001

Αποκωδικοποιητής (Decoder)

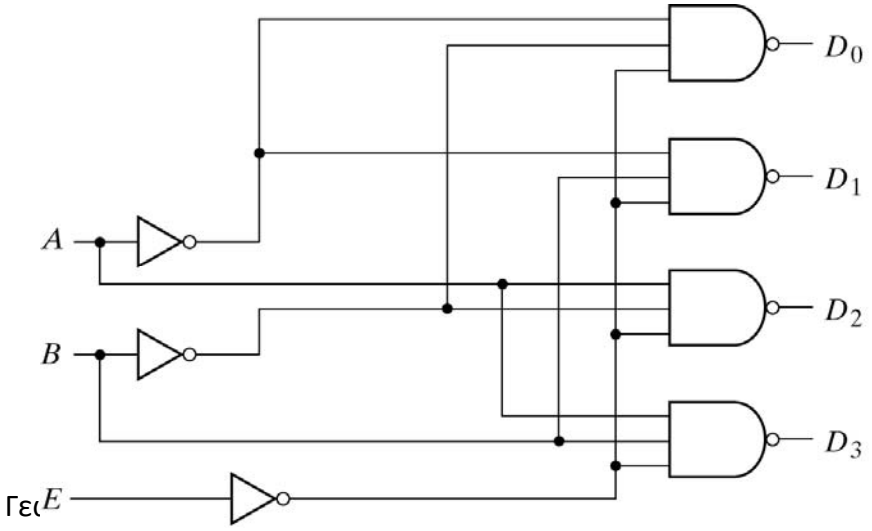
- **Αποκωδικοποιητής:** κύκλωμα που μετατρέπει τη δυαδική πληροφορία των n γραμμών εισόδου σε εως 2^n μοναδικές γραμμές εξόδου (ελαχιστόροι n μεταβλητών).
- **Παράδειγμα:** Αποκωδικοποιητής 3-σε-8
- Ο αποκωδικοποιητής παράγει τον 1 από 2^n κώδικα.

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



Αποκωδικοποιητής με Είσοδο Επίτρεψης

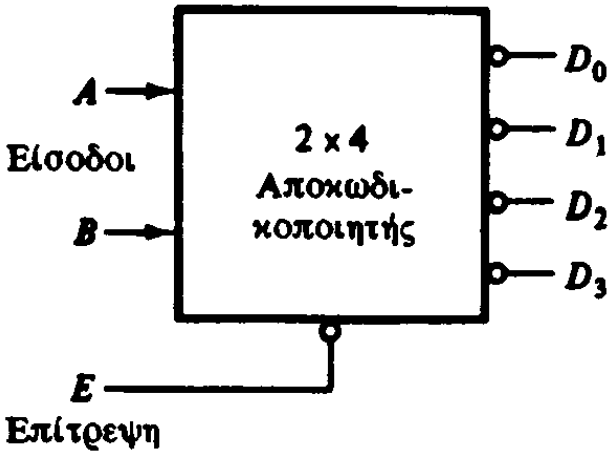
- **Ο αποκωδικοποιητής μπορεί να παράγει συμπληρωματικές εξόδους**
- **Ο αποκωδικοποιητής μπορεί να έχει είσοδο επίτρεψης (συνήθως χρησιμοποιούμε την ορολογία αποπλέκτης)**



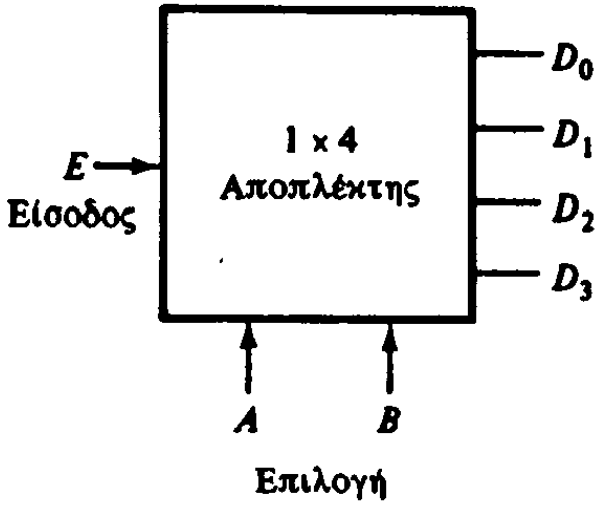
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Λειτουργία ως αποπλέκτης (Demultiplexer)

- **Ο αποπλέκτης δέχεται πληροφορίες από μία απλή γραμμή και τις μεταβιβάζει σε μία από τις 2^n δυνατές γραμμές εξόδου ανάλογα με τις τιμές των n γραμμών επιλογής.**

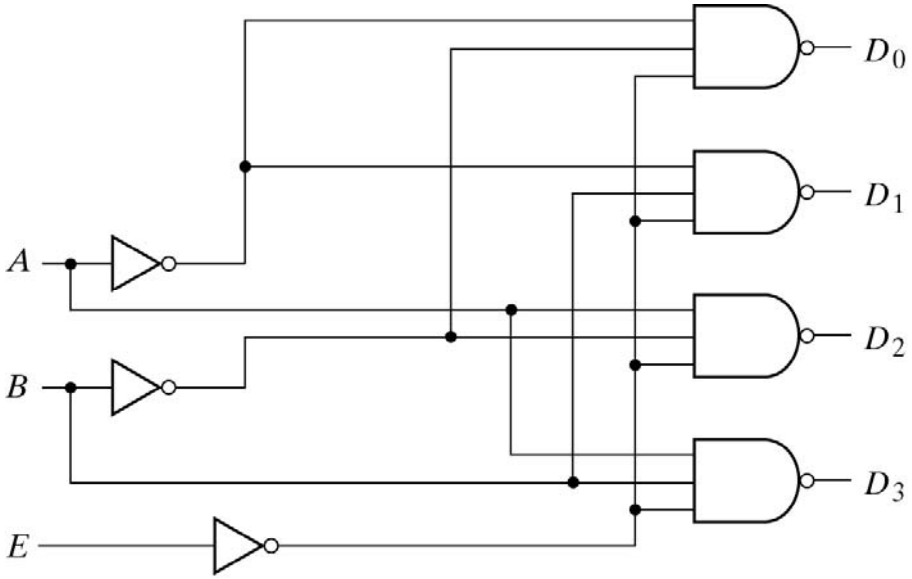


(α) Αποκωδικοποιητής με επίτρεψη



(β) Αποπλέκτης

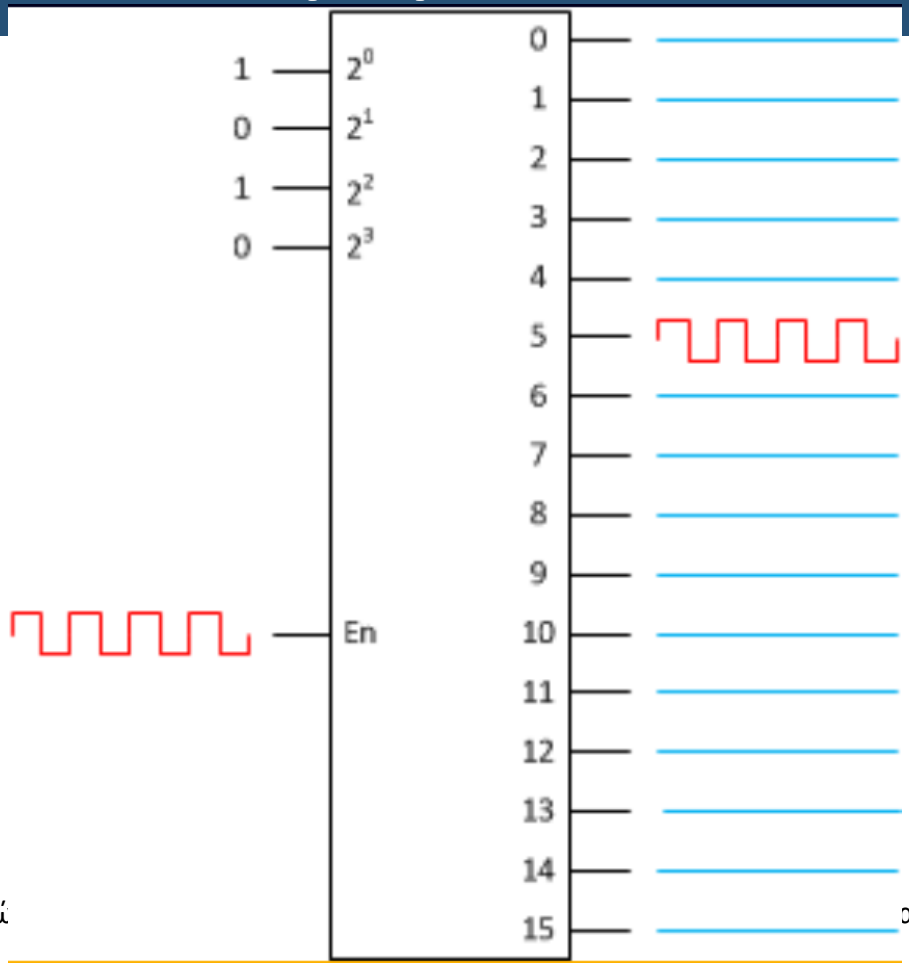
Λειτουργία ως Αποπλέκτης



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

- **Αν A=0, B=1,**
  - **Και E = 0 → D1 = 0**
  - **Και E = 1 → D1 = 1**
- **Η τιμή του E μεταφέρεται στην έξοδο D1**

Παράδειγμα Αποπλεξίας

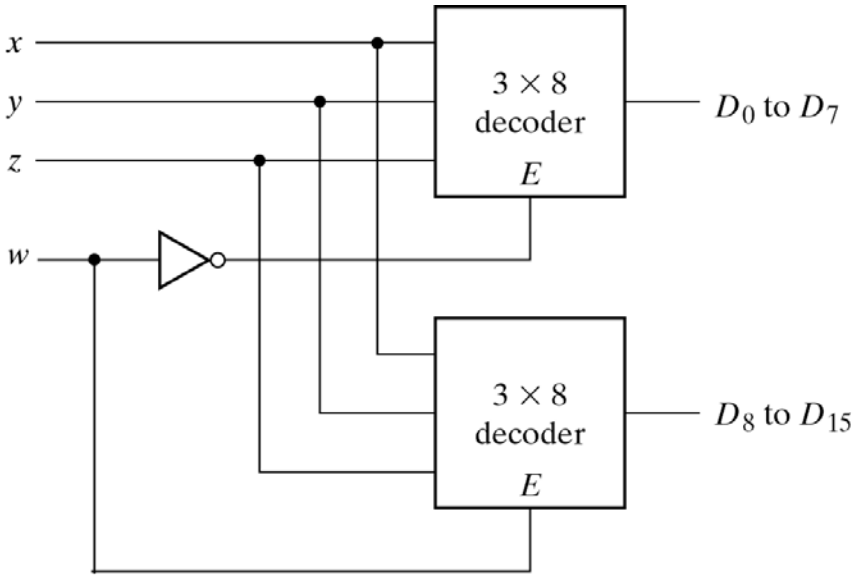


Αποκωδικοποιητής/Αποπλέκτης

Επέκταση αποκωδικοποιητή με χρήση πολλών αποπλεκτών

⇨

2 αποπλέκτες 3 σε 8 δίνουν 1 αποκωδικοποιητή 4 σε 16



Υλοποίηση Συνάρτησης με Αποκωδικοποιητή

- **Αφού ο αποκωδικοποιητής παράγει τους 2^n ελαχιστόρους μπορεί να χρησιμοποιηθεί για να υλοποιήσει οποιαδήποτε συνάρτηση.**

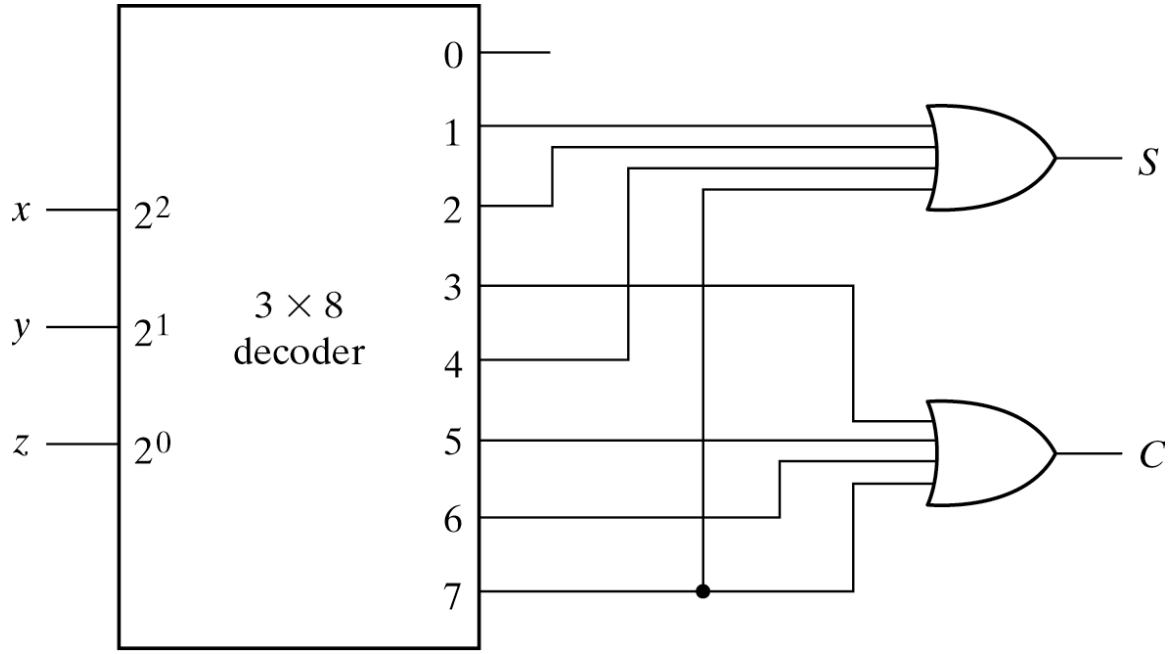
- **Κάθε συνδυαστικό κύκλωμα με n εισόδους και m εξόδους μπορεί να υλοποιηθεί με έναν αποκωδικοποιητή n-σε-2^n γραμμών και m πύλες OR.**

- **Εάν ο αριθμός των ελαχιστόρων μιας συνάρτησης είναι μεγαλύτερος από $2^n/2$, τότε μπορούμε να χρησιμοποιήσουμε μία πύλη NOR για να αθροίσουμε τους ελαχιστόρους της F΄. Η έξοδος της πύλης NOR δίνει τη συνάρτηση F.**

Παράδειγμα

- **Υλοποίηση πλήρους αθροιστή με αποκωδικοποιητή.**

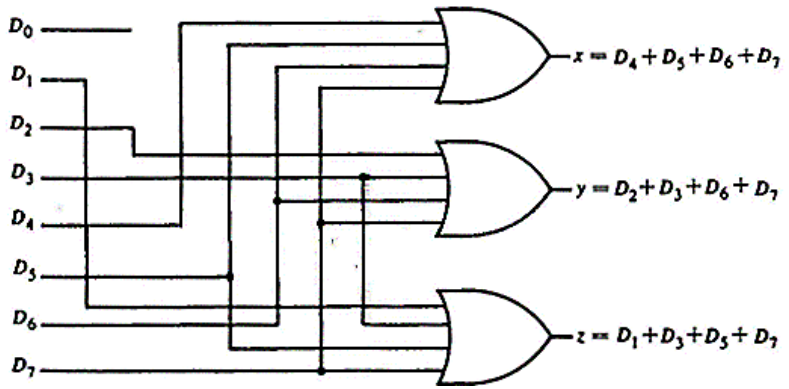
S(x,y,z) = Σ(1,2,4,7)
C(x,y,z) = Σ(3,5,6,7)



Κωδικοποιητής (Encoder)

- **Ο Κωδικοποιητής εκτελεί την αντίστροφη λειτουργία από τον Αποκωδικοποιητή: Έχει 2^n γραμμές εισόδου και n γραμμές εξόδου και δίνει στην έξοδο τον δυαδικό κώδικα που αντιστοιχεί στις γραμμές εισόδου.**

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



Κωδικοποιητής (Encoder)

- **Ο Κωδικοποιητής εκτελεί την αντίστροφη λειτουργία από τον Αποκωδικοποιητή: Έχει 2^n γραμμές εισόδου και n γραμμές εξόδου και δίνει στην έξοδο τον δυαδικό κώδικα που αντιστοιχεί στις γραμμές εισόδου.**

Προβλήματα:

- Όταν περισσότερες της μίας είσοδοι είναι 1 τότε η έξοδος είναι απροσδιόριστη. (Λύση: προτεραιότητα)
- Όταν όλες οι είσοδοι είναι 0 τότε η έξοδος είναι 0 που δεν είναι σωστό αφού η $D_0 \neq 1$. (Λύση: διάκριση της κατάστασης)

Κωδικοποιητής Προτεραιότητας (Priority Encode

- Ο κωδικοποιητής προτεραιότητας είναι ένα κύκλωμα κωδικοποιητή που περιλαμβάνει συνάρτηση προτεραιότητας.
- Λύνει το πρόβλημα της επιλογής όταν περισσότερες της μίας εισόδων είναι 1 επιλέγοντας αυτή με τη μεγαλύτερη προτεραιότητα.
- Παράδειγμα: Κωδικοποιητής προτεραιότητας 4 εισόδων

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

Μου καθορίζει αν οι έξοδοι X, Y έχουν valid τιμή (ενδείκτης έγκυρης εξόδου)

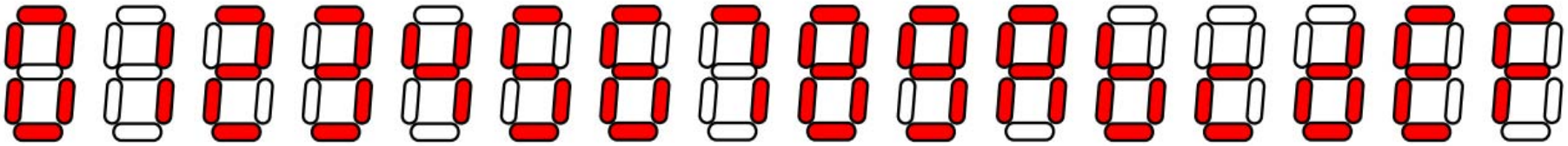
Κωδικοποιητής Προτεραιότητας (Priority Encode

- **Ο κωδικοποιητής προτεραιότητας είναι ένα κύκλωμα κωδικοποιητή που περιλαμβάνει συνάρτηση προτεραιότητας.**
- **Λύνει το πρόβλημα της επιλογής όταν περισσότερες της μίας εισόδων είναι 1 επιλέγοντας αυτή με τη μεγαλύτερη προτεραιότητα.**
- **Παράδειγμα: Κωδικοποιητής προτεραιότητας 4 εισόδων**

Άσκηση για το σπίτι: Σχεδίαση του Priority Encoder με πύλες

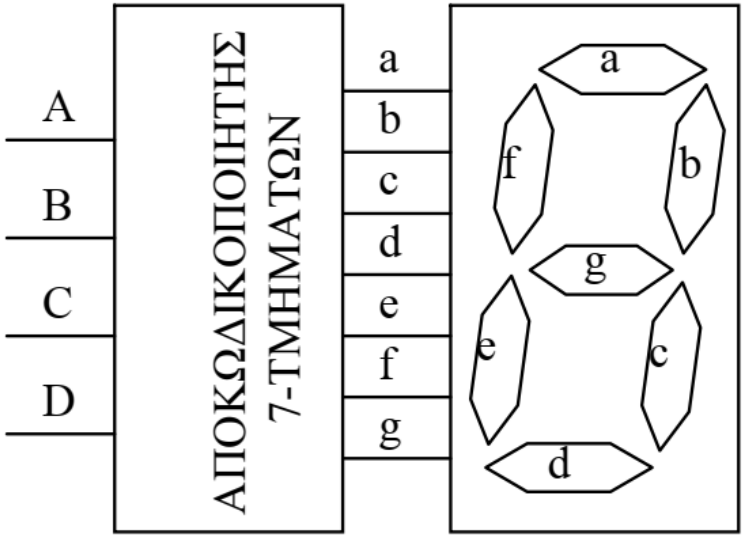
Άσκηση – Αποκωδικοποιητές

- **Να σχεδιάσετε έναν αποκωδικοποιητή «HEX σε απεικόνιση 7-τμημάτων» (7-segment decoder). Χρησιμοποιήστε χάρτες Karnaugh για την απλοποίηση των 7 λογικών συναρτήσεων a, b, c, d, e, f, g.**
- **Η απεικόνιση των αριθμών του δεκαεξαδικού συστήματος (0-F) στο display θα γίνεται ως εξής:**



Άσκηση – Αποκωδικοποιητές

- **Να σχεδιάσετε έναν αποκωδικοποιητή «HEX σε απεικόνιση 7-τμημάτων» (7-segment decoder). Χρησιμοποιήστε χάρτες Karnaugh για την απλοποίηση των 7 λογικών συναρτήσεων a, b, c, d, e, f, g.**



[0]={a,b,c,d,e,f}	[8]={a,b,c,d,e,f,g}
[1]={b,c}	[9]={a,b,c,d,f,g}
[2]={a,b,d,e,g}	[A]={a,b,c,e,f,g}
[3]={a,b,c,d,g}	[B]={c,d,e,f,g}
[4]={b,c,f,g}	[C]={d,e,g}
[5]={a,c,d,f,g}	[D]={b,c,d,e,g}
[6]={a,c,d,e,f,g}	[E]={a,d,e,f,g}
[7]={a,b,c}	[F]={a,e,f,g}

Άσκηση – Αποκωδικοποιητές

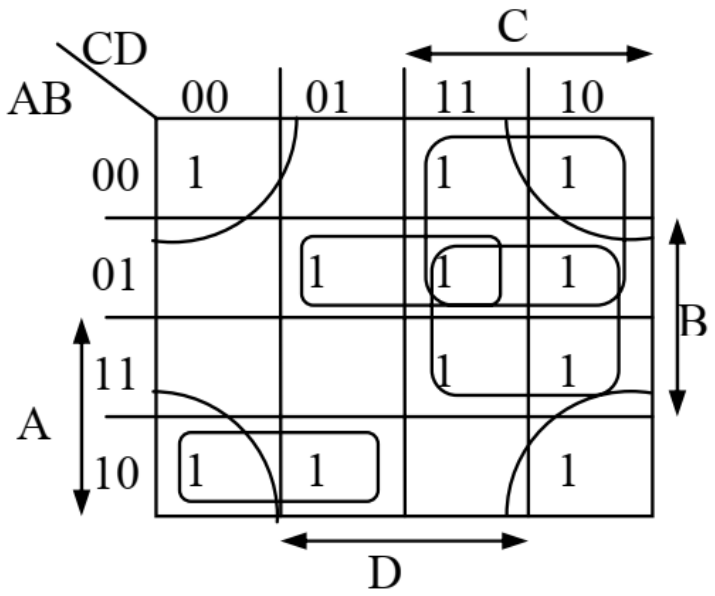
- **Να γράψετε όλες τις απλοποιημένες συναρτήσεις και να σχεδιάσετε τα λογικά κυκλώματα μόνο για τις συναρτήσεις f και g.**
- **Θεωρείστε ότι οι 4 είσοδοι του αποκωδικοποιητή είναι A, B, C, D (όπου A είναι το MSB).**
- **Καθεμιά από τις διόδους led a, b, c, d, e, f, g, του display φωτοβολεί όταν η αντίστοιχη είσοδός της βρίσκεται σε λογικό «1».**

Άσκηση – Αποκωδικοποιητές

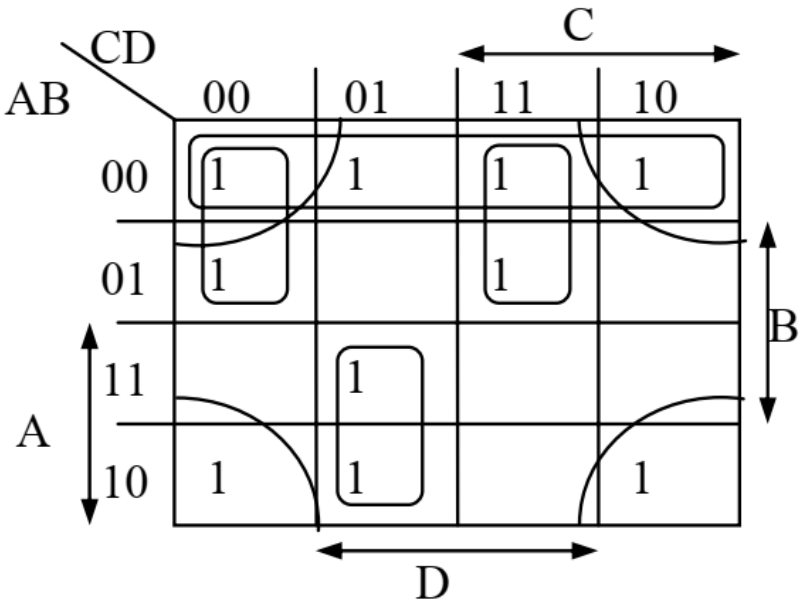
- **Υπολογίζουμε το Πίνακα Αληθείας**
- **Στην συνέχεια εφαρμόζουμε την μέθοδο του χάρτη Karnaugh για κάθε τιμή ξεχωριστά**

A	B	C	D	Αριθμός	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1
1	0	1	0	A	1	1	1	0	1	1	1
1	0	1	1	B	0	0	1	1	1	1	1
1	1	0	0	C	0	0	0	1	1	0	1
1	1	0	1	D	0	1	1	1	1	0	1
1	1	1	0	E	1	0	0	1	1	1	1
1	1	1	1	F	1	0	0	0	1	1	1

Άσκηση – Αποκωδικοποιητές

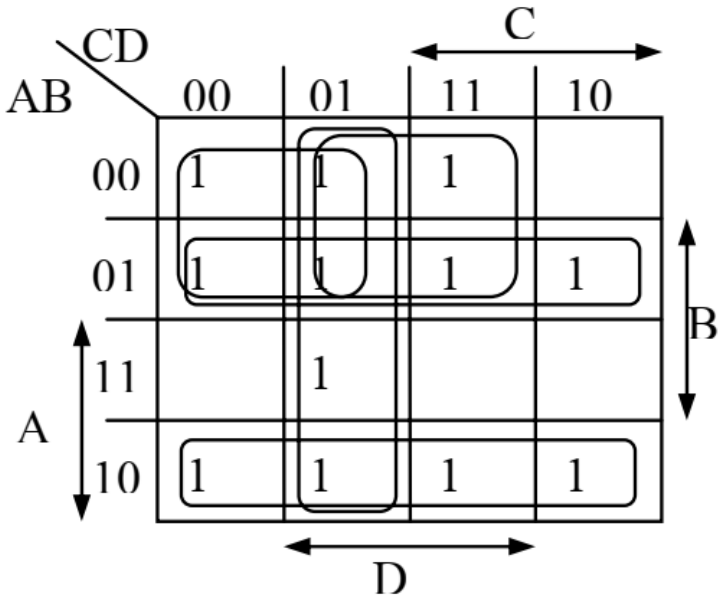


a = A'C+BC+B'D'+A'BD+AB'C'

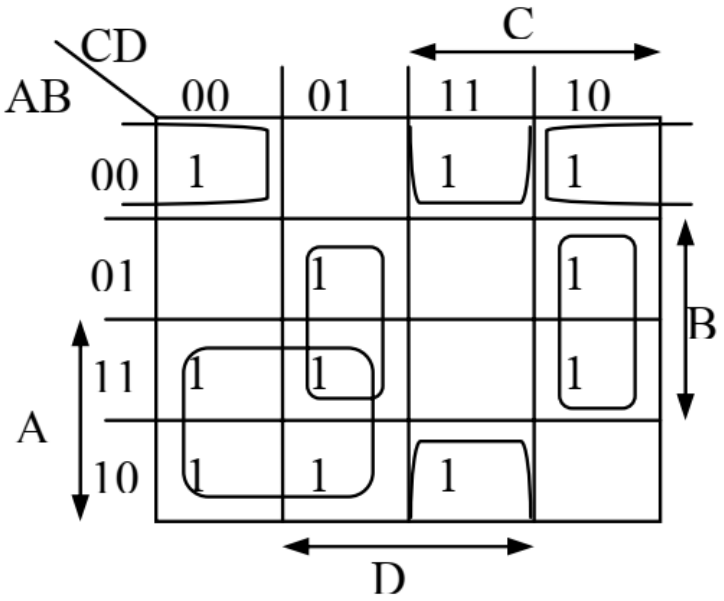


b = A'B'+B'D'+A'C'D'+A'CD+AC'D

Άσκηση – Αποκωδικοποιητές

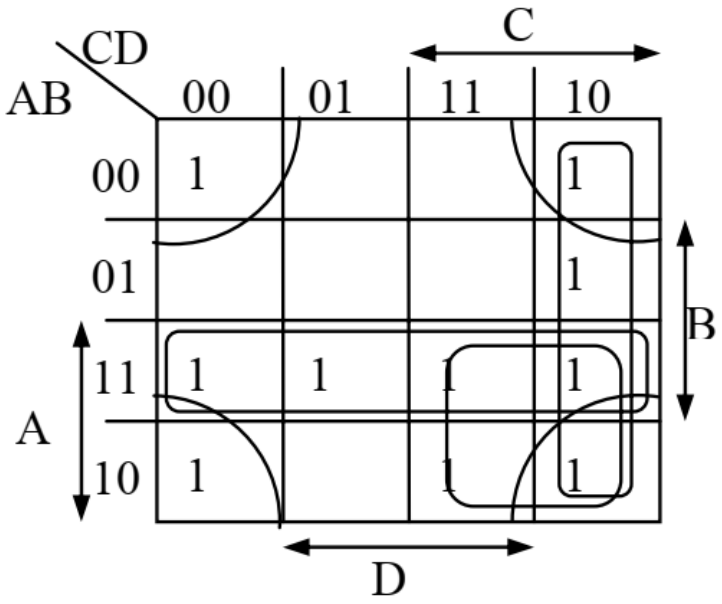


c = A'C'+A'D+A'B+AB'+C'D

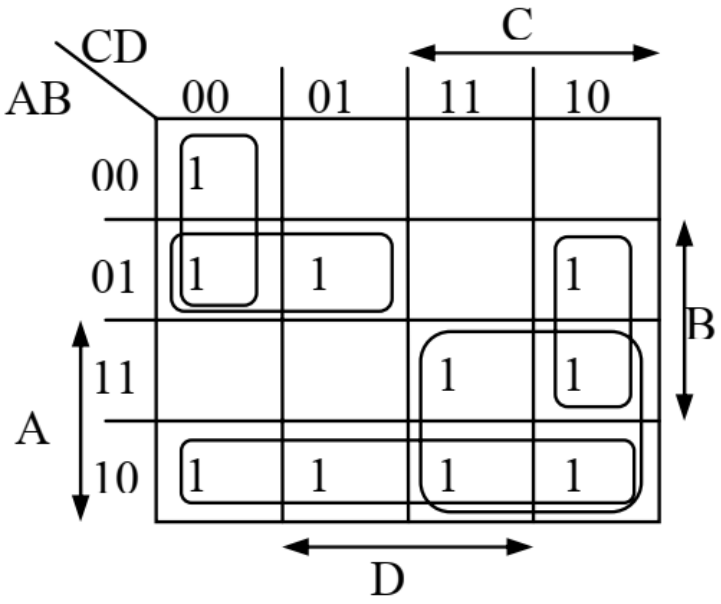


d = AC'+BC'D+BCD'+A'B'D'+B'CD

Άσκηση – Αποκωδικοποιητές

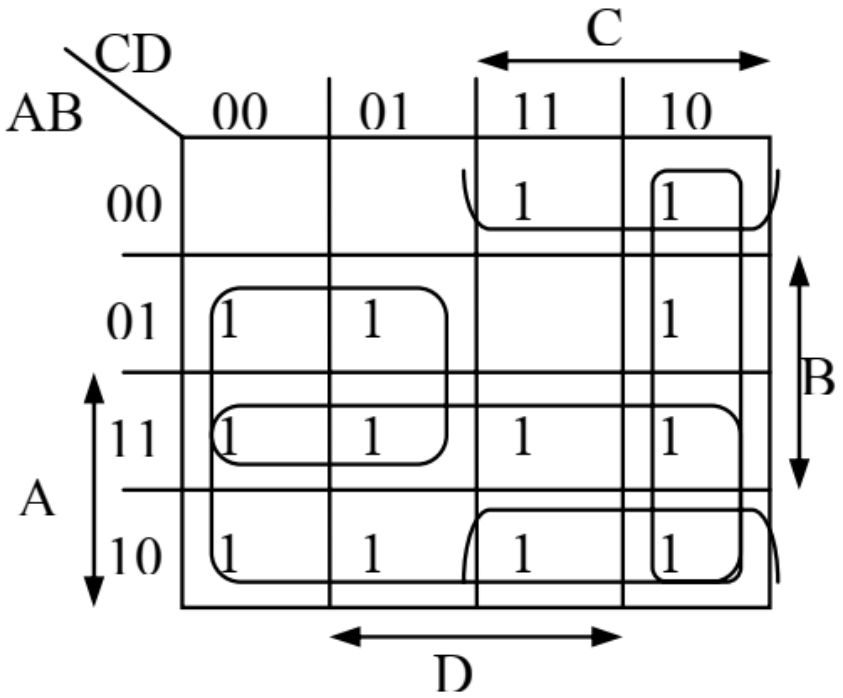


e = AB+AC+B'D'+CD'



f = AB'+AC+A'C'D'+A'BC'+BCD'

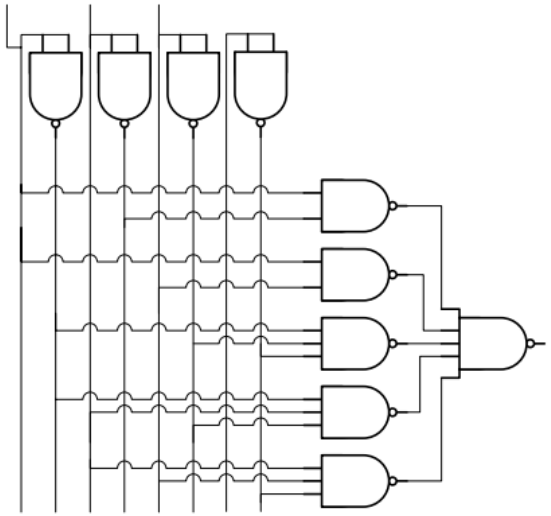
Άσκηση – Αποκωδικοποιητές



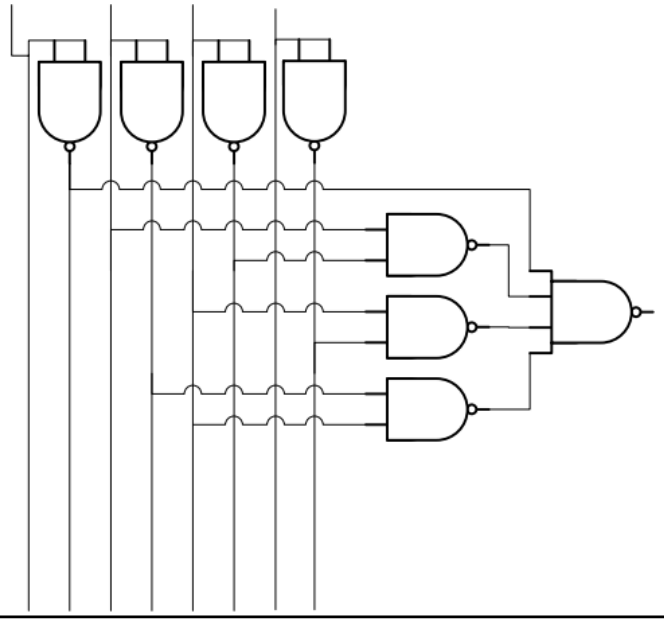
$\mathbf{g} = A+BC'+CD'+B'C$

Άσκηση – Αποκωδικοποιητές

- **Υλοποίηση των συναρτήσεων f και g με πύλες NAND**



f = AB’+AC+A’C’D’+A’BC’+BCD’



g = A+BC’+CD’+B’C

Άσκηση – Άρση Βαρών

- **Στην άρση βαρών μια προσπάθεια ενός αθλητή θεωρείται έγκυρη όταν τη δέχονται τουλάχιστον δύο από τους τρεις κριτές (Κ1,Κ2,Κ3) και άκυρη διαφορετικά. Σχεδιάστε συνδυαστικό κύκλωμα, χρησιμοποιώντας μόνο πύλες NOR, το οποίο να δίνει έξοδο (Χ) ίση με λογικό ‘1’ όταν μια προσπάθεια θεωρείται άκυρη.**
- **Συμβολίστε με λογικό ‘1’ την αποδοχή της προσπάθειας από έναν κριτή και με λογικό ‘0’ το αντίθετο.**

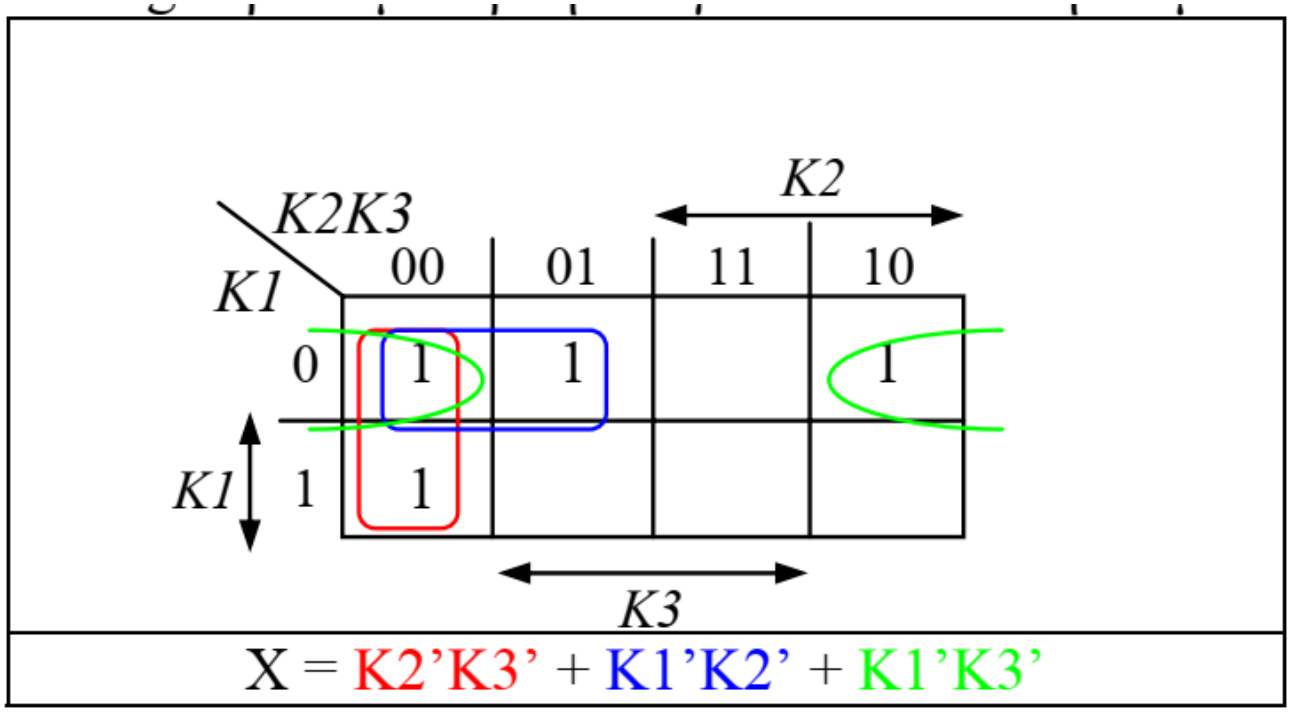
Άσκηση – Άρση Βαρών

- **Είσοδοι: ένα bit το «κουμπί» του κάθε κριτή (K1,K2,K3)**
- **Έξοδος: X, λογικό ‘1’ όταν μια προσπάθεια θεωρείται άκυρη.**
- **Πίνακας αλήθειας:**

K1	K2	K3	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

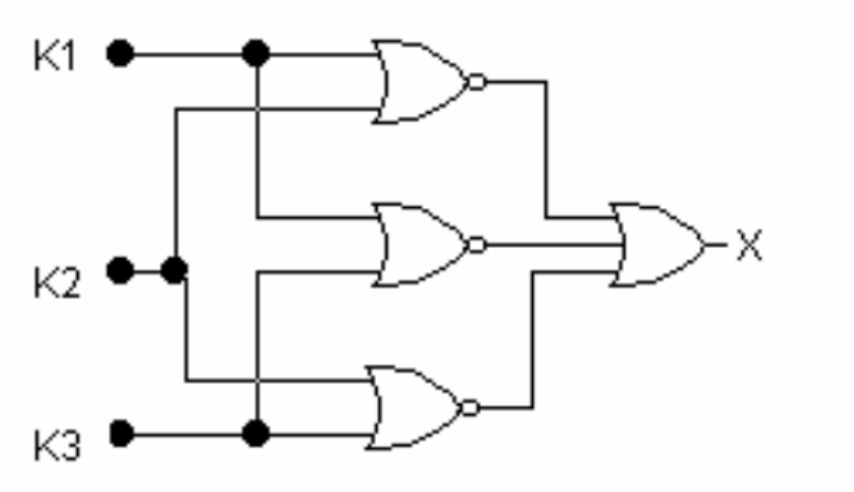
Άσκηση – Άρση Βαρών

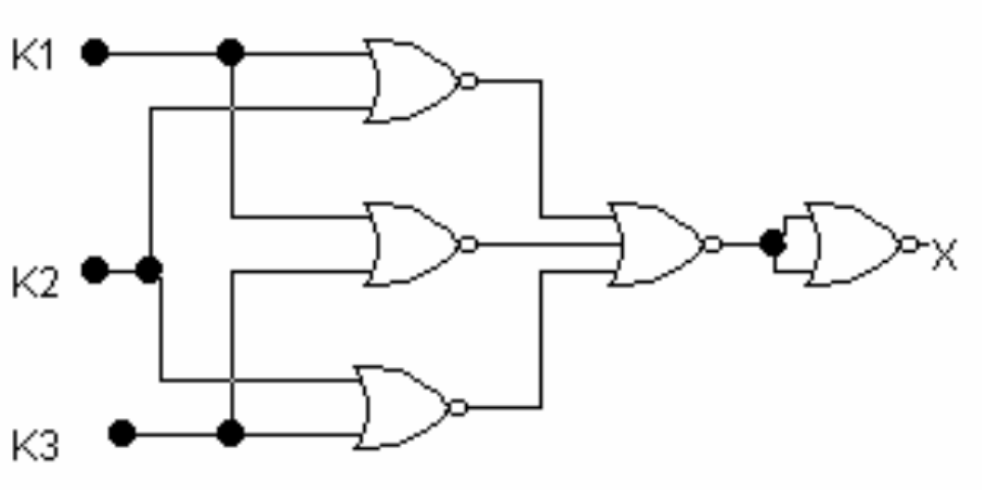
- **1 χάρτη Karnaugh τριών μεταβλητών για να απλοποιήσουμε την έξοδο X**



Άσκηση – Άρση Βαρών

- **X = (K2+K3)’ + (K1+K2)’ + (K1+K3)’**





Άσκηση – Μετατροπής Αριθμών

- Ένα συνδυαστικό κύκλωμα δέχεται στην είσοδο ένα τετραψήφιο δυαδικό αριθμό ($X=X_3X_2X_1X_0$) και δίνει στην έξοδο έναν δυαδικό αριθμό (Y) ίσο με Y=X-10.
  - Π.χ. για τον αριθμό $X=12_{10}=1100_2$, η έξοδος πρέπει να δίνει $Y=X-10=12-10=2_{10}$.
- Όταν η έξοδος έχει αρνητική τιμή δεν μας ενδιαφέρει να εμφανίζεται η σωστή ένδειξη
- Α. Καθορίστε τις εισόδους και τις εξόδους του κυκλώματος και κατασκευάστε τον πίνακα αλήθειας.
- Β. Απλοποιείστε τις συναρτήσεις εξόδου του κυκλώματος χρησιμοποιώντας χάρτες Karnaugh.
- Γ. Υλοποιήστε το κύκλωμα χρησιμοποιώντας μόνο πύλες NAND.
- Δ. Βρείτε τις απλοποιημένες συναρτήσεις εξόδου που προκύπτουν χωρίς την εκμετάλλευση των αδιάφορων όρων.

Άσκηση – Μετατροπής Αριθμών

- **Πρώτο βήμα καθορισμός εισόδων και εξόδων**
- **Είσοδοι: 4 bits**
- **Έξοδοι: Μεγαλύτερη δεκαδική τιμή της εξόδου → $X=15_{10}$ (μεγαλύτερη είσοδος) → $Y=X-10=15-10=5_{10}=10_2$ → 3 bits**
- **Άρα:**
  - **Είσοδος ($X=X_3X_2X_1X_0$)**
  - **Έξοδος ($Y=Y_2Y_1Y_0$)**

Άσκηση – Μετατροπής Αριθμών

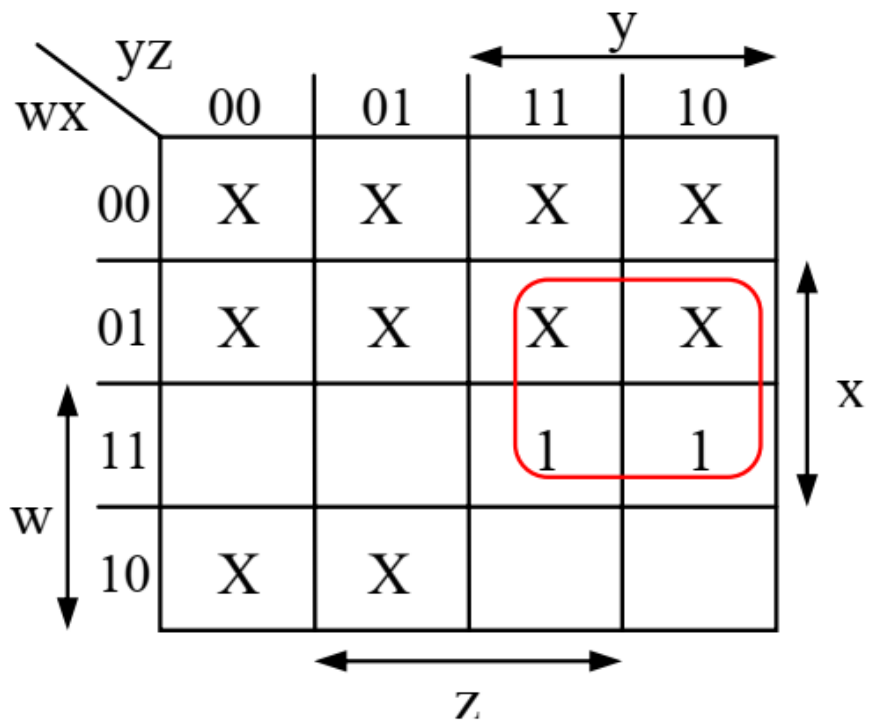
- **Κατασκευή Πίνακα**
- **Όταν η έξοδος έχει αρνητική τιμή δεν μας ενδιαφέρει να εμφανίζεται η σωστή ένδειξη → αδιάφοροι όροι (X) στις αντίστοιχες θέσεις**

Δυαδική Είσοδος				Δεκαδικοί Αριθμοί		Δυαδική Έξοδος		
X_3	X_2	X_1	X_0	X	Y	Y_2	Y_1	Y_0
0	0	0	0	0	-10	X	X	X
0	0	0	1	1	-9	X	X	X
0	0	1	0	2	-8	X	X	X
0	0	1	1	3	-7	X	X	X
0	1	0	0	4	-6	X	X	X
0	1	0	1	5	-5	X	X	X
0	1	1	0	6	-4	X	X	X
0	1	1	1	7	-3	X	X	X
1	0	0	0	8	-2	X	X	X
1	0	0	1	9	-1	X	X	X
1	0	1	0	10	0	0	0	0
1	0	1	1	11	1	0	0	1
1	1	0	0	12	2	0	1	0
1	1	0	1	13	3	0	1	1
1	1	1	0	14	4	1	0	0
1	1	1	1	15	5	1	0	1

Άσκηση – Μετατροπής Αριθμών

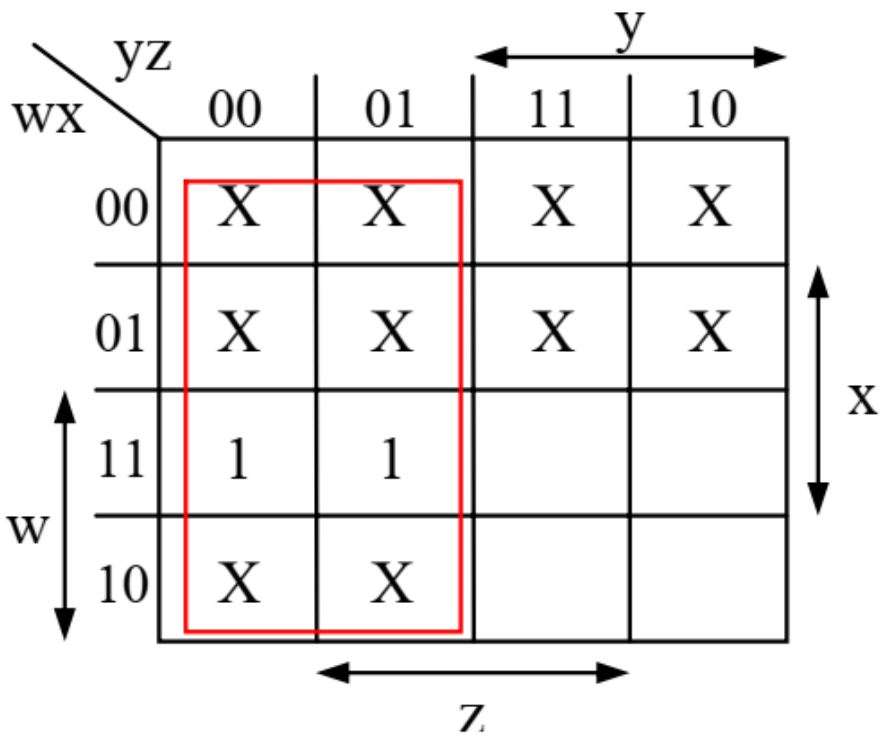
- **Φτιάχνουμε τους χάρτες Karnaugh για κάθε μεταβλητή εξόδου**
- **Προσέχουμε την χρήση των αδιάφορων όρων**
- **Για ομοιομορφία: (w,x,y,z) → ($X_3X_2X_1X_0$)**

Άσκηση – Μετατροπής Αριθμών



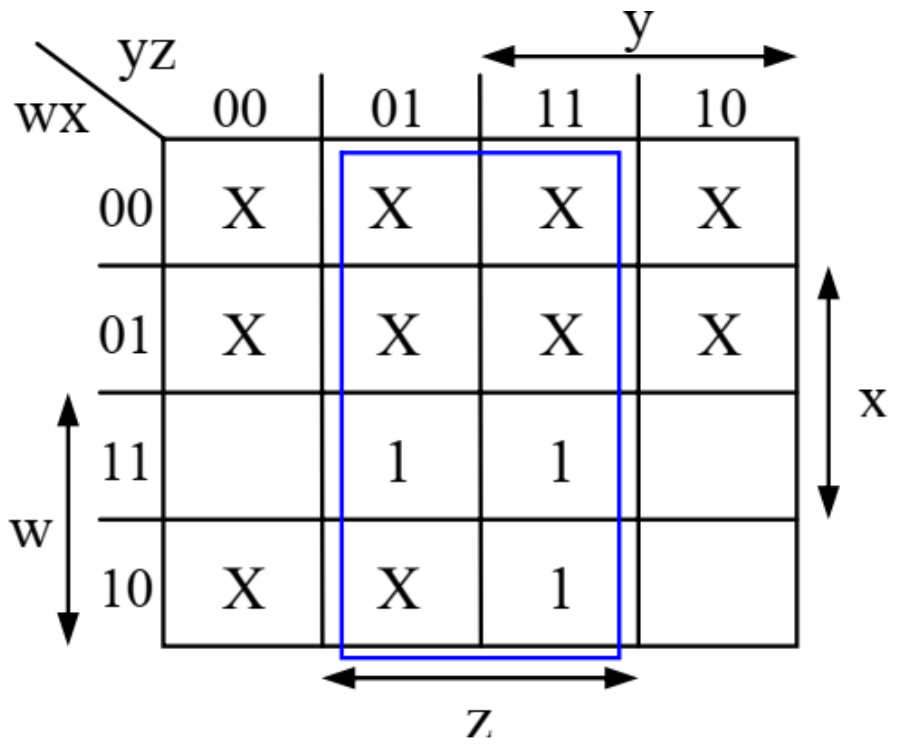
$$Y_2(w,x,y,z) = x\,y$$

Άσκηση – Μετατροπής Αριθμών



$$Y_1(w,x,y,z) = y'$$

Άσκηση – Μετατροπής Αριθμών



$$Y_0(w,x,y,z) = z$$

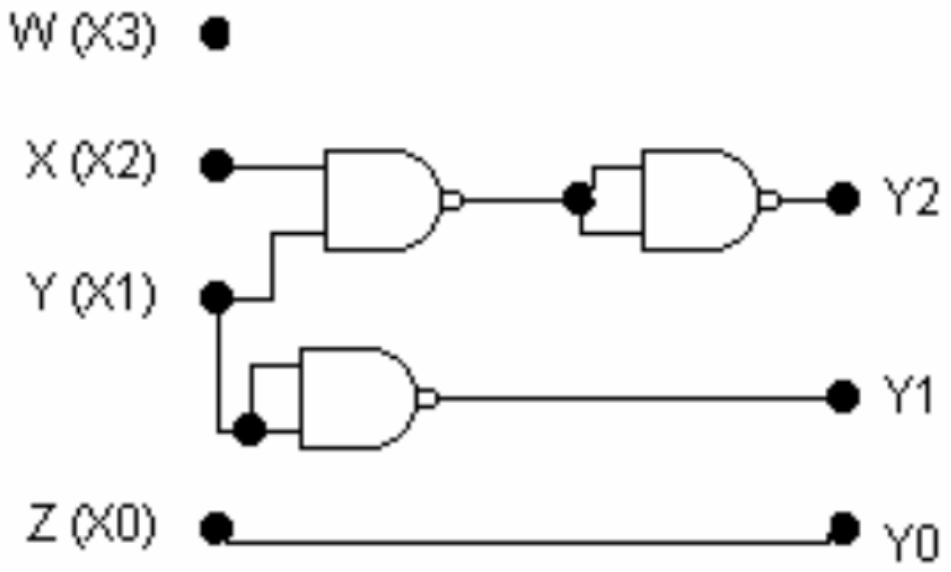
Άσκηση – Μετατροπής Αριθμών

- **Για την υλοποίηση του κυκλώματος μόνο με πύλες NAND οι συναρτήσεις των εξόδων πρέπει να τροποποιηθούν ως εξής:**

Y2 = xy = [(xy)']' = [(xy)' (xy)']'

Y1 = y' = (yy)'

Y0 = z



Άσκηση – Μετατροπής Αριθμών

- **Χωρίς την εκμετάλλευση των αδιάφορων όρων προκύπτουν οι παρακάτω απλοποιημένες συναρτήσεις των εξόδων:**

Y2 = wxy

Y1 = wxy’

Y0 = wxz + wyz

Πολυπλέκτης (Multiplexer)

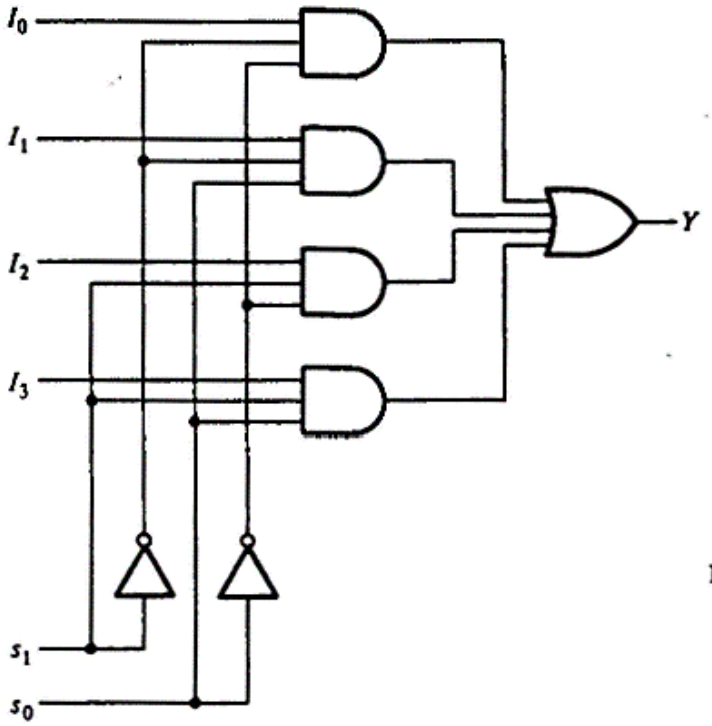
- **Ο πολυπλέκτης είναι ένα συνδυαστικό κύκλωμα που επιλέγει δυαδικές πληροφορίες ανάμεσα σε πολλές γραμμές εισόδου και τις κατευθύνει σε μία γραμμή εξόδου.**

- **Η επιλογή της μιας συγκεκριμένης γραμμής εισόδου γίνεται μέσω μερικών γραμμών επιλογής.**

- **Ένας πολυπλέκτης 2^n-σε-1 γραμμή κατασκευάζεται από έναν αποκωδικοποιητή n-σε-2^n προσθέτοντας σε αυτόν 2^n εισόδους μια για κάθε πύλη AND. Οι έξοδοι των πυλών AND εφαρμόζονται σε μια μοναδική πύλη OR για να δώσουν τη μία γραμμή εξόδου.**

Παράδειγμα

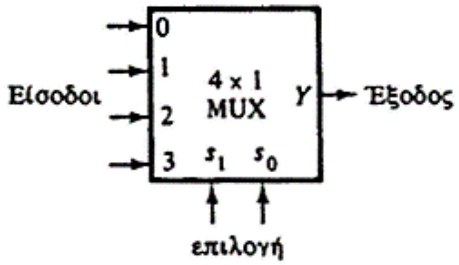
Πολυπλέκτης 4-σε-1



(α) Λογικό διάγραμμα

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

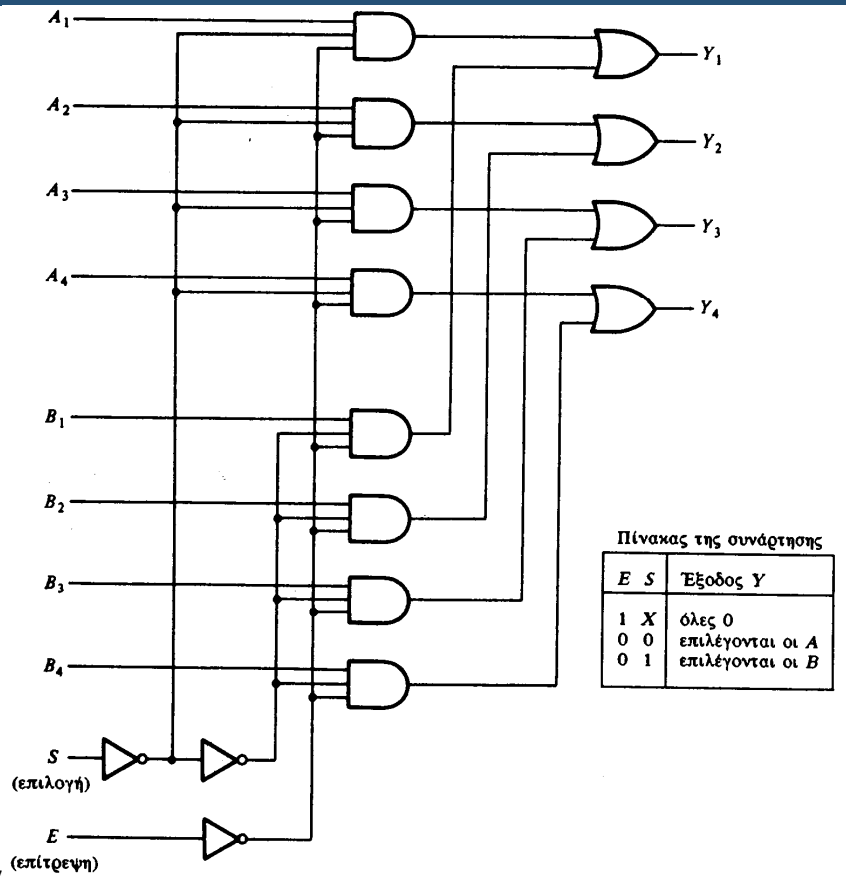
(β) Πίνακας της συνάρτησης



(γ) Σχηματικό διάγραμμα

Πολυπλέκτες με Κοινή Είσοδο Επίτρεψης

Η είσοδος Επίτρεψης (E) τοποθετείται για λόγους επέκτασης.



Πίνακας της συνάρτησης

E	S	Έξοδος Y
1	X	όλες 0
0	0	επιλέγονται οι A
0	1	επιλέγονται οι B

Υλοποίηση Συνάρτησης με Πολυπλέκτη

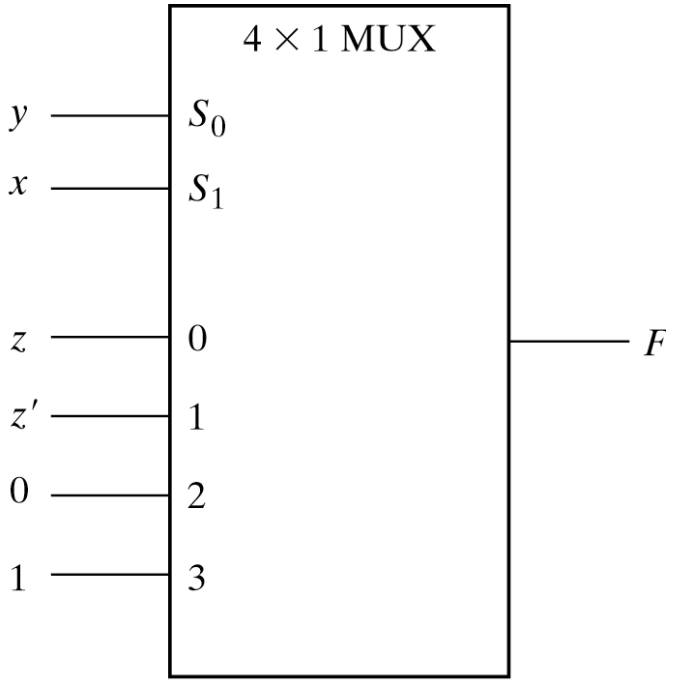
- **Κάθε πολυπλέκτης 2^n σε 1 μπορεί να υλοποιήσει οποιαδήποτε συνάρτηση n+1 μεταβλητών ως εξής:**
  1. **Βάζουμε τις n μεταβλητές στις εισόδους επιλογής.**
  2. **Χρησιμοποιούμε την τελευταία μεταβλητή για τις εισόδους.**

Παράδειγμα

F(x,y,z) = Σ(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	

(a) Truth table



(b) Multiplexer implementation

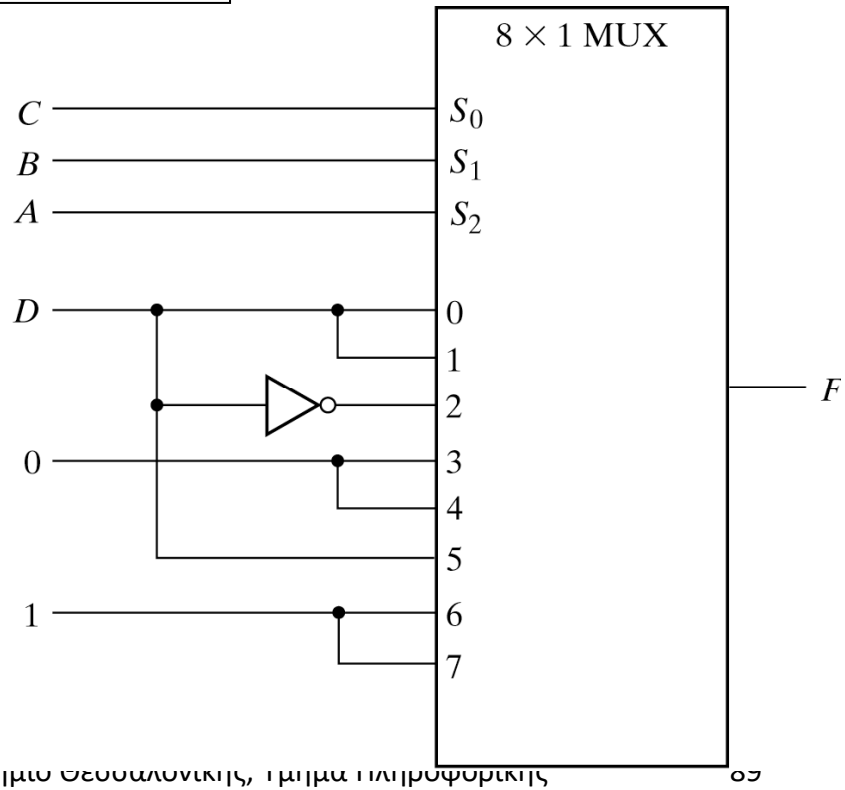
Υλοποίηση Συναρτήσεων Boole

- **Αλγόριθμος υλοποίησης συνάρτησης n μεταβλητών με χρήση πολυπλέκτη:**
  1. **Χρησιμοποιούμε πολυπλέκτη 2^{n-1}-σε-1 με n-1 γραμμές επιλογής.**
  2. **Δημιουργούμε τον πίνακα αλήθειας της συνάρτησης.**
  3. **Συνδέουμε τις n - 1 περισσότερο σημαντικές μεταβλητές στις γραμμές επιλογής και κρατάμε τη δεξιότερη (λιγότερο σημαντική).**
  4. **Για κάθε συνδυασμό των μεταβλητών των γραμμών επιλογής, εκφράζουμε την έξοδο ως συνάρτηση της τελευταίας μεταβλητής.**

Παράδειγμα

F(A,B,C,D) = Σ(1,3,4,11,12,13,14,15)

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	



Στοιχεία τριών καταστάσεων

- Ο Α και ο Β θέλουν να διαμοιραστούν την ίδια αρτηρία. Ο Α και ο Β απαγορεύεται να οδηγήσουν μαζί την αρτηρία. Πως μπορώ να αποκλείσω αυτή τη πιθανότητα ?
- Με ένα πολυπλέκτη επιτρέπω είτε στον Α είτε στο Β να οδηγούν την αρτηρία !
- Καλή λύση, αν εκ των προτέρων ξέρω πόσοι και ποιοι θα διαμοιράζονται την αρτηρία !
- Ξέρει ο κατασκευαστής του PC σας πόσους δίσκους θα βάλετε ? Πόσα περιφερειακά θα προσθέσετε στην αρτηρία USB ?

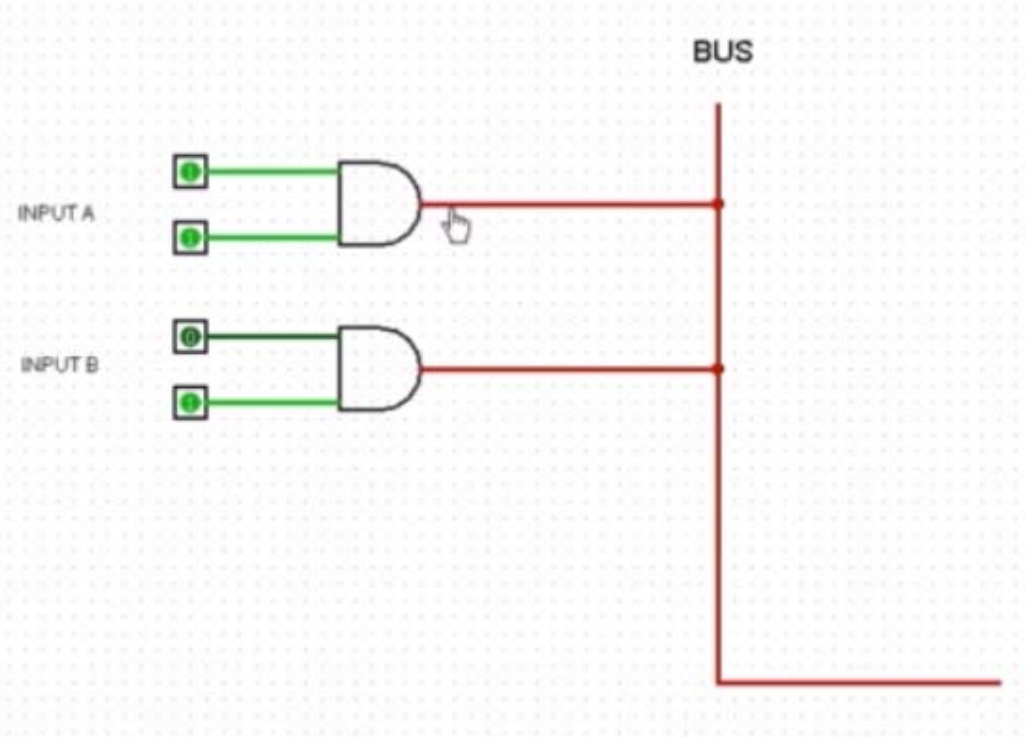
Στοιχεία τριών καταστάσεων

- **Τα στοιχεία τριών καταστάσεων μας επιτρέπουν να προσθέτουμε συσκευές σε μια αρτηρία, προσφέροντάς μας τη δυνατότητα να αποκόπτουμε τη συσκευή μας από την αρτηρία στους χρονικούς κύκλους που δε της επιτρέπεται να οδηγήσει την αρτηρία.**

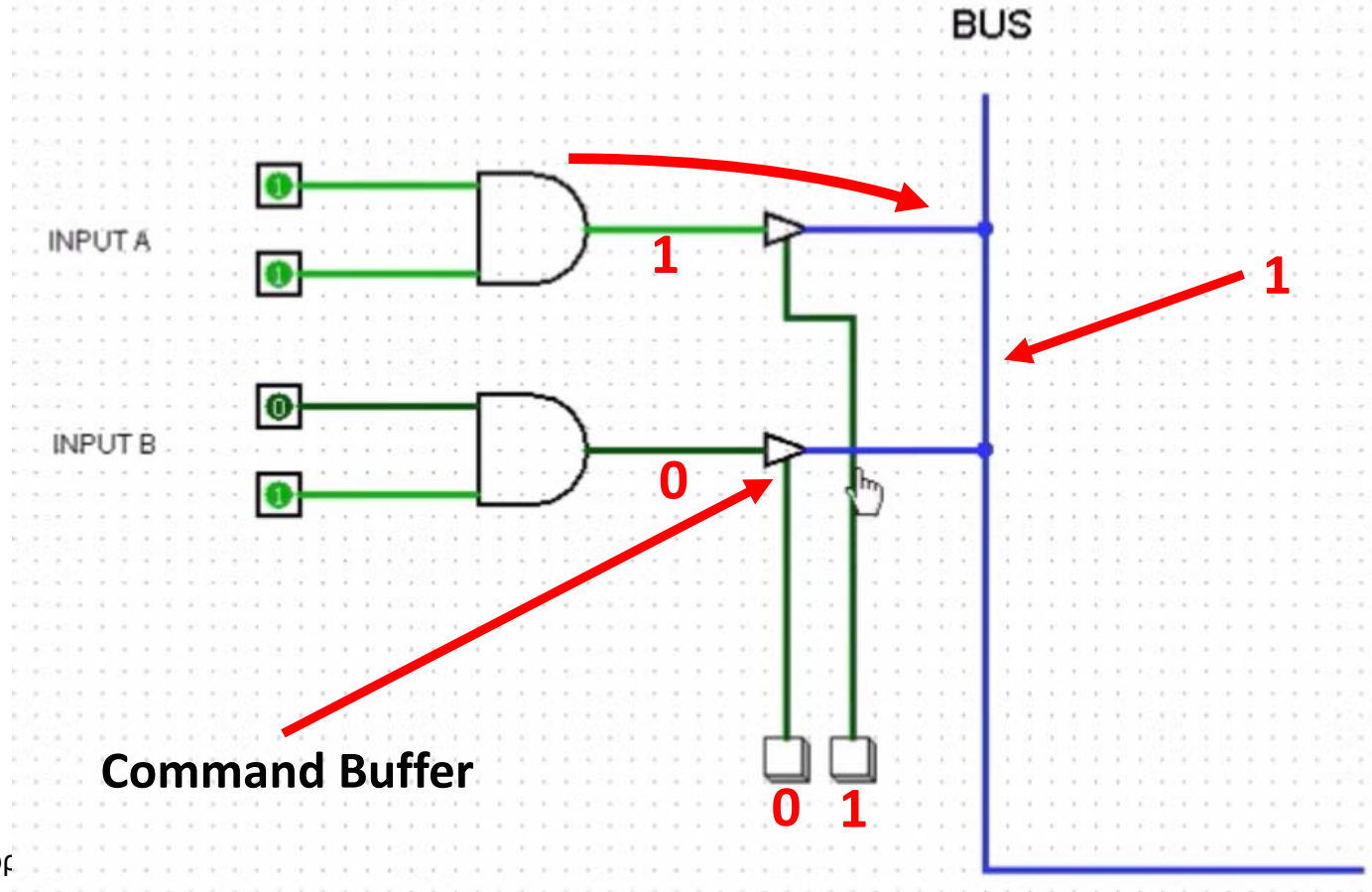
- **Στο Logisim το αντίστοιχο στοιχείο αναφέρεται ως control buffer**
  - **Θα κάνετε εργαστηριακή άσκηση στο επόμενο εργαστήριο**

Στοιχεία τριών καταστάσεων

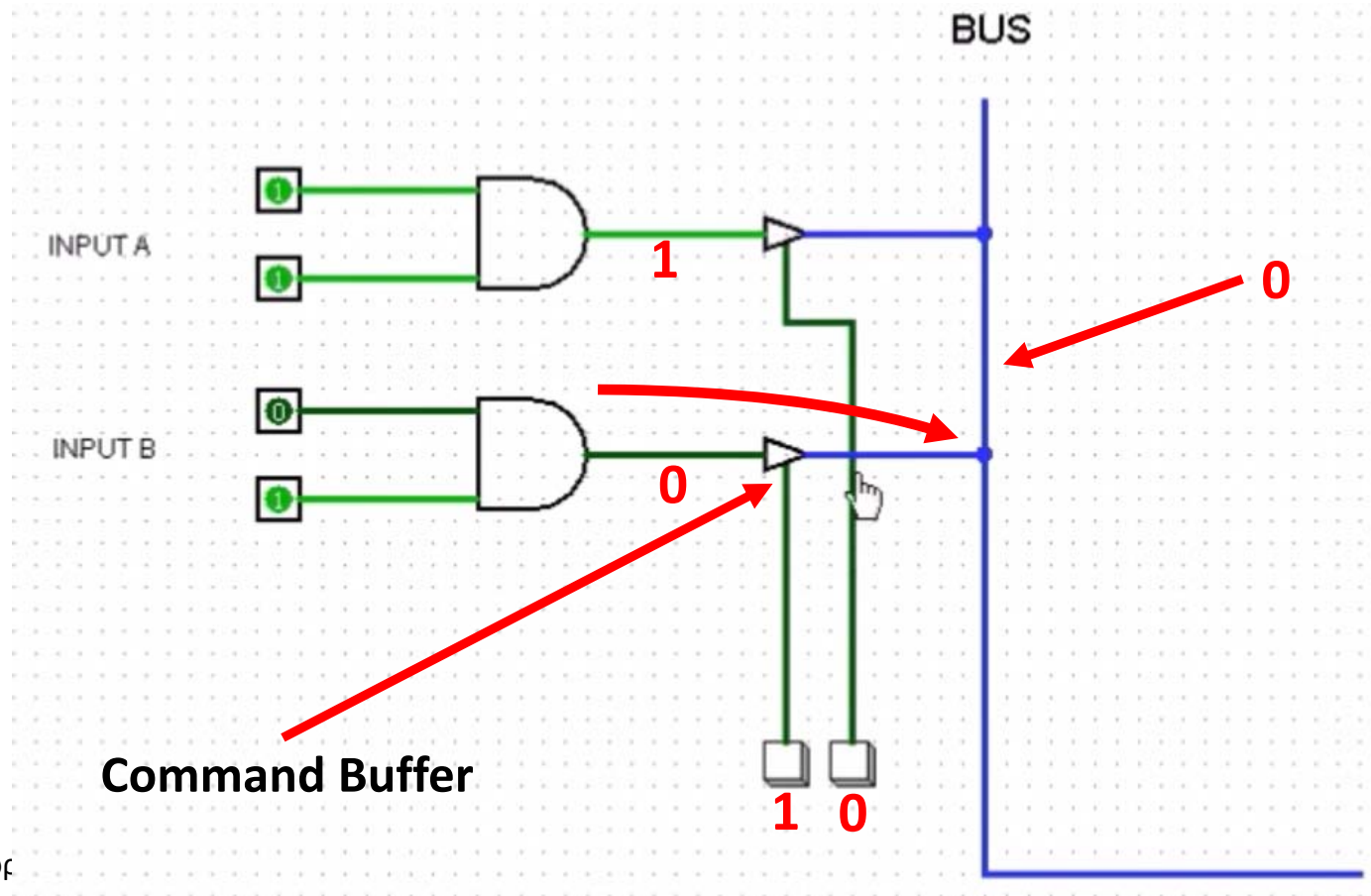
- **Το bus έχει απροσδιόριστη τιμή**



Στοιχεία τριών καταστάσεων

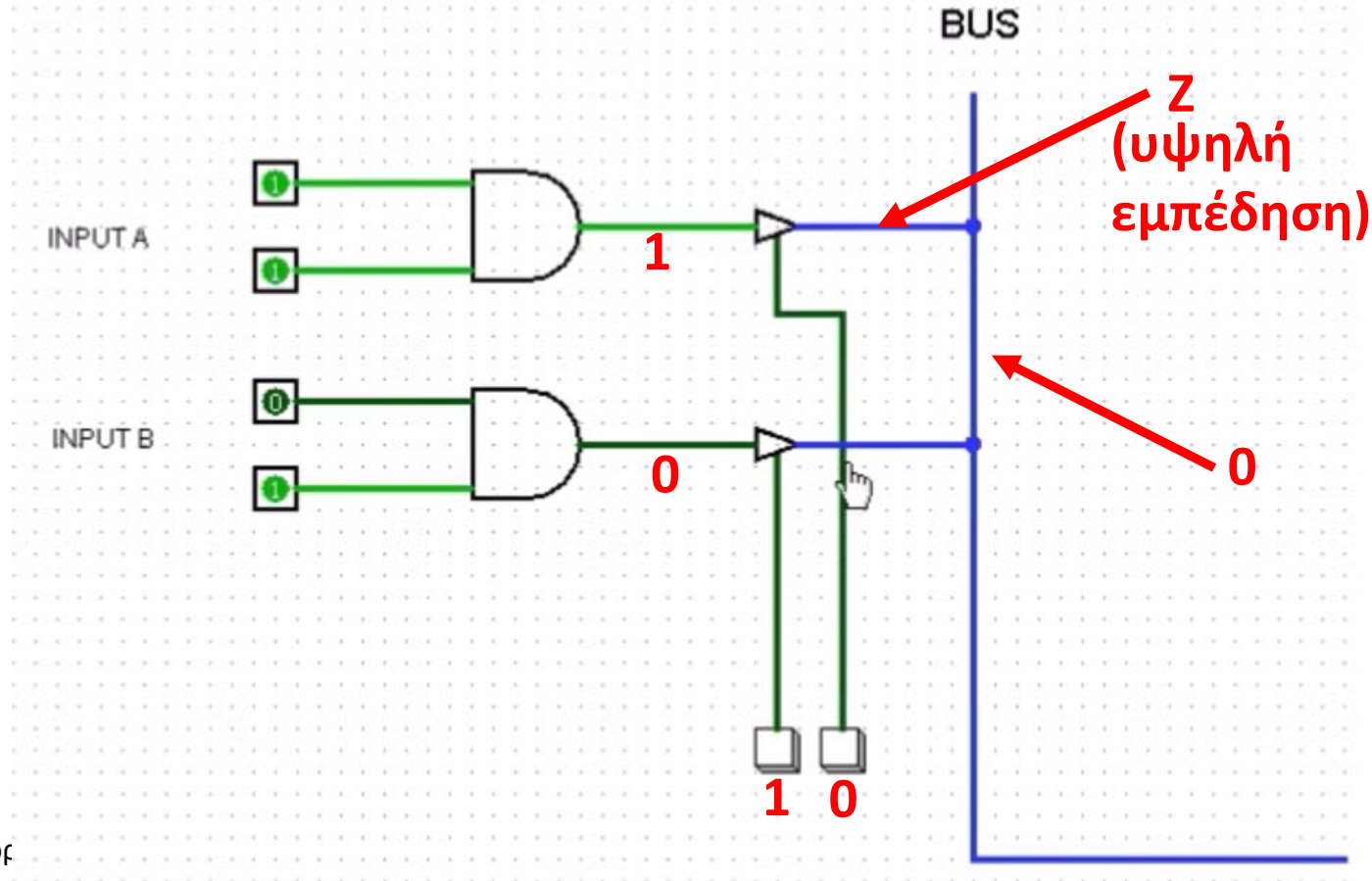


Στοιχεία τριών καταστάσεων



Στοιχεία τριών καταστάσεων

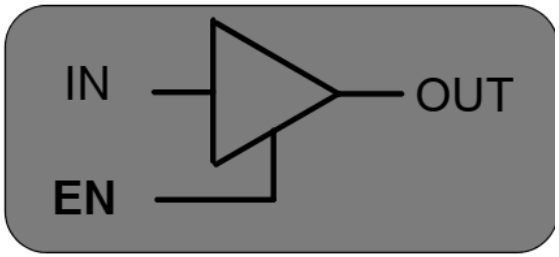
- Μπορούμε να συνδέσουμε πολλαπλές εξόδους/κυκλώματα σε ένα bus
- Υψηλή εμπέδηση (Z)



Στοιχεία τριών καταστάσεων

- Ουσιαστικά στην έξοδο έχουμε μία τρίτη κατάσταση (υψηλή εμπέδηση-high impendence-hiZ) (εκτός από το 0 και 1)
- Στην hiZ κατάσταση η πύλη/καλώδιο δεν επιδρά στην έξοδο. Είναι ανοικτοκύκλωμα (σαν να μην υπάρχει καλώδιο εξόδου στην πύλη)
  - Μπορούμε να συνδέσουμε εξόδους πολλών πυλών ταυτόχρονα εάν δεν έχουμε περισσότερες από μία σε κατάσταση διαφορετική από hiZ

Σύμβολο



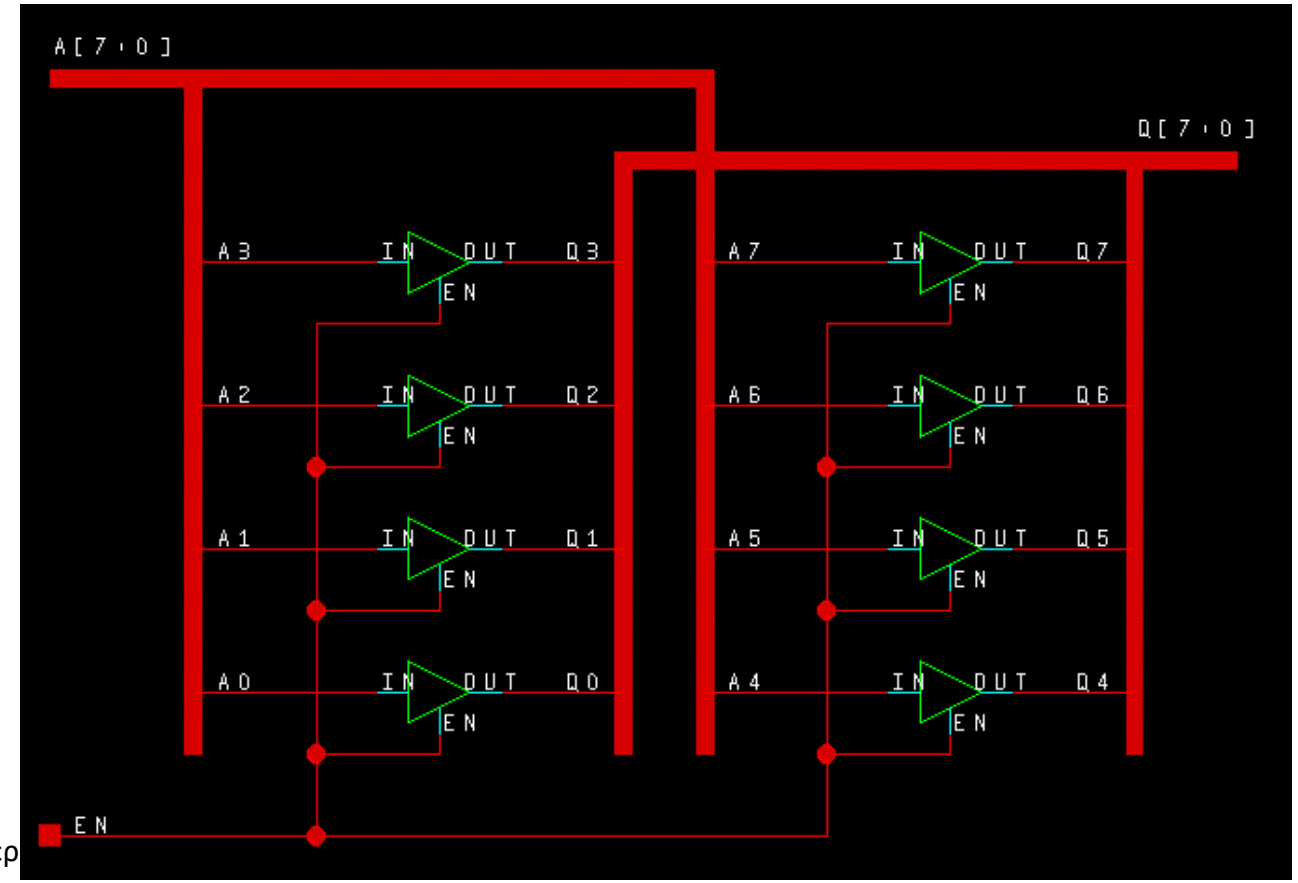
Πίνακας Αληθείας

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

1

Στοιχεία τριών καταστάσεων

In	En	Out
0	0	Z
1	0	Z
0	1	0
1	1	1



Άσκηση – Εκφώνηση

- **α. Να απλοποιήσετε τις συναρτήσεις: f() = x’y’z + x’yz + y’z και g() = xy’z + xz + xyz′**
- **β. Να υπολογίσετε τις αντίστροφες συναρτήσεις f ′ και g′, χρησιμοποιώντας τις απλοποιημένες εκφράσεις που προέκυψαν στο βήμα α**
- **γ. Να γραφούν οι f και g ως άθροισμα ελαχιστόρων**
- **δ. Αποδείξτε ότι ισχύει f f ′ = 0 και g g ′ = 0 , χρησιμοποιώντας τις απλοποιημένες εκφράσεις των συναρτήσεων f , g , f ′ και g′ που προέκυψαν στα βήματα α και β**

Άσκηση – Λύση (α)

Ερώτημα α

HINT:
Θεώρημα της απορρόφησης:
y’ + x’y = y’ + x’

$$f = x'y'z + x'yz + y'z = (x' + 1)\,y'z + x'yz =$$
$$= y'z + x'yz = (y' + x'y)\,z = (y' + x')\,z = y'z + x'z$$

$$g = xy'z + xz + xyz' = (y' + 1)\,zx + xyz' = zx + xyz' =$$
$$= (z + yz')\,x = x\,(z + y) = xz + xy$$

Άσκηση – Λύση (β)

Ερώτημα β

Εφαρμόζουμε το Θεώρημα του DeMorgan

$$f' = (y'z + zx')' = (y + z')(z' + x) = yz' + xy + z'z' + z'x =$$
$$= yz' + xy + z' + z'x = z'(y + 1 + x) + xy = z' + xy$$

$$g' = (x(z + y))' = x' + (z + y)' = x' + z'y'$$

Άσκηση – Λύση (γ)

Ερώτημα γ

- **Σε κάθε όρο της απλοποιημένης μορφής εμφανίζουμε ως παράγοντα άθροισμα των αντίθετων μεταβλητών που δεν περιλαμβάνονται στο όρο:**

$$f = y'z + x'z = (x + x')y'z + x'(y + y')z =$$
$$= xy'z + x'y'z + x'y'z + x'yz = \Sigma(1, 3, 5)$$

$$g = xz + xy = \Sigma(5, 6, 7)$$

Άσκηση – Λύση (δ)

Ερώτημα δ

Κάνουμε πράξεις:

$$f'f = (z' + xy)(y'z + x'z) = z'y'z + z'x'z + xyy'z + xyx'z$$

$$= 0 + 0 + 0 + 0 = 0$$

Ομοίως:

$$g'g = 0$$

Άσκηση – Εκφώνηση

- **Δίνεται η συνάρτηση f :**

$$f(n) = \text{trunc}(\sqrt{n})$$

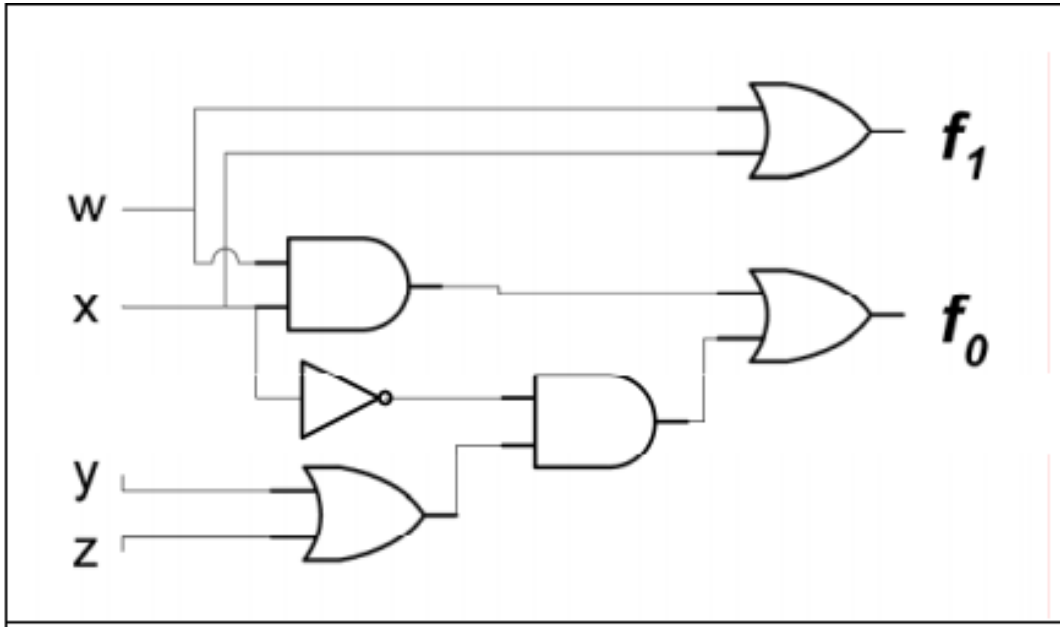
που υπολογίζει το ακέραιο μέρος (= με αποκοπή) της τετραγωνικής ρίζας του ακέραιου θετικού αριθμού n. Το n είναι αριθμός 4 bit.

- **α) Να κατασκευαστεί ο πίνακας αλήθειας της f**
- **β) Να γίνει απλοποίηση με χάρτες Karnaugh**
- **γ) Να γίνει υλοποίηση μόνο με πύλες NAND**
- **δ) Να γίνει υλοποίηση της συνάρτησης f (μόνο το LSB) με έναν πολυπλέκτη "8-σε-1"**

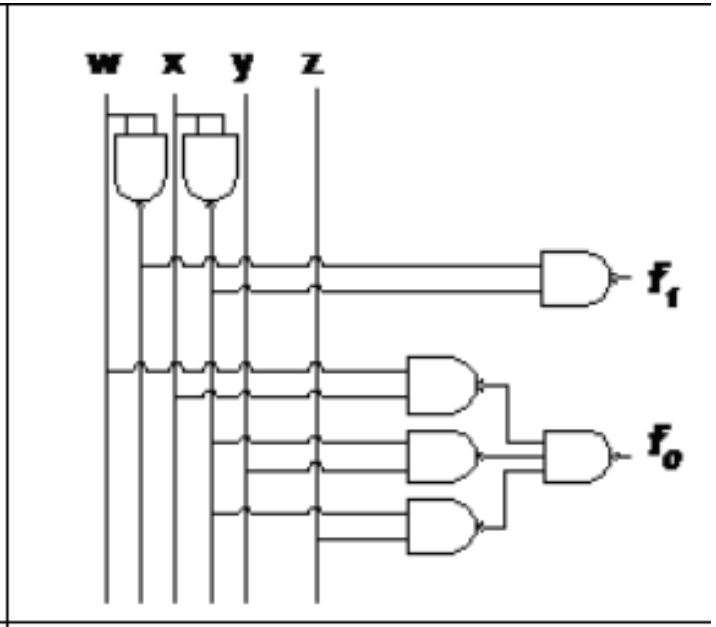
Άσκηση – Λύση

w	x	y	z	n	f(n)= trunc(√n)	f_1	f_0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	1
0	0	1	0	2	1	0	1
0	0	1	1	3	1	0	1
0	1	0	0	4	2	1	0
0	1	0	1	5	2	1	0
0	1	1	0	6	2	1	0
0	1	1	1	7	2	1	0
1	0	0	0	8	2	1	0
1	0	0	1	9	3	1	1
1	0	1	0	10	3	1	1
1	0	1	1	11	3	1	1
1	1	0	0	12	3	1	1
1	1	0	1	13	3	1	1
1	1	1	0	14	3	1	1
1	1	1	1	15	3	1	1

Άσκηση – Λύση



$f_1 = w + x$ $f_0 = wx + x'(y+z)$

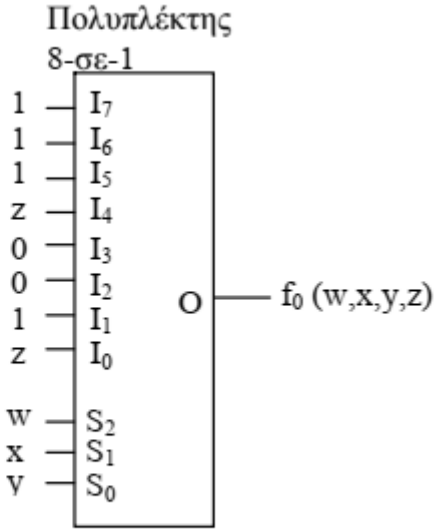


$f_1 = w + x$ $f_0 = wx + x'y + x'z$

Άσκηση – Λύση

- **Τα 3 MSB της εισόδου (w, x, y,) χρησιμοποιούνται σαν μεταβλητές επιλογής (γραμμές select) και στις γραμμές εισόδου I0-I7 συνδέουμε είτε λογικό 0, είτε 1, είτε z, είτε z', συγκρίνοντας τις στήλες f0 και z, ανά 2 γραμμές κάθε φορά.**

w	x	y	z	f_0	Είσοδοι
0	0	0	0	0	I_0=z
0	0	0	1	1	
0	0	1	0	1	I_1=1
0	0	1	1	1	
0	1	0	0	0	I_2=0
0	1	0	1	0	
0	1	1	0	0	I_3=0
0	1	1	1	0	
1	0	0	0	0	I_4=z
1	0	0	1	1	
1	0	1	0	1	I_5=1
1	0	1	1	1	
1	1	0	0	1	I_6=1
1	1	0	1	1	
1	1	1	0	1	I_7=1
1	1	1	1	1	



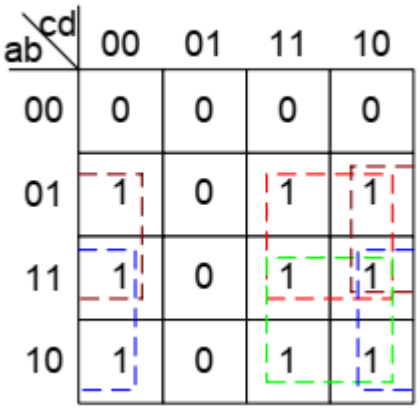
Άσκηση – Εκφώνηση

- **Να υλοποιηθεί η συνάρτηση**

F(a, b, c, d) = Σ(4, 6, 7, 8, 10, 11, 12, 14, 15)

χρησιμοποιώντας μόνο αποκωδικοποιητές 2-σε-4 (με κανονικές εξόδους και χωρίς είσοδο επίτρεψης)

Άσκηση – Λύση



- **Κάθε αποκωδικοποιητής 2-σε-4 υλοποιεί όλους τους ελαχιστόρους 2 μεταβλητών. Με χρήση πυλών τύπου OR μπορούμε να υλοποιήσουμε οποιαδήποτε συνάρτηση 2 μεταβλητών.**
- **Στόχος: να αναλύσουμε την συνάρτηση 4 μεταβλητών σε συναρτήσεις 2 μεταβλητών, έτσι ώστε κάθε μία από αυτές να υλοποιηθεί από έναν αποκωδικοποιητή.**
- **Από τον χάρτη Karnaugh παιρνουμε:**

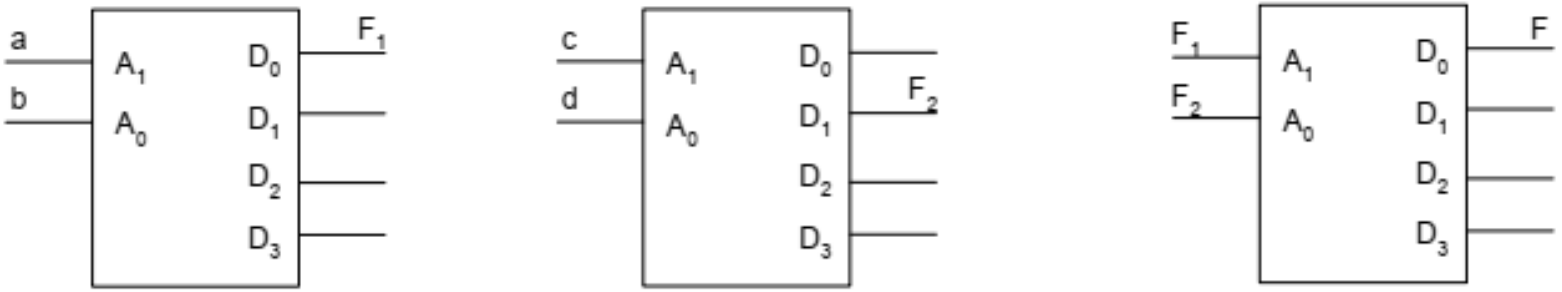
F = bc+ac+bd′+ad′ = (a+b)c+(a+b)d′ = (a+b)(c+d′)

Άσκηση – Λύση

- **Με εφαρμογή του θεωρήματος DeMorgan:**

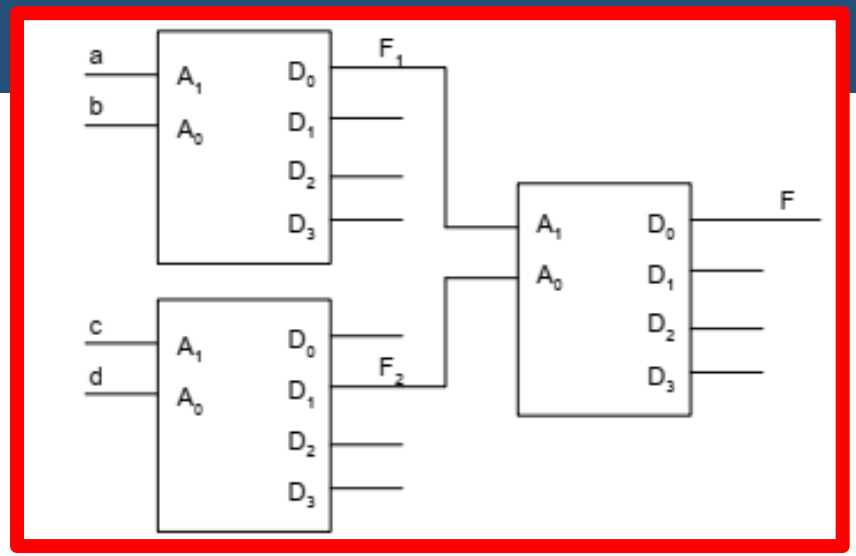
F = (a+b)(c+d′) = (a′b′)′(c′d)′=F1′F2′, όπου F1= a′b′ και F2= c′d.

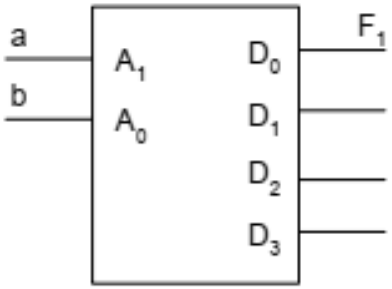
- **Η F1 είναι ο ελαχιστόρος 0, και υλοποιείται από την έξοδο D0**
- **Η F2 είναι ο ελαχιστόρος 1, και υλοποιείται από την έξοδο D1**
- **Ο όρος F1′F2′ είναι ουσιαστικά ο ελαχιστόρος 0, και μπορεί να υλοποιηθεί από την έξοδο D0**

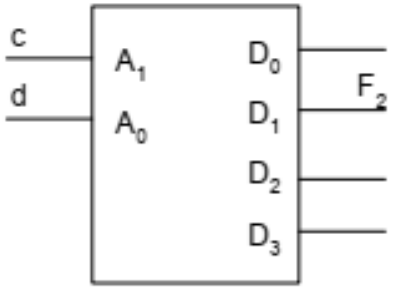


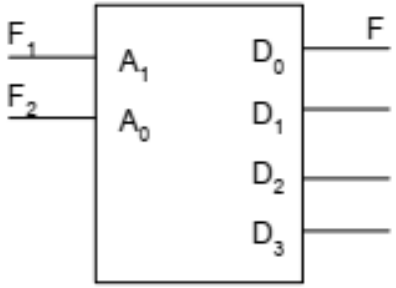
Άσκηση – Λύση

- **Τελικό κύκλωμα**









Άσκηση – Εκφώνηση

- **Σχεδιάστε ένα συνδυαστικό κύκλωμα που δέχεται δύο μη προσημασμένους δυαδικούς αριθμούς Α και Β των δύο bit ο κάθε ένας, $A=a_1a_0$ και $B=b_1b_0$, και εκτελεί την πράξη A^B. Υλοποιήστε το κύκλωμα χρησιμοποιώντας αποκωδικοποιητές 3 σε 8 με κανονικές εξόδους και είσοδο επίτρεψης ενεργή στο λογικό 1, καθώς και οποιαδήποτε πύλη θέλετε. (Υπενθύμιση: $0^0=1$).**

Άσκηση - Λύση

- Αποτέλεσμα → 5 bits (μεγαλύτερο δυνατό αποτέλεσμα: 3^3=27)
- Θα πρέπει να υλοποιήσουμε 5 συναρτήσεις με αποκωδικοποιητές 3 σε 8 (έχοντας ως είσοδο 4 bits)
- → Άρα χρειαζόμαστε 2 αποκωδικοποιητές
- Ο ένας αποκωδικοποιητής θα υλοποιεί τους ελαχιστόρους 0…7, και ο άλλος τους ελαχιστόρους 8…15.

a_1 a_0	b_1 b_0	p_4 p_3 p_2 p_1 p_0
0 0	0 0	0 0 0 0 1
0 0	0 1	0 0 0 0 0
0 0	1 0	0 0 0 0 0
0 0	1 1	0 0 0 0 0
0 1	0 0	0 0 0 0 1
0 1	0 1	0 0 0 0 1
0 1	1 0	0 0 0 0 1
0 1	1 1	0 0 0 0 1
1 0	0 0	0 0 0 0 1
1 0	0 1	0 0 0 1 0
1 0	1 0	0 0 1 0 0
1 0	1 1	0 1 0 0 0
1 1	0 0	0 0 0 0 1
1 1	0 1	0 0 0 1 1
1 1	1 0	0 1 0 0 1
1 1	1 1	1 1 0 1 1

Άσκηση - Λύση

- Οι παραπάνω συναρτήσεις υλοποιούνται ως αθροίσματα ελαχιστόρων:
  - p4=Σ(15)
  - p3=Σ(11,14,15)
  - p2=Σ(10)
  - p1=Σ(9,13,15)
  - p0=Σ(0,4,5,6,7,8,12,13,14,15)

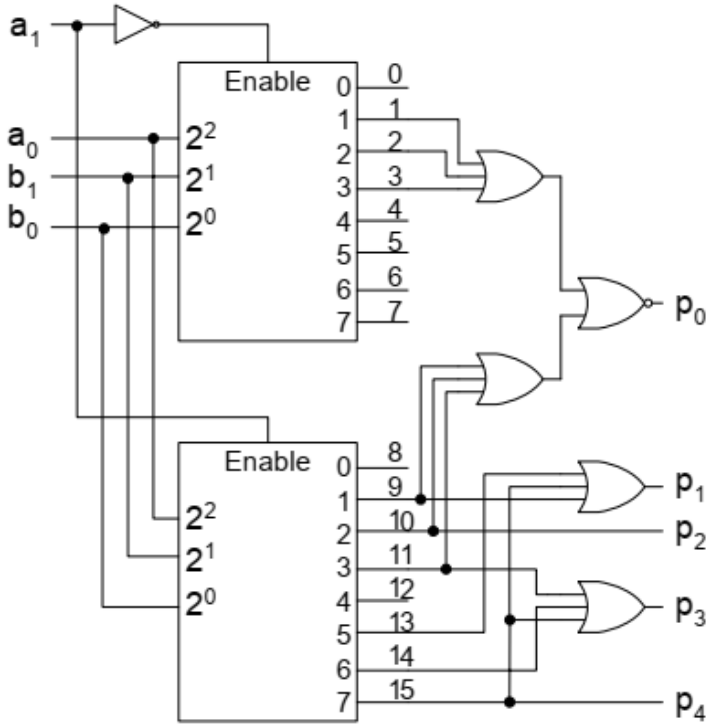
- Επίσης, προκειμένου να μειώσω το υλικό, παρατηρώ ότι:
  - p0=(Σ(1,2,3,9,10,11))΄

a_1 a_0	b_1 b_0	p_4 p_3 p_2 p_1 p_0
0 0	0 0	0 0 0 0 1
0 0	0 1	0 0 0 0 0
0 0	1 0	0 0 0 0 0
0 0	1 1	0 0 0 0 0
0 1	0 0	0 0 0 0 1
0 1	0 1	0 0 0 0 1
0 1	1 0	0 0 0 0 1
0 1	1 1	0 0 0 0 1
1 0	0 0	0 0 0 0 1
1 0	0 1	0 0 0 1 0
1 0	1 0	0 0 1 0 0
1 0	1 1	0 1 0 0 0
1 1	0 0	0 0 0 0 1
1 1	0 1	0 0 0 1 1
1 1	1 0	0 1 0 0 1
1 1	1 1	1 1 0 1 1

Άσκηση - Λύση

- **Οι παραπάνω συναρτήσεις υλοποιούνται ως αθροίσματα ελαχιστόρων:**
  - **p4=Σ(15)**
  - **p3=Σ(11,14,15)**
  - **p2=Σ(10)**
  - **p1=Σ(9,13,15)**
  - **p0=Σ(0,4,5,6,7,8,12,13,14,15)**

- **Επίσης, προκειμένου να μειώσω το υλικό, παρατηρώ ότι:**
  - **p0=(Σ(1,2,3,9,10,11))΄**

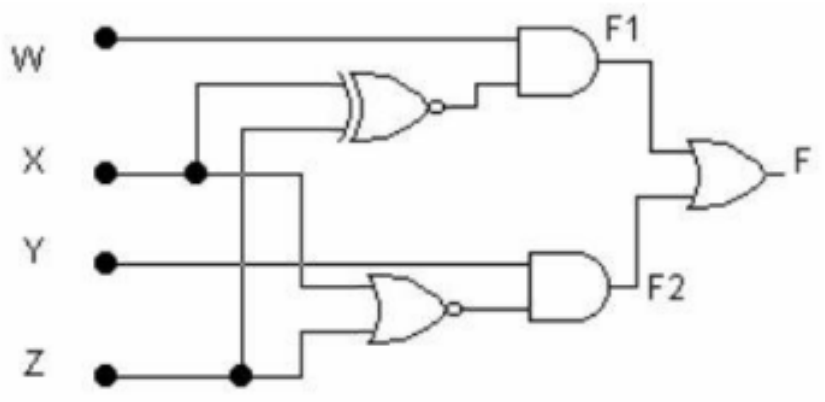


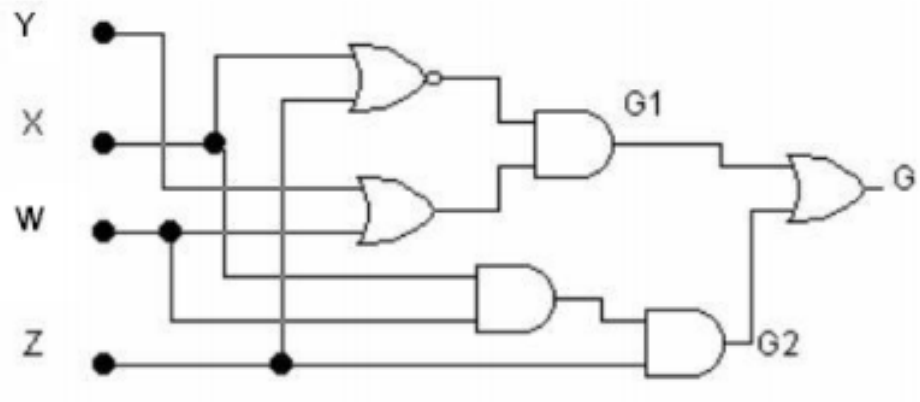
Ασκήσεις για το σπίτι

- **Άσκηση 1**
  - ***Σχεδιάστε ένα ελάχιστο κύκλωμα που στην είσοδό του να δέχεται τον αριθμό Β που σε δυαδική αναπαράσταση έχει 2 δυαδικά ψηφία, αναπαρίσταται δηλαδή ως b_1b_0 και στην έξοδό του να παράγει το 3Β+2***
- **Άσκηση 2**
  - ***Σχεδιάστε ένα ελάχιστο κύκλωμα που στην είσοδό του να δέχεται τον αριθμό Α που σε δυαδική αναπαράσταση έχει 7 δυαδικά ψηφία, αναπαρίσταται δηλαδή ως $A_6A_5A_4A_3A_2A_1A_0$ και παράγει 2 εξόδους. Η πρώτη μας δείχνει τον αριθμό των 1 της δυαδικής παράστασης του Α ενώ η άλλη τον αριθμό των 0***

Ασκήσεις για το σπίτι

- **Άσκηση 3**
  - ***Βρείτε τις λογικές συναρτήσεις F(w,x,y,z) και G(w,x,y,z) που υλοποιούνται από τα κυκλώματα των παρακάτω σχημάτων, χρησιμοποιώντας αλγεβρικούς μετασχηματισμούς. Είναι αυτές οι λογικές συναρτήσεις ισοδύναμες μεταξύ τους;***





Ασκήσεις για το σπίτι

- **Άσκηση 4**
  - ***Να κατασκευάσετε έναν πλήρη αθροιστή BCD 4 ψηφίων. Έχετε στη διάθεσή σας μόνο δύο πλήρεις δυαδικούς αθροιστές 4 ψηφίων και πύλες NAND***

Υποσημείωση: Binary-Coded Decimal Code. Ο κώδικας BCD είναι δεκαδικός κώδικας. Μετατρέπει έναν δυαδικό σε έναν δεκαδικό αριθμό.

Binary-Coded Decimal (BCD)

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001