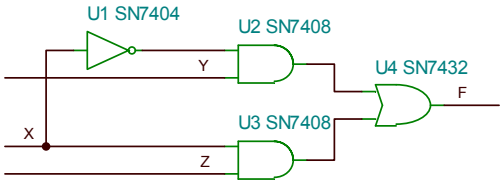
5. ΚΑΘΟΛΙΚΕΣ ΠΥΛΕΣ NAND – NOR – ΑΛΓΕΒΡΑ BOOLE – ΘΕΩΡΗΜΑ DE MORGAN

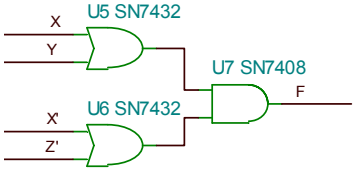
1. Διεπίπεδες υλοποιήσεις

Με δεδομένη τη συνάρτηση λογικής ενός κυκλώματος, μπορούμε να το υλοποιήσουμε με AND-OR ή OR – AND τεχνική.

Για παράδειγμα, η συνάρτηση P = X'Y+XZ = (X+Y)(X'+Z). Η πρώτη μορφή είναι εκφρασμένη σε άθροισμα γινομένων και υλοποιείται εύκολα με AND-OR τεχνική, ενώ η δεύτερη μορφή είναι εκφρασμένη σε γινόμενο αθροισμάτων και υλοποιείται εύκολα με OR-AND τεχνική. Οι δύο αυτές ισοδύναμες υλοποιήσεις φαίνονται στη συνέχεια (**Εικόνα 1**).



α) AND-OR υλοποίηση



b) OR-AND υλοποίηση

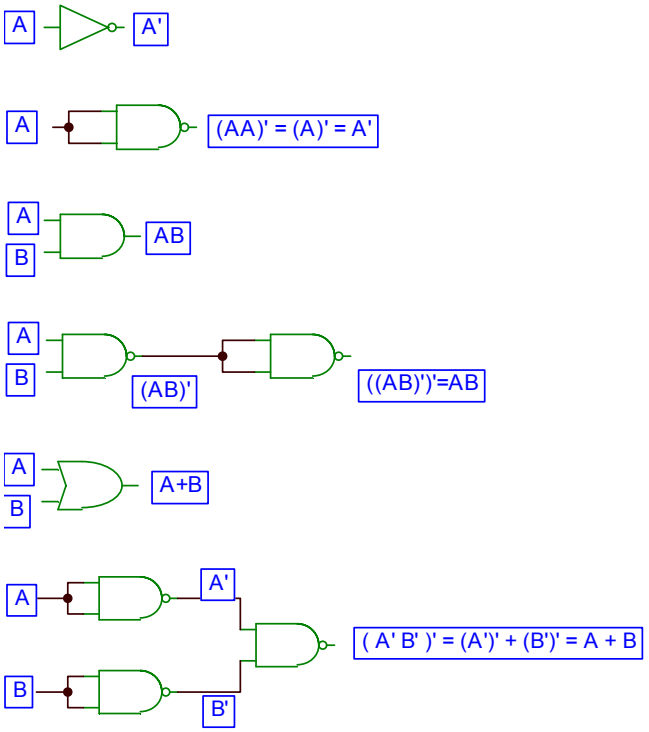
Εικόνα 1. Ισοδύναμες AND-OR και OR-AND υλοποιήσεις.

Τέτοιες υλοποιήσεις απλών κυκλωμάτων, ονομάζονται δι-επίπεδες εξαιτίας του ενός επιπέδου με πύλες AND (OR) και του $2^{ου}$ επιπέδου με πύλες OR (AND).

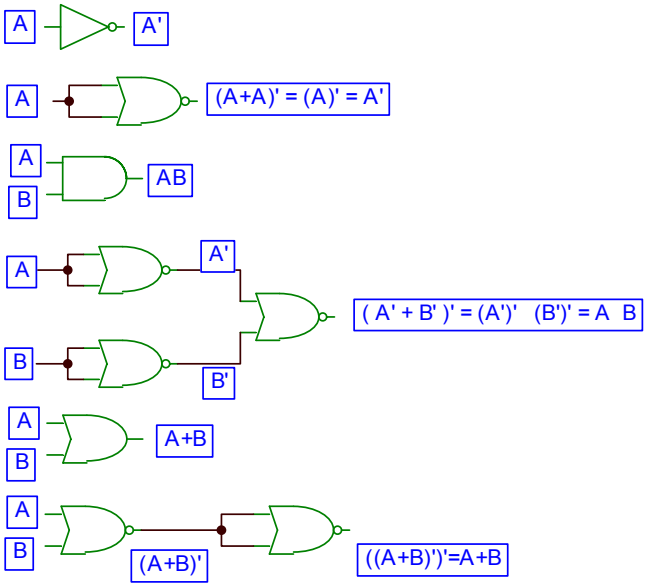
2. Η καθολικότητα των πυλών NAND και NOR

Κάθε ψηφιακό κύκλωμα που μπορεί να εκφραστεί με κανονική μορφή λογικής συνάρτησης μπορεί να υλοποιηθεί με πύλες NOT, AND, OR. Καθεμία από αυτές τις πύλες μπορεί να υλοποιηθεί μόνο με πύλες NAND ή μόνο με πύλες NOR. Συνεπώς κάθε ψηφιακό κύκλωμα μπορεί να υλοποιηθεί μόνο με πύλες NAND ή μόνο με πύλες NOR. Για το λόγο αυτό οι πύλες αυτές λέγονται καθολικές.

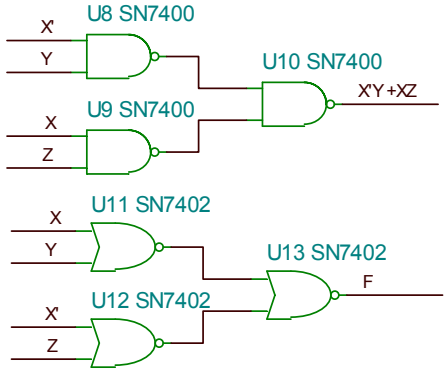
Στη συνέχεια (**Εικόνα 3**) δείχνουμε την ισοδυναμία των πυλών NOT, AND, OR μόνο πύλες NAND ή μόνο με πυλες NOR.



Εικόνα 2. Η καθολικότητα της NAND.



Εικόνα 3. Η καθολικότητα της NOR.



Εικόνα 4. Δύο ισοδύναμα κυκλώματα, a) ένα με με NAND υλοποίηση και b) ένα με NOR

Αποδεικνύεται ότι κάθε διεπίπεδη υλοποίηση στην οποία τα σήματα εισόδου συνδέονται όλα μόνο στο πρώτο επίπεδο πυλών, είναι ισοδύναμο με το ίδιο κύκλωμα όπου όλες οι πύλες ανεξαιρέτως έχουν αντικατασταθεί με πύλες NAND ή μόνο με πύλες NOR. Π.χ. στην **Εικόνα 4** φαίνονται δύο ισοδύναμα κυκλώματα, έναν με με NAND υλοποίηση και ένα με NOR, για το κύκλωμα της F= X'Y+XZ.

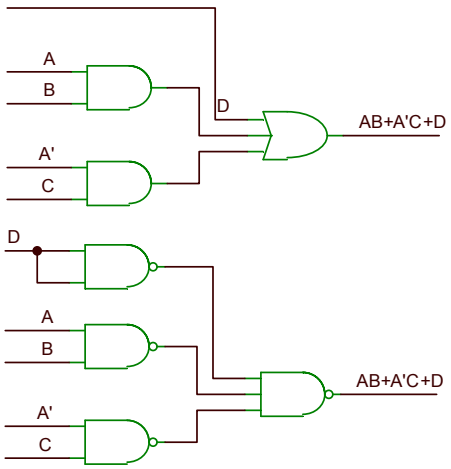
Η διαδικασία μετατροπής των υλοποιήσεων AND-OR και OR-AND σε NAND – NAND και NOR – NOR αντίστοιχα, μπορεί να εφαρμοστεί σε όλες τις 2-επιπεδες υλοποιήσεις εφόσον δεν υπάρχει απευθείας είσοδος στο 2ο επίπεδο πυλών !!

Διαφορετικά, η είσοδος που εισάγεται απευθείας στο 2ο επίπεδο πυλών, θα πρέπει να αντιστραφεί πριν εισαχθεί στο 2ο επίπεδο της NAND – NAND ή NOR – NOR υλοποίησης. Μια τέτοια περίπτωση φαίνεται στην **Εικόνα 5** για την είσοδο D.

Αν το κύκλωμα που θέλουμε να μετασχηματίσουμε αποτελείται από περισσότερα από 2 επίπεδα πυλών, η προηγούμενη διαδικασία δεν ισχύει !!!

Κάθε πύλη στο κύκλωμα πρέπει να αντικατασταθεί από το αντίστοιχο ισοδύναμο NAND ή NOR κύκλωμα. Στη συνέχεια το κύκλωμα θα πρέπει να αναλυθεί παραπέρα για να απομακρύνουμε τις πλεονάζουσες πύλες.

Οι πύλες NAND και NOR είναι συμπληρωματικές. Η συνάρτηση NOR είναι το δυϊκό της συνάρτησης NAND. Δηλαδή **NAND' = NOR** και **NOR ' = NAND**. Για το λόγο αυτό, όλες οι διαδικασίες και οι κανόνες για τα κυκλώματα με πύλες NOR είναι τα δυϊκά των αντίστοιχων για τα κυκλώματα NAND.



Εικόνα 5. a) Η είσοδος που εισάγεται απευθείας στο 2ο επίπεδο πυλών, θα πρέπει να b) αντιστραφεί πριν εισαχθεί στο 2ο επίπεδο της NAND – NAND ή NOR – NOR υλοποίησης.

Τα ψηφιακά κυκλώματα κατασκευάζονται πολύ συχνότερα με πύλες NAND, ή NOR παρά με πύλες NOT, AND, OR.

Οι πύλες NAND και NOR κατασκευάζονται ευκολότερα με ηλεκτρονικά κυκλώματα και είναι οι βασικές πύλες που χρησιμοποιούνται σε όλες τις οικογένειες ψηφιακών ολοκληρωμένων κυκλωμάτων.

Για την υλοποίηση μιας συνάρτησης Boole με πύλες NAND χρειαζόμαστε τη συνάρτηση απλοποιημενη σε μορφή αθροίσματος γινομένων. Ενώ για την υλοποίησή της με πύλες NOR, χρειαζόμαστε τη συνάρτηση απλοποιημένη σε μορφή γινομένου αθροισμάτων.

Ο κανόνας για να βρίσκουμε το κύκλωμα μιας συνάρτησης Boole με πύλες NAND είναι ο εξής:

1.Απλοποιούμε τη συνάρτηση και την εκφράζουμε ως άθροισμα γινομένων.

2.Σχεδιάζουμε μια πύλη NAND για κάθε όρο γινομένου της συνάρτησης που περιέχει τουλάχιστον δύο παράγοντες. Οι είσοδοι κάθε τέτοιας πύλης είναι οι παράγοντες των όρων. Αυτές είναι οι πύλες του πρώτου επιπέδου.

3.Σχεδιάζουμε μια πύλη NAND στο δεύτερο επίπεδο, με εισόδους που τροφοδοτούνται από τις εξόδους του πρώτου επιπέδου.

4.Ένας όρος με έναν μόνο παράγοντα χρειάζεται έναν αντιστροφέα στο πρώτο επίπεδο ή αν έχουμε ήδη το συμπλήρωμα του μπορούμε να το τροφοδοτήσουμε κατευθείαν σε μια είσοδο πύλης NAND δεύτερου επιπέδου.

Η υλοποίηση συναρτήσεων Boole με πύλες NOR απαιτεί αυτές να έχουν απλοποιηθεί σε γινόμενο αθροισμάτων. Μια τέτοια έκφραση μεταφράζεται σε μια ομάδα πυλών OR για τους όρους αθροίσματος, ακολουθούμενη από μια πύλη AND για το γινόμενο.

Ο κανόνας για την υλοποίηση συναρτήσεων Boole με πύλες NOR είναι παρόμοιος με τον κανόνα για την υλοποίηση με πύλες NAND (τρία-βήματα), μόνο που η απλοποιημένη έκφραση πρέπει να είναι σε μορφή γινομένου αθροισμάτων και οι όροι για τις πύλες NOR του πρώτου επιπέδου είναι τα αθροίσματα.

Οι όροι αθροίσματος που έχουν μόνον έναν όρο χρειάζονται πύλες NOR μιας εισόδου ή αντιστροφείς.

Ένας δεύτερος τρόπος για να υλοποιήσουμε μια συνάρτηση με πύλες NOR, είναι να χρησιμοποιήσουμε το συμπλήρωμα της συνάρτησης εκφρασμένο σαν γινόμενο αθροισμάτων. Αυτό δίνει μια υλοποίηση της F' σε δύο επίπεδα λογικής ή μιας υλοποίηση της F σε

τρία επίπεδα.

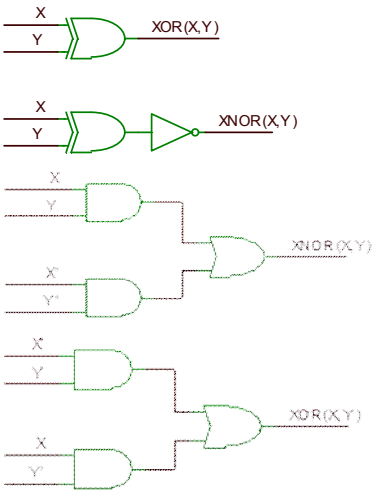
Όπως θα δούμε στο επόμενο κεφάλαιο, για να πάρουμε το απλοποιημένο γινόμενο αθροισμάτων από το χάρτη Karnaugh, πρέπει να συνδυάσουμε τα μηδενικά του χάρτη και μετά να πάρουμε το συμπλήρωμα της συνάρτησης. Για να πάρουμε το απλοποιημένο γινόμενο αθροισμάτων για το συμπλήρωμα της συνάρτησης πρέπει να συνδυάσουμε τις μονάδες του χάρτη και να πάρουμε το συμπλήρωμα της συνάρτησης.

3. Η πύλη XOR και η πύλη EQUIVALENCE

Μια πύλη που χρησιμοποιείται συχνά στην κατασκευή κυκλωμάτων υπολογισμών είναι η XOR, ενώ στην κατασκευή κυκλωμάτων αναγνώρισης/διόρθωσης χρησιμοποιείται συχνά η EQUIVALENCE.

Αυτές οι δύο πύλες είναι συμπληρωματικές.

Ο πίνακας αληθείας τους, τα σύμβολά τους και η ισοδύναμες AND-OR υλοποιήσεις τους φαίνονται στην **Εικόνα 6**.



XY	$X \oplus Y$	$X \odot Y$
00	0	1
01	1	0
10	1	0
11	0	1

Εικόνα 6. Ο πίνακας αληθείας, τα σύμβολά και η ισοδύναμες AND-OR υλοποιήσεις των XOR και EQUIVALENCE.

Μια XOR n-εισόδων είναι ισοδύναμη προς μια Boolean συνάρτηση με 2^{n-1} ελαχιστόρους, των οποίων το δυαδικό ισοδύναμο έχει περιττό πλήθος 1.

Μια EQUIVALENCE n-εισόδων είναι ισοδύναμη προς μια Boolean συνάρτηση με 2^{n-1} ελαχιστόρους των οποίων το δυαδικό ισοδύναμο έχει άρτιο πλήθος 1.

Όταν το πλήθος των μεταβλητών της συνάρτησης είναι περιττός, οι ελαχιστόροι με άρτιο αριθμό 0 είναι όσοι με περιττό αριθμό 1. Τότε οι XOR και EXCLUSIVE είναι ίσες.

Όταν το πλήθος των μεταβλητών είναι άρτιο, αυτές οι δύο πύλες είναι συμπληρωματικές.

4. Αξιώματα – Θεωρήματα Άλγεβρας Boole

Η άλγεβρα Boole είναι το μαθηματικό εργαλείο με το οποίο χειριζόμαστε τα ψηφιακά κυκλώματα.

Πρόκειται για μια αλγεβρική δομή που ορίζεται στο σύνολο B={0,1} μαζί με δύο δυαδικούς τελεστές + και *.

Τα **αξιώματα της άλγεβρας Boole** αναφέρονται στη συνέχεια:

1. Είναι κλειστή ως προς τον τελεστή + και τον τελεστή *.
2. Το στοιχείο 0 είναι το ουδέτερο στοιχείο ως προς την πρόσθεση: x+0=x. Το στοιχείο 1 είναι το ουδέτερο στοιχείο ως προς τον πολλαπλασιασμό: x*1=x.
3. Ισχύει η αντιμεταθετικότητα ως προς τον τελεστή +: x+y=y+x. Ισχύει η αντιμεταθετικότητα ως προς τον τελεστή *: x*y=y*x.
4. Ισχύει η επιμεριστικότητα του * ως προς τον + και το αντίστροφο: x*(y+z)=(x*y)+(x*z) και x+(y*z)=(x+y)*(x+z).
5. Για κάθε στοιχείο x στο B, υπάρχει το στοιχείο x' του B (που ονομάζεται συμπλήρωμα του x), τέτοιο ώστε: x' + x = 1 και x' * x = 0.

Υπάρχουν τουλάχιστον 2 στοιχεία x,y στο B τέτοια ώστε $x \neq y$.

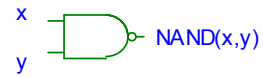
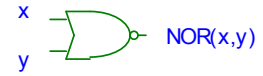
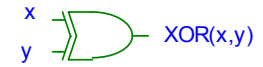
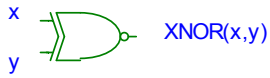
Τα **θεωρήματα της άλγεβρας Boole** αναφέρονται στη συνέχεια:

1. x*x=x, x+x=x.
2. 1+x=1, 0*x=0.
3. x''=x.
4. x*(y*z) = (x*y)*z, x+(y+z)=(x+y)+z.
5. x*(x+y)=x.
6. x+x*y=x.
7. x+x'*y=x+y.
8. Θεώρημα de Morgan: (x+y)'=x'*y' , (x*y)'=x'+y'.

5. Ασκήσεις

1. **Σχεδιάστε το λογικό σύμβολο, τον πίνακα αληθείας και την πράξη των επόμενων πυλών (2 εισόδων για τις πύλες NAND,NOR και XNOR).**

Λύση.

NAND	
NOR	
XOR	
XNOR	

NAND	NAND(x,y) = (x y)’
NOR	NOR(x,y) = (x + y)’
XOR	XOR(x,y) = x’y + xy’ = x(+)y
XNOR	XNOR(x,y) = xy + x’y’ = (x(+)y)’

NAND

x	y	NAND(x,y)
0	0	1
0	1	1
1	0	1
1	1	0

NOR

x	y	NOR(x,y)
0	0	1
0	1	0
1	0	0
1	1	0

XOR

x	y	XOR(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

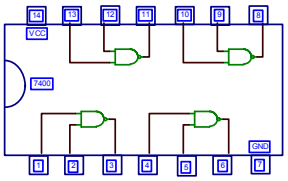
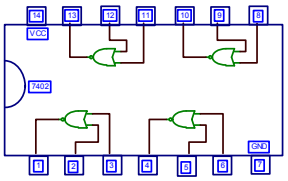
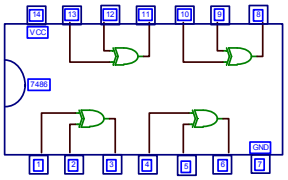
XNOR

x	y	XNOR(x,y)
0	0	1
0	1	0
1	0	0
1	1	1

Εικόνα 7. NAND, NOR, XOR, XNOR.

2. **Σχεδιάστε τα περιεχόμενα των ολοκληρωμένων κυκλωμάτων των πυλών NAND, NOR και XOR και αριθμήστε τους ακροδέκτες τους.**

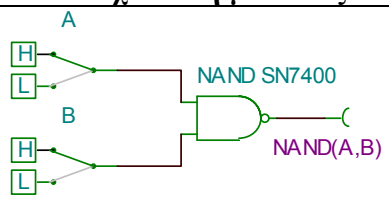
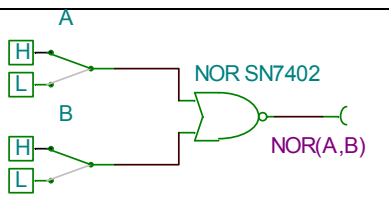
Λύση.

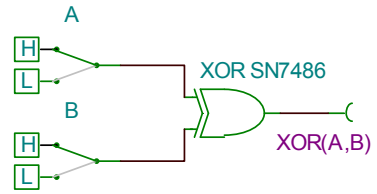
Ονομα Πύλης	Κωδικός OK TTL	Διάγραμμα σύνδεσης ολοκληρωμένου κυκλώματος
NAND	**74LS00**	
NOR	**74LS02**	
XOR	**74LS86**	

Εικόνα 8. Τοπολογία OK NAND, NOR, XOR.

3. **Προσομοιώστε τη λειτουργία των πυλών NAND, NOR, XOR στο TINA: a) χρησιμοποιώντας πύλες και b) χρησιμοποιώντας ολοκληρωμένα κυκλώματα.**
4. **Να σχεδιάσετε τα κυκλώματα στον επόμενο πίνακα (Εικόνα 9).**

Λύση. Εκκρεμεί η σχεδίαση με ολοκληρωμένο κύκλωμα.

ΠΡΑΞΗ	Σχεδίαση με πύλες
C = NAND(A,B)	
C = NOR(A,B)	

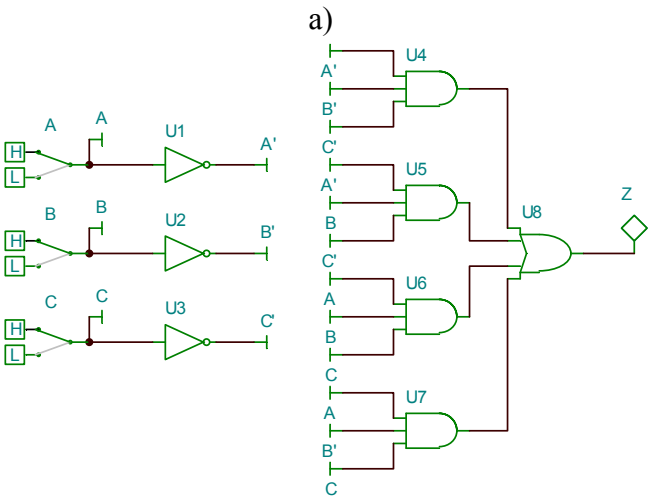
C = XOR(A,B)	

Εικόνα 9. Κυκλώματα προσομοίωσης NAND, NOR, XOR στο TINA.

5. a) Να σχεδιαστεί το κύκλωμα της λογικής συνάρτησης Z = A'B'C'+A'BC'+ABC+AB'C με λογική σχεδίασης AND-OR. b) Να απλοποιηθεί η συνάρτηση Z και να υλοποιηθεί το λογικό της κύκλωμα με πύλες NAND. c) Να σημειωθούν όλα τα άκρα των πυλών που θα χρησιμοποιηθούν. Πόσα και ποια IC απαιτούνται για την πραγματοποίηση της σχεδίασης;

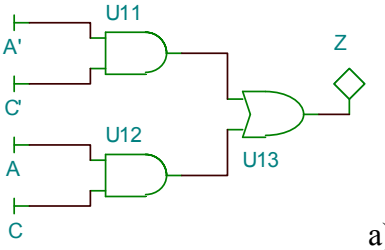
Λύση.

a)

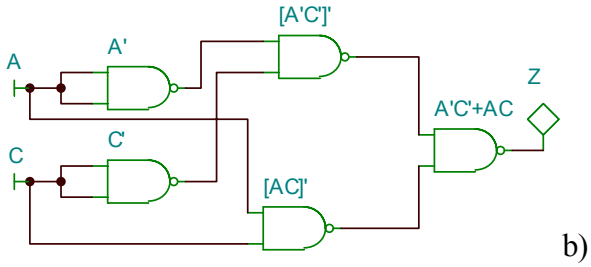


Εικόνα 10. Z = A'B'C'+A'BC'+ABC+AB'C με λογική σχεδίασης AND-OR.

b) Z = A'B'C' + A'BC' + ABC + AB'C = A'C'(B'+B) + AC(B+B') = A'C'1 + AC1 = Z = A'C' + AC

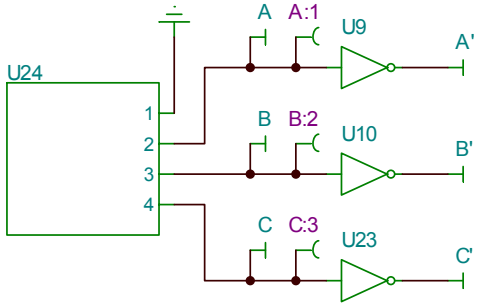


a)

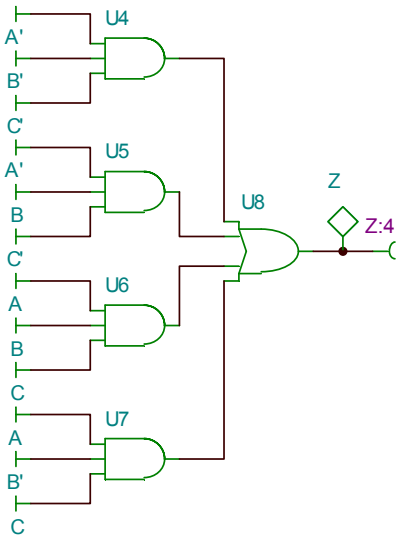


b)

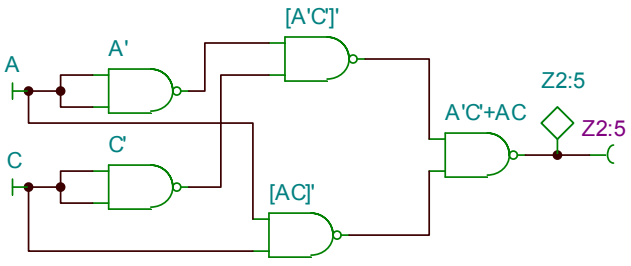
Εικόνα 11. a) Η απλοποιημένη Z. b) Υλοποίηση της απλοποιημένης συνάρτησης με NAND.



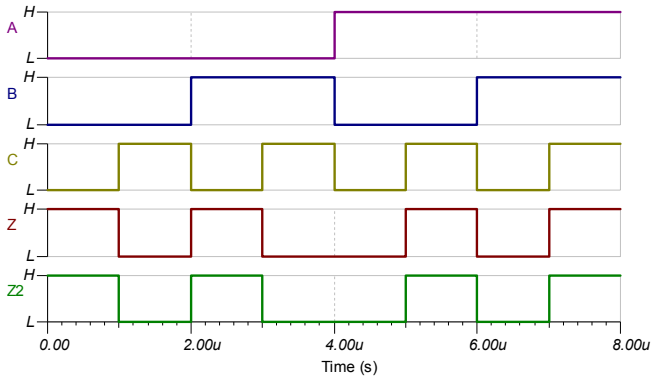
a)



b)



c)

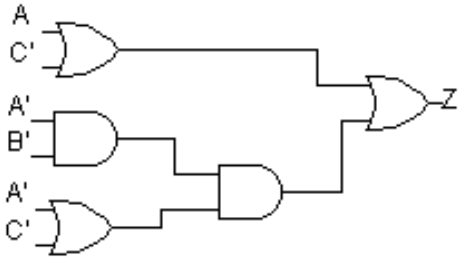


d)

Εικόνα 12. Προσομοίωση με α) γεννήτρια παραγωγής 4bit b) αρχικού και c) απλοποιημένου κυκλώματος στο TINA. d) Το τελικό διάγραμμα χρονισμού.

c) Αφήνεται για τον αναγνώστη.

6. a) Να βρεθεί η έξοδος του λογικού κυκλώματος της Εικόνας 13. b) Να απλοποιηθεί και να σχεδιαστεί το λογικό της κύκλωμα με λογική πυλών NAND. c) Πόσα και ποια ολοκληρωμένα απαιτούνται για την πραγματοποίηση του κυκλώματος; Δ) Να γίνει υλοποίηση του απλοποιημένου κυκλώματος με πύλες NAND στο PENCIL-BOX.



Εικόνα 13. Κύκλωμα άσκησης 6.

7. Να δείξετε με τη βοήθεια πινάκων αληθείας την ισχύ των επόμενων ιδιοτήτων: a) Το θεώρημα DeMorgan για τρεις μεταβλητές: (x+y+z)' = x'y'z' και (xyz)'=x'+y'+z'. b) Την ιδιότητας διαμέρισης (distributive law): x+yz = (x+y)(x+z).

Λύση.

a)

x	y	z	x+y+z	(x+y+z)'
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

x'	y'	z'	x'y'z'
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

x	y	z	xyz	(xyz)'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

x'	y'	z'	x'+y'+z'
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

b)

x	y	z	yz	x+yz
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

x+y	x+z	(x+y)(x+z)
0	0	0
0	1	0
1	0	0
1	1	1
1	1	1
1	1	1
1	1	1
1	1	1

8. Να απλοποιήσετε τις επόμενες εκφράσεις Boole στον αντίστοιχο αριθμό παραμέτρων. a) A'C' + ABC + AC', 3 παραμέτρους. b) (x'y'+z)' + z + xy + wz, 3 παραμέτρους. c) A'B(D'+C'D) + B(A+A'CD), 1 παράμετρος. d) (A'+C)(A'+C')(A+B+C'D), 4 παραμέτρους.

Λύση.

(a)

A'C' + ABC + AC' =

A'C' + AC' + ABC

= C’(A’+ A) + ABC
= C’·1 + ABC
= C’ + ABC
= (C’+ AB)(C’+C) [distributive]
= AB + C’

(b)

(x’y’+z)’ + z + xy + wz
= (x’y’+z)’ + z + wz + xy
= (x’y’+z)’ + z(1+ w) + xy
= (x’y’+z)’ + z + xy
= (x + y)z’ + z + xy [DeMorgan]
= (z + (x + y)) · (z + z’)
+ xy [distributive]
= (z + (x + y)) · 1 + xy
= x + y + z + xy
= x + y + z [absorption]

(c)

A’B(D’ + C’D) + B(A+A’CD)
= A’BD’ + A’BC’D + AB+ A’BCD
= A’BD(C+C’)+ A’BD’+ AB
= A’BD+ A’BD’+ AB
= A’B(D+D’)+ AB
= A’B+ AB
= B(A’+ A)
= B

(d)

(A’+C)(A’+C’)(A+B+C’D)
= (A’+C)(A’+C’)(A+B+C’D)
= (A’ + CC’)(A + B + C’D)
= A’(A + B + C’D)
= A’A + A’B + A’C’D
= A’B + A’C’D
= A’(B + C’D)

9. Βρείτε το συμπληρωματικό της $F = x + yz$; Στη συνέχεια δείξτε ότι $FF' = 0$ και $F + F' = 1$.

Λύση.

$F = x + yz$

$Dual(F) = x \bullet (y + z)$

$F' = x' \bullet (y' + z')$

$FF' = (x + yz) \bullet (x' \bullet (y' + z'))$

$= (xx' + x'yz) \bullet (y' + z')$

$= x'yz \bullet (y' + z')$

$= x'yy'z + x'yzz'$

$= 0$

$F + F'$

$= (x + yz) + (x' \bullet (y' + z'))$

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

$= (1 + yz) + (x + yz + y' + z')$

$= 1 + (x + yz + y' + z')$

$= 1$

10. Διατυπώστε τον πίνακα αληθείας της επόμενης συνάρτησης: $F = xy + xy' + y'z$.

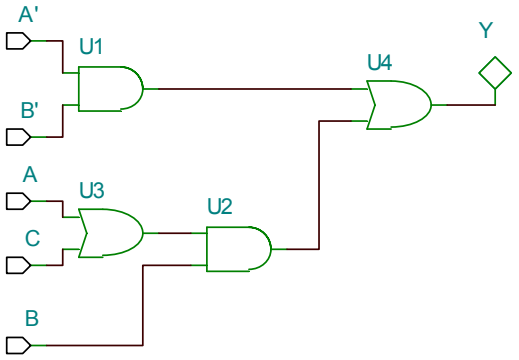
Λύση.

x	**y**	**z**	**F**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

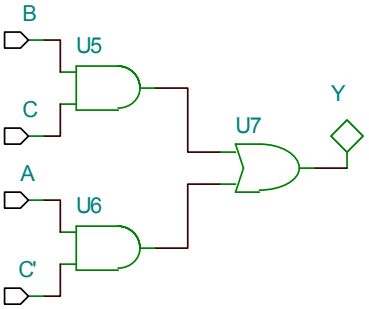
11. Σχεδιάστε το λογικό διάγραμμα των επόμενων εκφράσεων: Y=A’B’+B(A+C), Y=BC+AC’, Y=A+CD, Y=(A+B)(C’+D).

Λύση.

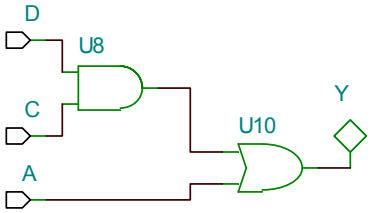
Y=A’B’+B(A+C).



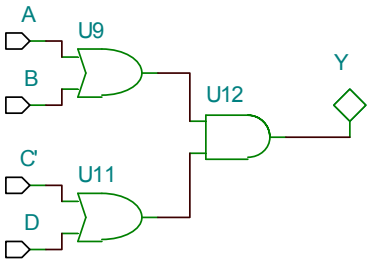
Y=BC+AC’.



Y=A+CD.



Y=(A+B)(C'+D).



Εικόνα 14. Τα κυκλώματα της άσκησης.

12. Δίνεται η συνάρτηση : F=XY'Z+ X'Y'Z+ W'XY+ WX'Y+ WXY. a) Διατυπώστε τον πίνακα αληθείας της. b) Σχεδιάστε το λογικό της διάγραμμα με πύλες. c) Απλοποιήστε τη συνάρτηση σε ελάχιστο αριθμό παραμέτρων με χρήση ιδιοτήτων της άλγεβρας Boole. d) Διατυπώστε τον πίνακα αληθείας της απλοποιημένης συνάρτησης. e) Σχεδιάστε με πύλες το λογικό διάγραμμα της απλοποιημένης συνάρτησης και συγκρίνετε το πλήθος των πυλών σε σχέση με αυτό που χρησιμοποιήσατε στο b).

Λύση.

a) Ο πίνακας αληθείας:

X	Y	Z	W	F	Fs
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

b) Το λογικό διάγραμμα της F φαίνεται στην **Εικόνα 15a.**

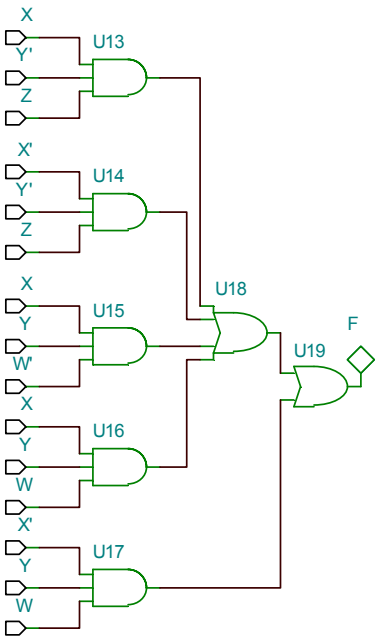
c) Η απλοποιημένη μορφή της F:

F= XY'Z+ X'Y'Z+ WXY + W'XY+ WX'Y =
Y'Z (X+X') + XY (W+W') + WX'Y=
Y'Z+ XY+ WX'Y

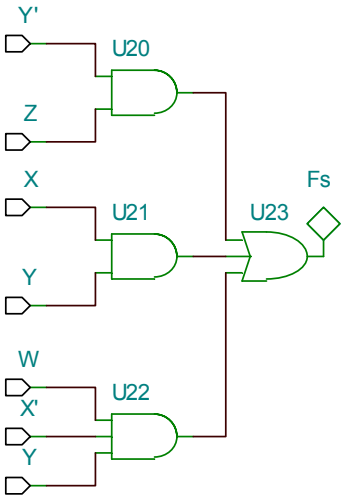
d) Ο πίνακας αληθείας της απλοποιημένης έκφρασης της συνάρτησης έχει προστεθεί στον πίνακα αληθείας στο μέρος a) στη στήλη με τίτλο Fs. Προφανώς F = Fs.

e) Το λογικό διάγραμμα της F φαίνεται στην **Εικόνα 15b.**

Παρατηρούμε ότι για τη σχεδίαση της F χρειαζόμαστε 5 AND με 3 inputs και 1 OR με 5 inputs. Για τη σχεδίασης της F(simplified) χρειαζόμαστε 2 AND gates με 2 inputs, 1 OR με 3 inputs και 1 AND με 3 inputs.



a)



b)

Εικόνα 15. Πίνακες αληθείας και κυκλώματα της άσκησης. a) Το αρχικό κύκλωμα. b) Το απλοποιημένο κύκλωμα.

13. Να μετατρέψετε τις επόμενες εκφράσεις στη συμπληρωματική κανονική μορφη: (a) F(x, y, z) = ∑(1,3,7) (b) F(A,B,C,D)= ∏(0,1,2,3,4,6,12).

Λύση.
(a)
F(x, y, z)=
$\sum(1,3,7)$ =
$\prod(0,2,4,5,6)$=

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

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

$(x' + y' + z)$

(b)
F(A,B,C,D)=
$\prod(0,1,2,3,4,6,12) = \sum(5,7,8,9,10,11,13,14,15)$ =
$(A'BC'D) + (A'BCD) +$

$(AB'C'D') + (AB'C'D) +$

$(AB'CD') + (AB'CD) +$

$(ABC'D) + (ABCD') +$

$(ABCD)$

14. Δείξτε ότι η δυαδική πύλη της XOR είναι ίση με τη συμπληρωματική της.

Λύση.
XOR:
X⊕Y = XY' + X'Y
Dual of XOR: =
(X +Y') (X'+Y) =
XX'+ XY +X'Y' +YY' =
XY + X'Y'
Complement of XOR =
(XNOR) =
(X⊕Y)' =
(XY' + X'Y)' =
(X'+Y) (X +Y') =
XX'+ XY + X'Y'+ YY' =
XY + X'Y'

15. Δείξτε ότι η θετικής λογικής πύλη NAND είναι η αρνητικής λογικής πύλη NOR και αντίστροφα.

Λύση.

Πίνακας αληθείας πύλης NAND			**Πίνακας αληθείας θετικής λογικής NAND (L = 0, H = 1)**			**Πίνακας αληθείας θετικής λογικής NAND (L = 1, H = 0)**		
X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
0	0	1	L	L	H	1	1	0
0	1	1	L	H	H	1	0	0
1	0	1	H	L	H	0	1	0
1	1	0	H	H	L	0	0	1

Ο τελευταίος πίνακας είναι αυτός μιας πύλης NOR αρνητικής λογικής.

16. Να αποδείξετε το νόμο x*(x+y)=x μέσω της συμπλήρωσης του πίνακα αληθείας. Να διαπιστώσετε την ισχύ του νόμου μέσω προσομοίωσης. Να σχεδιάσετε το κύκλωμα.

x	**y**	**x+y**	**x*(x+y)**
0	0		
0	1		
1	0		
1	1		

17. Να αποδείξετε το νόμο x+x*y=x μέσω της συμπλήρωσης του πίνακα αληθείας. Να διαπιστώσετε την ισχύ του νόμου μέσω προσομοίωσης. Να σχεδιάσετε το κύκλωμα.

x	**y**	**x*y**	**x+x*y**
0	0		
0	1		
1	0		
1	1		

18. Να αποδείξετε το νόμο x+x'*y=x+y μέσω της συμπλήρωσης του πίνακα αληθείας. Να διαπιστώσετε την ισχύ του νόμου μέσω προσομοίωσης. Να σχεδιάσετε το κύκλωμα.

x	**y**	**x'**	**x'*y**	**x+y**
0	0			
0	1			
1	0			
1	1			

19. Να αποδείξετε το θεώρημα de Morgan: (x+y)'=x'*y' , (x*y)'=x'+y'μέσω της συμπλήρωσης του πίνακα αληθείας. Να διαπιστώσετε την ισχύ του νόμου μέσω προσομοίωσης. Να σχεδιάσετε το κύκλωμα.

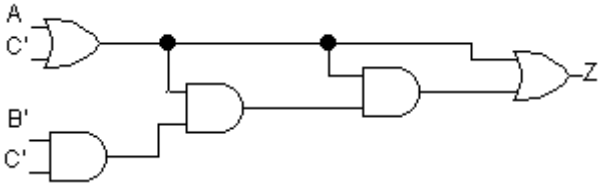
x	**y**	**x*y**	**(x*y)'**	**x'**	**y'**	**x'+y'**	**x'*y'**	**x+y**	**(x+y)'**
0	0								
0	1								
1	0								
1	1								

20. (a) Χρησιμοποιείστε κατάλληλα ολοκληρωμένα κυκλώματα για να αποδείξετε πειραματικά με το pencilbox logic designer την ισχύ των δύο εκφράσεων του θεωρήματος de Morgan. (b) Να σχεδιάσετε τα κυκλώματα με ξεκάθαρη αρίθμηση των ακροδεκτών.

21. (a) Να εξεταστεί αν ισχύουν οι παρακάτω ισότητες A,B,C. (b) Να σχεδιαστούν τα κυκλώματα επαλήθευσής τους με την κατάλληλη λογική (NAND ή NOR) για κάθε μέλος με αρίθμηση των άκρων. (c) Πόσα και ποια ολοκληρωμένα απαιτούνται σε κάθε περίπτωση;
A) A B C + A C + B C = (A+B) C
B) A B D' + A B' D + A B' D' = A (B' + D')

C) A' B C + A' B' C + A B + A B' + A' B' C' = A + B' + C

22. (a) Να βρεθεί η έξοδος του λογικού κυκλώματος της Εικόνας 16. (b) Να απλοποιηθεί και να σχεδιαστεί το λογικό της κύκλωμα με λογική πυλών NAND. (c) Πόσα και ποια ολοκληρωμένα απαιτούνται για την πραγματοποίηση του κυκλώματος; (d) Να γίνει υλοποίηση του απλοποιημένου κυκλώματος με πύλες NAND στο pencil-box



Εικόνα 16. Κύκλωμα άσκησης 22.

23. Απλοποιήστε τις παρακάτω συναρτήσεις Boole σε έναν ελάχιστο αριθμό παραγόντων:

a) x'y' + xy + x'y
b) (x+y)(x+y')
c) x'y + xy' + xy + x'y'
d) x'+xy+xz'+xy'z'
e) xy'+y'z+x'z' (Χρησιμοποιήστε το θεώρημα απαλοιφής)

24. Απλοποιήστε τις παρακάτω συναρτήσεις Boole σε έναν ελάχιστο αριθμό παραγόντων:

a) xyz + x'y + xyz'
b) x'yz + xz
c) (x+y)'(x'+y')
d) xy+x(wz+wz')
e) (yz'+x'w)(xt'+zw')

25. Βρείτε το συμπλήρωμα των παρακάτω συναρτήσεων:

a) xy' + x'y
b) (xy'+z)w' + f
c) xy(z'w+zw') + x'y'(z'+w)(x+w')
d) (x+y'+z)(x'+z')(x+y)

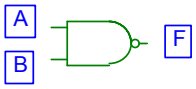
26. Χρησιμοποιώντας το θεώρημα De Morgan, μετατρέψτε τις παρακάτω συναρτήσεις Boole σε ισοδύναμες συναρτήσεις που να περιέχουν μόνο πράξεις OR και συμπληρώματος. Δείξτε ότι οι συναρτήσεις μπορούν να εκφραστούν με λογικά διαγράμματα που θα έχουν μόνο πύλες OR και αντιστροφείς:

a) F = x'y' + x'z + y'z
b) F = (y+z')(x+y)(y'+z)

Υλοποιήσεις με VHDL

Πύλη NAND

```
--import std_logic from the IEEE library
library ieee;
use ieee.std_logic_1164.all;

--ENTITY DECLARATION: name, inputs,
outputs
entity nandGate is
   port( A, B : in std_logic;
         F : out std_logic);
end nandGate;
```




```
--FUNCTIONAL DESCRIPTION: how the NAND
Gate works
architecture func of nandGate is
begin
   F <= A nand B;
end func;
```

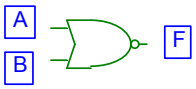
Testbench:

```
--import std_logic from the IEEE library
library ieee;
use ieee.std_logic_1164.all;

--ENTITY DECLARATION: no inputs, no
outputs
entity nandGate_tb is
end nandGate_tb;

-- Describe how to test the NAND Gate
architecture tb of nandGate_tb is
   --pass nandGate entity to the
testbench as component
   component nandGate is
   port( A, B : in std_logic;
         F : out std_logic);
   end component;

   signal  inA, inB, outF : std_logic;
begin
   --map the testbench signals to the
ports of the nandGate
   mapping: nandGate port map(inA, inB,
outF);

   process
      --variable to track errors
      variable errCnt : integer := 0;
   begin
      --TEST 1
      inA <= '0';
      inB <= '0';
      wait for 15 ns;
      assert(outF = '1')  report "Error
1" severity error;
```

```
      if(outF /= '1') then
         errCnt := errCnt + 1;
      end if;

      --TEST 2
      inA <= '0';
      inB <= '1';
      wait for 15 ns;
      assert(outF = '1')  report "Error
2" severity error;
      if(outF /= '1') then
         errCnt := errCnt + 1;
      end if;

      --TEST 3
      inA <= '1';
      inB <= '1';
      wait for 15 ns;
      assert(outF = '0')  report "Error
3" severity error;
      if(outF /= '0') then
         errCnt := errCnt + 1;
      end if;

      ------------- SUMMARY -----------
--
      if(errCnt = 0) then
         assert false report "Good!"
severity note;
      else
         assert true report "Error!"
severity error;
      end if;

   end process;
end tb;
-----------------------------------------
----
configuration cfg_tb of nandGate_tb is
   for tb
   end for;
end cfg_tb;
```

Πύλη NOR

```
--import std_logic from the IEEE library
library ieee;
use ieee.std_logic_1164.all;

--ENTITY DECLARATION: name, inputs,
outputs
entity norGate is
   port( A, B : in std_logic;
         F : out std_logic);
end norGate;
```




```
--FUNCTIONAL DESCRIPTION: how the NOR
Gate works

architecture func of norGate is
begin
   F <= A nor B;
end func;
```

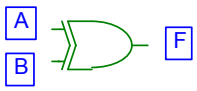
Testbench:

```
--import std_logic from the IEEE library
library ieee;
use ieee.std_logic_1164.all;

--ENTITY DECLARATION: no inputs, no
outputs
entity norGate_tb is
end norGate_tb;

-- Describe how to test the NOR Gate
architecture tb of norGate_tb is
   --pass norGate entity to the
testbench as component
   component norGate is
   port( A, B : in std_logic;
         F : out std_logic);
   end component;

   signal  inA, inB, outF : std_logic;
begin
   --map the testbench signals to the
ports of the norGate
   mapping: norGate port map(inA, inB,
outF);

   process
      --variable to track errors
      variable errCnt : integer := 0;
   begin
      --TEST 1
      inA <= '0';
      inB <= '0';
      wait for 15 ns;
      assert(outF = '1')  report "Error
1"  severity error;
      if(outF /= '1') then
         errCnt := errCnt + 1;
      end if;

      --TEST 2
      inA <= '0';
      inB <= '1';
      wait for 15 ns;
      assert(outF = '0')  report "Error
2"  severity error;
      if(outF /= '0') then
         errCnt := errCnt + 1;
      end if;

      --TEST 3
      inA <= '1';
      inB <= '1';
      wait for 15 ns;
      assert(outF = '0')  report "Error
3"  severity error;
      if(outF /= '0') then
         errCnt := errCnt + 1;
      end if;

      ------------- SUMMARY -----------
```

```
--
      if(errCnt = 0) then
         assert false report "Good!"
severity note;
      else
         assert true report "Error!"
severity error;
      end if;

   end process;
end tb;
----------------------------------------
----
configuration cfg_tb of norGate_tb is
   for tb
   end for;
end cfg_tb;
```

Πύλη XOR

```
--import std_logic from the IEEE library
library ieee;
use ieee.std_logic_1164.all;

--ENTITY DECLARATION: name, inputs,
outputs
entity xorGate is
   port( A, B : in std_logic;
         F : out std_logic);
end xorGate;
```




```
--FUNCTIONAL DESCRIPTION: how the XOR
Gate works
architecture func of xorGate is
begin
   F <= A xor B;
end func;
```

Testbench:

```
--import std_logic from the IEEE library
library ieee;
use ieee.std_logic_1164.all;

--ENTITY DECLARATION: no inputs, no
outputs
entity xorGate_tb is
end xorGate_tb;

-- Describe how to test the XOR Gate
architecture tb of xorGate_tb is
   --pass xorGate entity to the
testbench as component
   component xorGate is
   port( A, B : in std_logic;
         F : out std_logic);
   end component;

   signal  inA, inB, outF : std_logic;
begin
   --map the testbench signals to the
ports of the xorGate
   mapping: xorGate port map(inA, inB,
outF);

   process
      --variable to track errors
      variable errCnt : integer := 0;
   begin
      --TEST 1
      inA <= '0';
      inB <= '0';
      wait for 15 ns;
      assert(outF = '0')  report "Error
1"  severity error;
      if(outF /= '0') then
         errCnt := errCnt + 1;
      end if;

      --TEST 2
      inA <= '0';
      inB <= '1';
      wait for 15 ns;
      assert(outF = '1')  report "Error
2"  severity error;
      if(outF /= '1') then
         errCnt := errCnt + 1;
      end if;

      --TEST 3
      inA <= '1';
      inB <= '1';
      wait for 15 ns;
      assert(outF = '0')  report "Error
3"  severity error;
      if(outF /= '0') then
         errCnt := errCnt + 1;
      end if;

      -------------- SUMMARY -----------
--
      if(errCnt = 0) then
         assert false report "Good!"
        severity note;
      else
         assert true report "Error!"
        severity error;
     end if;

   end process;
end tb;
----------------------------------------
configuration cfg_tb of xorGate_tb is
   for tb
   end for;
end cfg_tb;
```

Άσκηση.

Γράψτε VHDL κώδικα και προσομοιώστε τη λειτουργία του επόμενου κυκλώματος:

13