



Λογική Σχεδίαση



Λογική σχεδίαση

- Θεωρία: **Βασίλειος Καρακώστας, Επίκουρος Καθηγητής**
Διονύσιος Βασιλόπουλος, ΕΔΙΠ
 - *E-mail:* vkarakos@di.uoa.gr | denis@di.uoa.gr
- Εργαστήριο: **Διονύσιος Βασιλόπουλος, ΕΔΙΠ**
- Πληροφορίες για το μάθημα θα βρείτε στο eclass:
<https://eclass.uoa.gr/courses/D13/>
 - Οι διαλέξεις του μαθήματος
 - Λυμένες ασκήσεις
- Το εργαστήριο προσφέρει την κατανόηση των βασικών αρχών και πρακτικών της ψηφιακής λογικής, σύμφωνα με το αρχαίο κινέζικο ρητό:
 - **“Ακούω και ξεχνώ,
βλέπω και θυμάμαι,
εφαρμόζω και κατανοώ (μαθαίνω)”**

Λογική σχεδίαση

■ Μαθησιακοί στόχοι:

- Να μυηθούν βήμα-βήμα οι αμύητοι φοιτητές, χωρίς την αναγκαιότητα κάποιας πρότερης γνώσης, αρχικά στα αριθμητικά συστήματα και στις λογικές πύλες, κατόπιν στη συνδυαστική λογική και τέλος στην ακολουθιακή λογική
- Η μύηση συμπεριλαμβάνει και θεμελιώδεις γνώσεις προγραμματισμού στη γλώσσα περιγραφής υλικού (VHDL)
 - **Σήμερα, το hardware είναι κυρίως προγραμματισμός**
- *Το ιδιαίτερο τελετουργικό μύησης στην ψηφιακή σχεδίαση, που ακολουθείται στο παρόν μάθημα, αφορά όλους τους φοιτητές που ενδιαφέρονται να εντρυφήσουν:*
 - στην ανάπτυξη του υλικού και του λογισμικού υπολογιστικών συστημάτων, τηλεπικοινωνιακών και δικτυακών συστημάτων, και συστημάτων ψηφιακής επεξεργασίας σήματος και πληροφοριών
 - στην επιστήμη και τεχνολογία των υπολογιστών

Λογική σχεδίαση

- **Προσδοκώμενα μαθησιακά αποτελέσματα**
- Με την επιτυχή ολοκλήρωση του μαθήματος ο φοιτητής/η φοιτήτρια θα είναι σε θέση να:
 - περιγράφει τις αρχές και πρακτικές της συνδυαστικής και ακολουθιακής λογικής,
 - σχεδιάζει όλες τις βασικές μονάδες ενός ψηφιακού συστήματος στο επίπεδο της λογικής σχεδίασης,
 - αναλύει τον χρονισμό των ψηφιακών συστημάτων,
 - περιγράφει τις σύγχρονες διατάξεις μνήμης και λογικής,
 - κατέχει βασική γνώση προγραμματισμού σε γλώσσα περιγραφής υλικού (VHDL)
 - κυρίως μέσω του εργαστηρίου του μαθήματος

Λογική σχεδίαση

■ Βιβλιογραφία

- **ΨΗΦΙΑΚΗ ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ, ΕΚΔΟΣΗ ARM®**
- **Κωδικός Βιβλίου στον Εύδοξο:** 86055864
- **Έκδοση:** 1η/2019
- **Συγγραφείς:** SARAH L. HARRIS, DAVID MONEY HARRIS
- **ISBN:** 978-960-461-961-0
- **Τύπος:** Σύγγραμμα
- **Διαθέτης (Εκδότης):** ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ



Τα ψηφιακά συστήματα βρίσκονται παντού!

- Ψηφιακές συσκευές καταναλωτών (consumer electronics):

1960

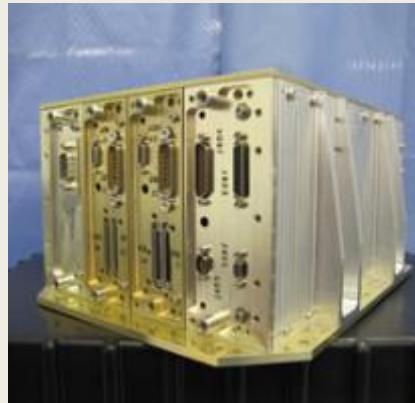


1980



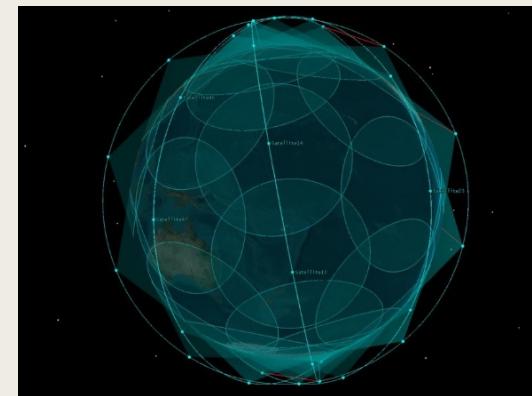
Τα ψηφιακά συστήματα βρίσκονται παντού!

- Μερικά παραδείγματα από την αεροδιαστημική:



Adaptive image & video processing/compression

Earth observation and hyperspectral imaging

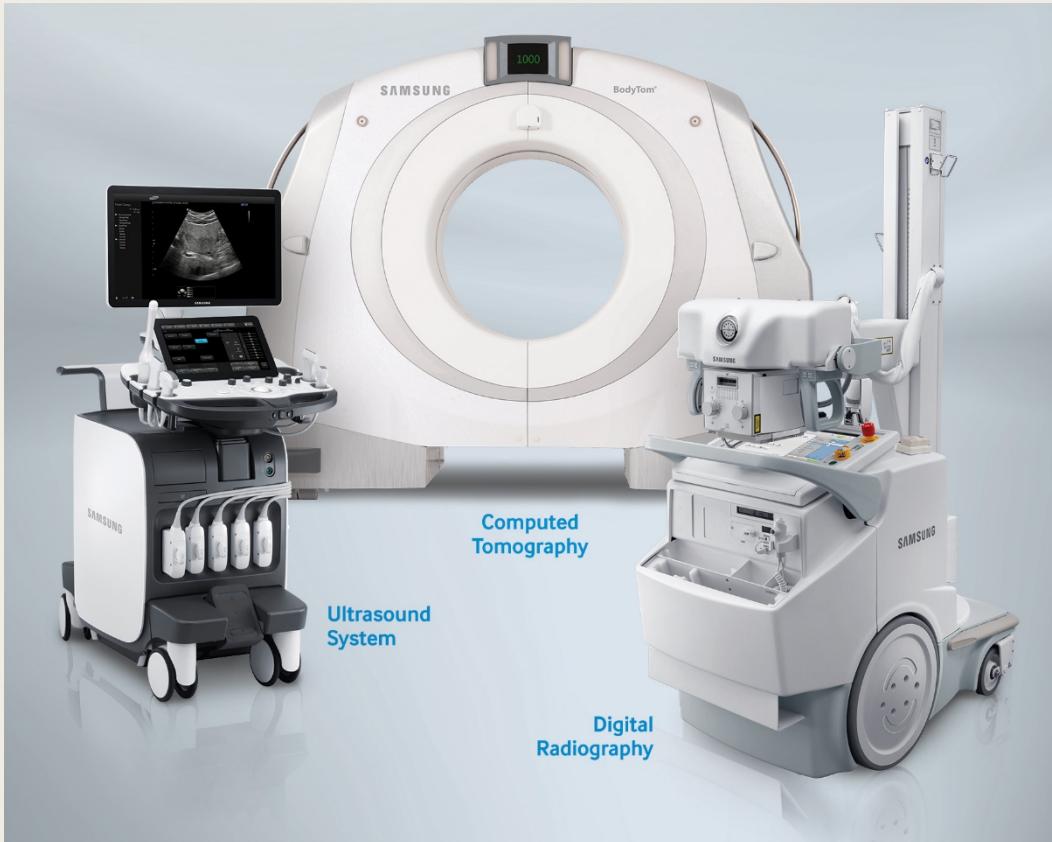


Internet of space and navigation



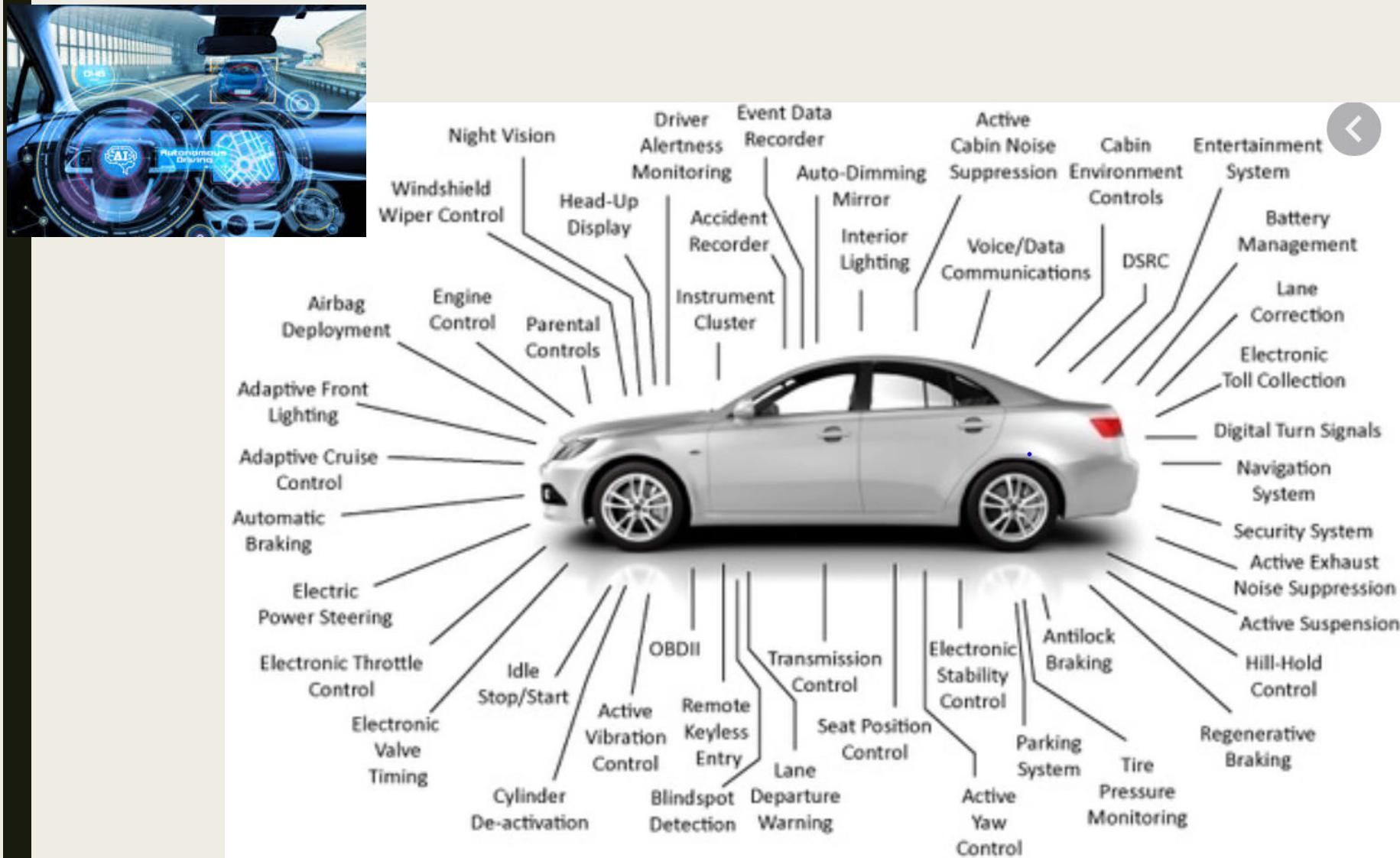
Τα ψηφιακά συστήματα βρίσκονται παντού!

- Μερικά παραδείγματα από την ιατρική:



Τα ψηφιακά συστήματα βρίσκονται παντού!

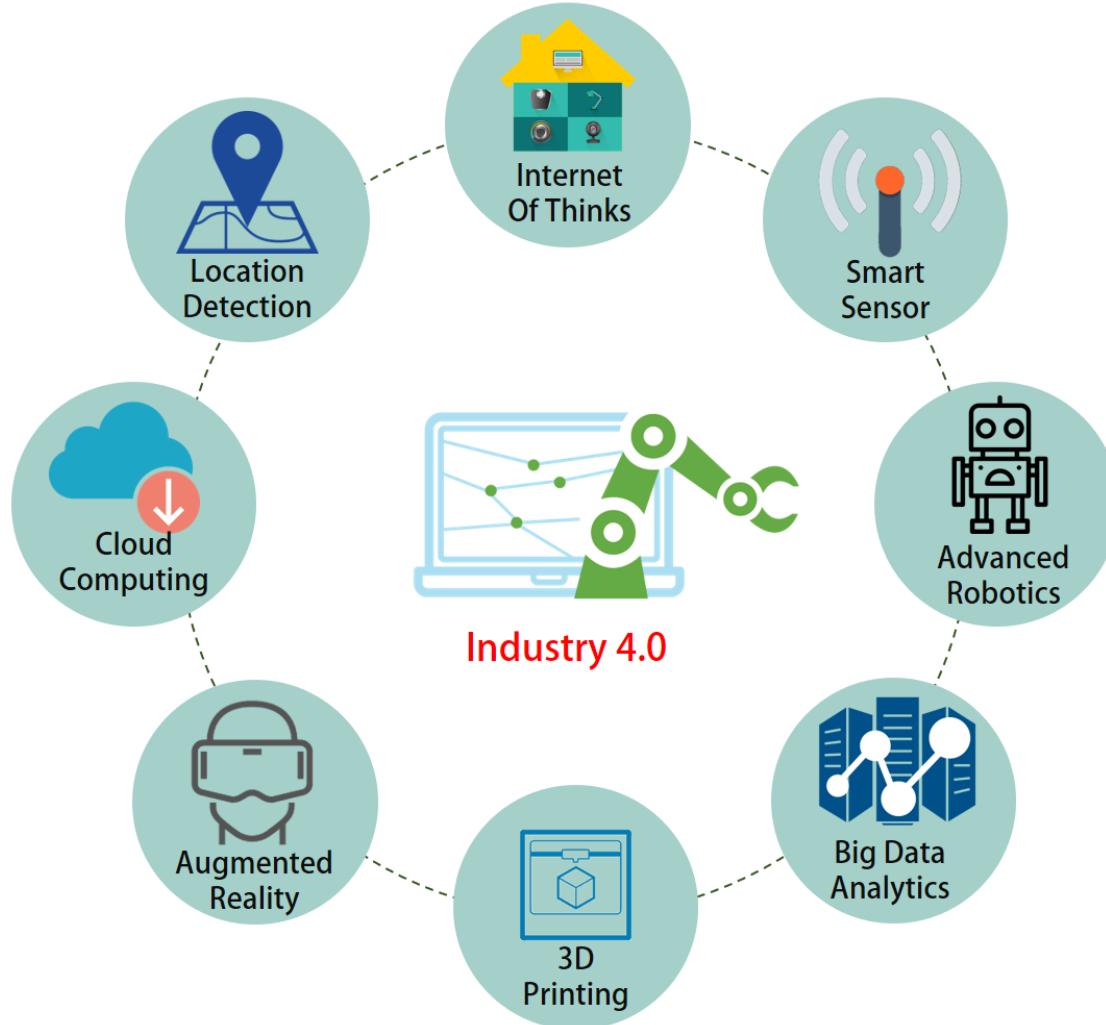
- ## ■ Μερικά παραδείγματα από την αυτοκινητοβιομηχανία (AI, IoT):



Τα ψηφιακά συστήματα βρίσκονται παντού!

- Μερικά παραδείγματα από την ψηφιακή βιομηχανία:

INDUSTRY 4.0 FRAMEWORK – THE DIGITAL TECHNOLOGIES



Robotics automation and machine learning @ industry

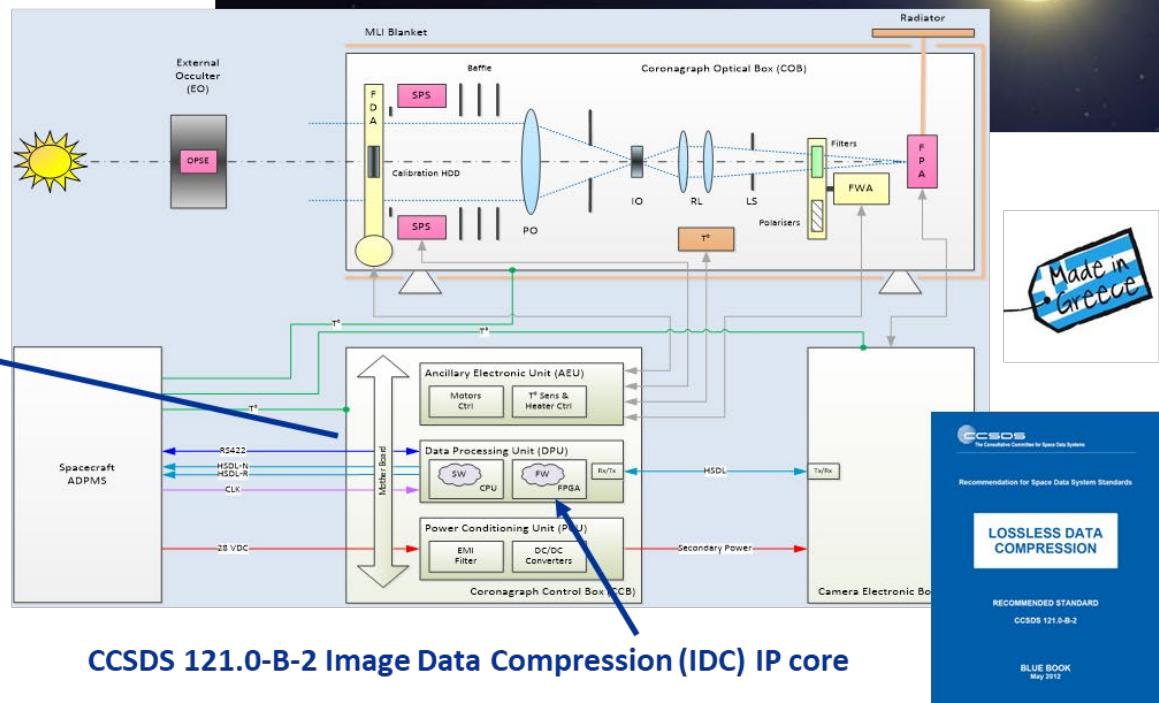
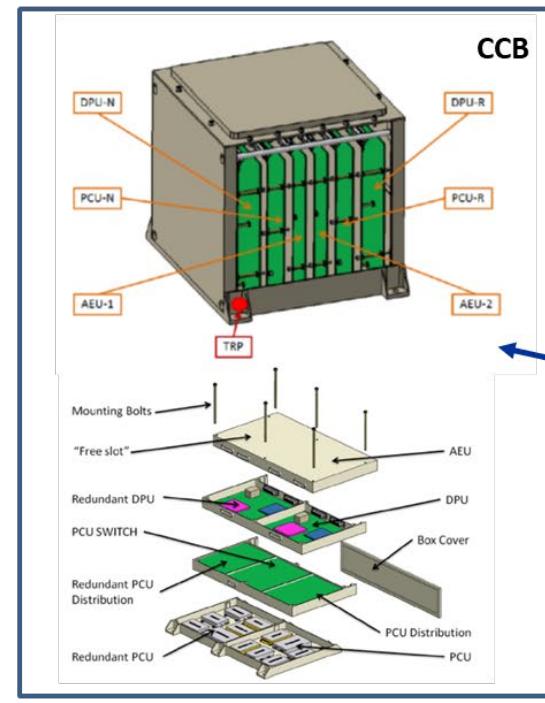
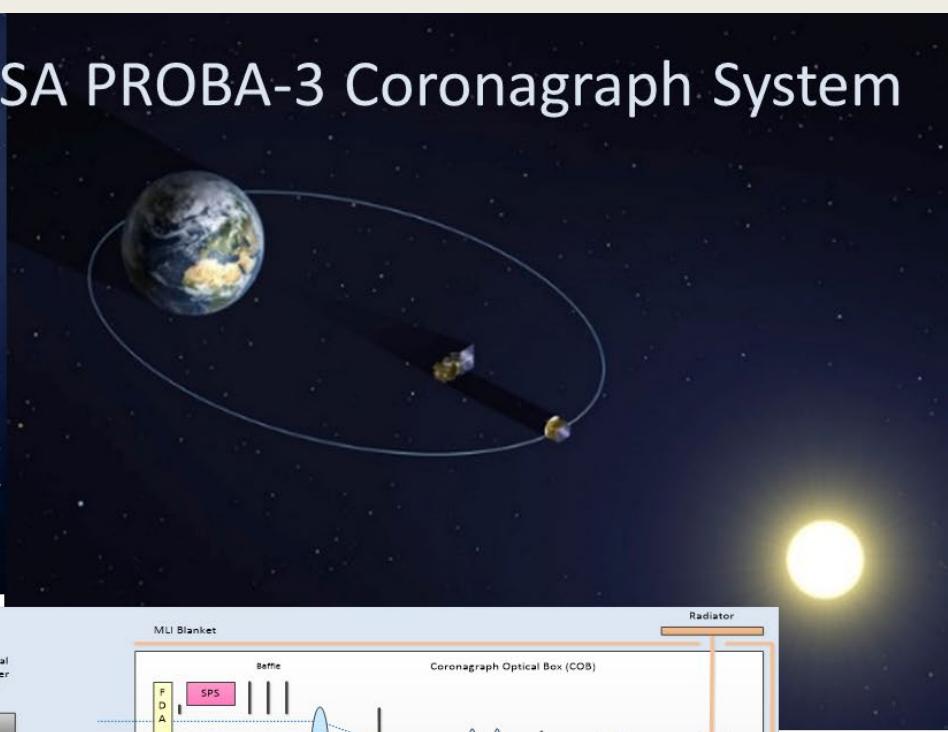


Big data analytics and machine learning @ data centers



dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

ESA PROBA-3 Coronagraph System



CCSDS 121.0-B-2 Image Data Compression (IDC) IP core



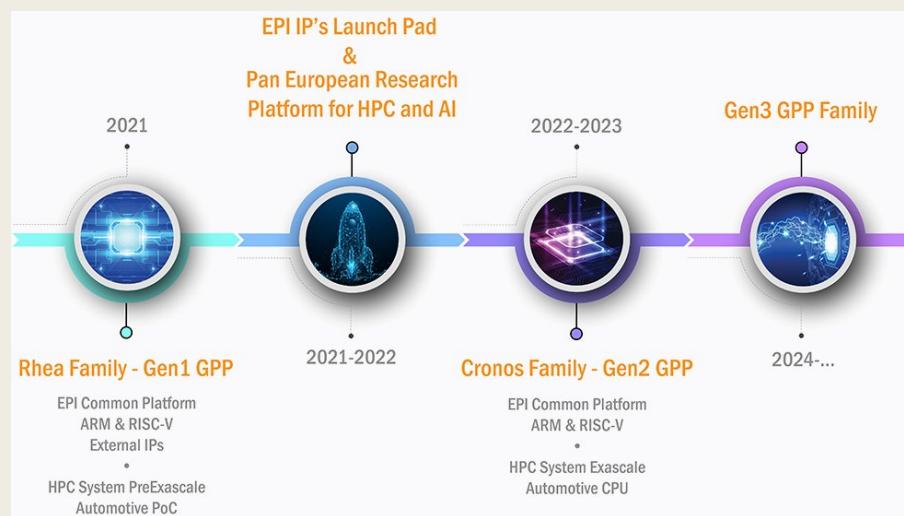
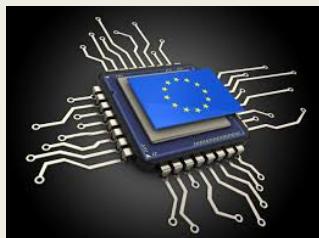
AWS Graviton Processors

BEST PRICE PERFORMANCE IN AMAZON EC2

Graviton
2018

Graviton2
2019

Graviton3
2021





Κεφάλαιο 1

Εισαγωγή στη ψηφιακή σχεδίαση και
τεχνολογία (από το μηδέν στο ένα)

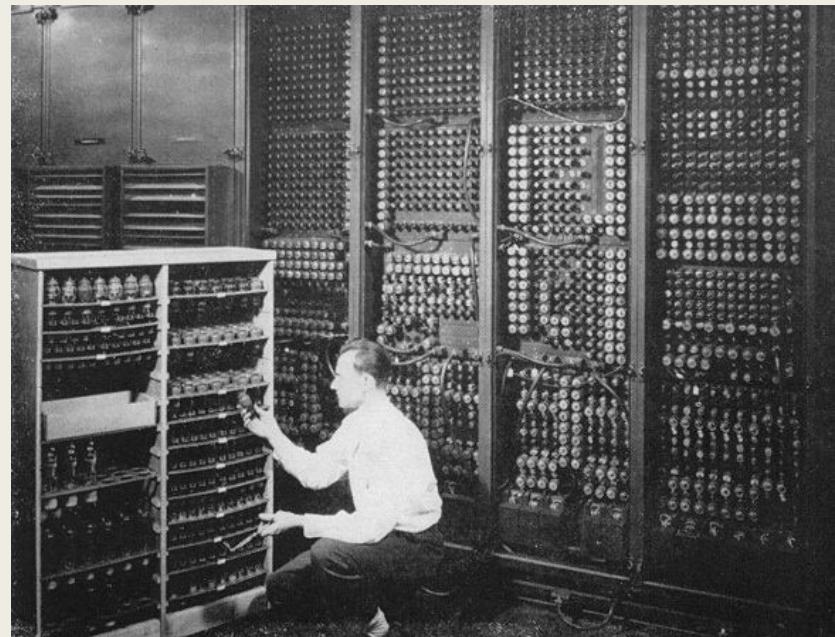
Γιώργος Παπαδημητρίου, Αντώνης Πασχάλης,
Διονύσης Βασιλόπουλος

Περιεχόμενα κεφαλαίου 1

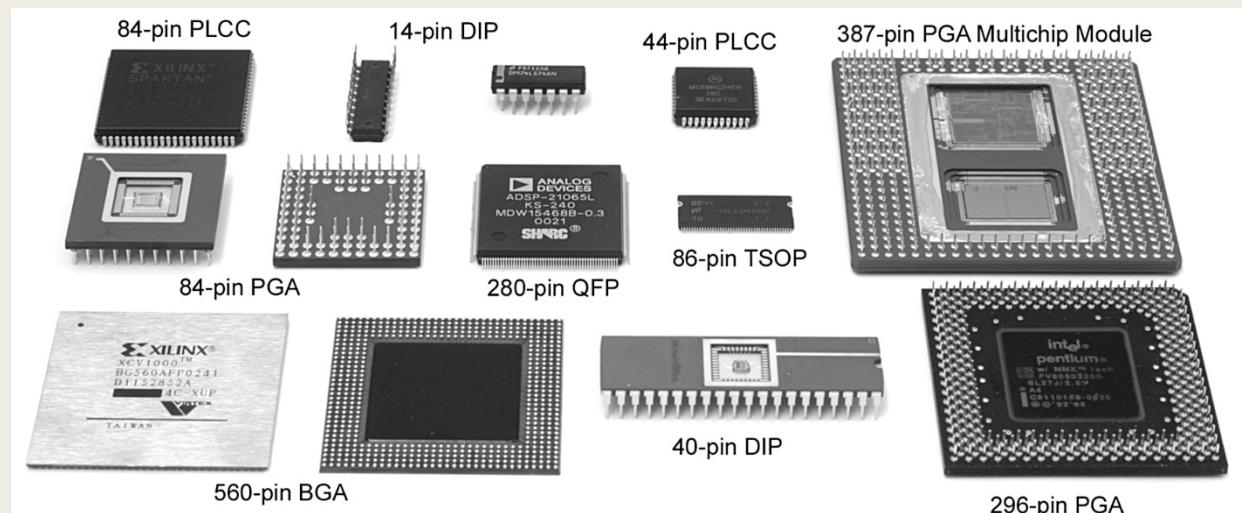
- Εξέλιξη ψηφιακής τεχνολογίας (Εισαγωγή στην Πληροφορική)
- Περί μικροεπεξεργαστών (Εισαγωγή στην Πληροφορική)
- Διαχείριση πολυπλοκότητας υπολογιστικών συστημάτων (Εισαγωγή στην Πληροφορική)
- Ψηφιακή «αφαίρεση» (Εισαγωγή στην Πληροφορική)
- Αναπαράσταση αριθμών σε δυαδική μορφή
- Πράξεις (+, -) μεταξύ δυαδικών αριθμών
- Αθροιστές - Αφαιρέτες
- Λογικές πύλες
- Λογικά επίπεδα – Περιθώρια θορύβου
- Τρανζίστορ CMOS – Μοντέλο διακοπτών – CMOS πύλες
- Κατανάλωση ισχύος

Οι παρουσιάσεις βασίζονται στο βιβλίο με τίτλο «Ψηφιακή σχεδίαση και αρχιτεκτονική υπολογιστών – Έκδοση ARM» των Sarah L. Harris & David Money Harris. Τίτλοι με κόκκινο υποδηλώνουν εκπαιδευτικό υλικό εκτός αυτού του βιβλίου.

- ENIAC Αντικατάσταση λυχνίας
(19.000 πιθανές βλάβες)

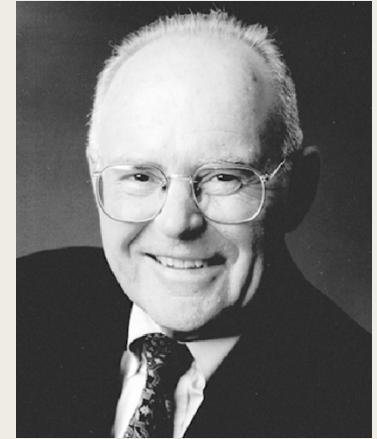


- Ολοκληρωμένα κυκλώματα



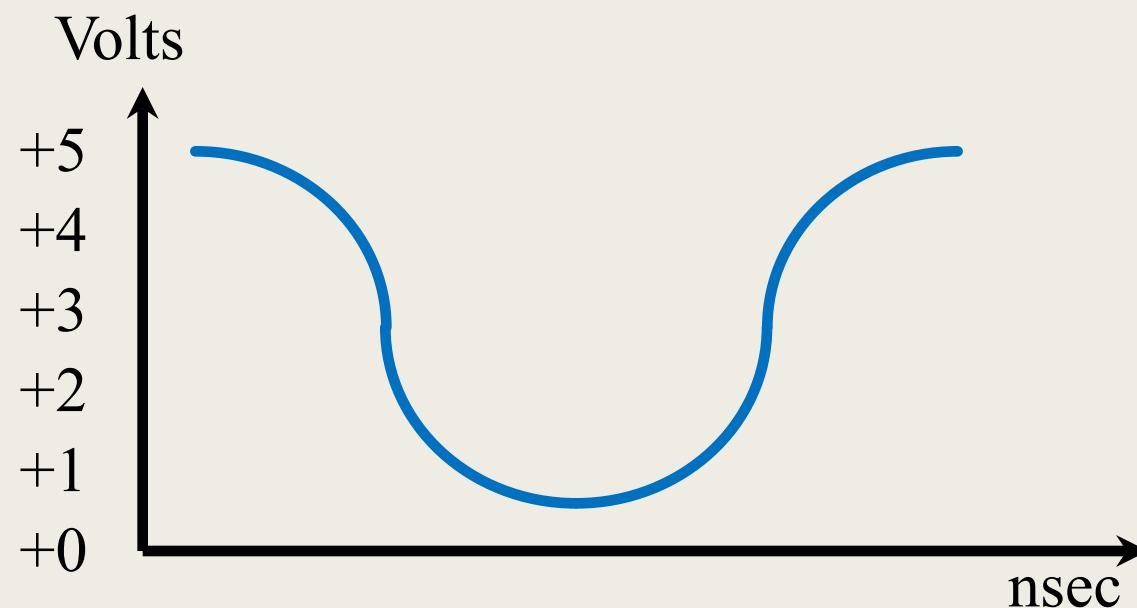
Νόμος του Moore

- Gordon Moore (Σαν Φρανσίσκο, 1929)
- Κάτοχος προπτυχιακού διπλώματος στη Χημεία από το Πανεπιστήμιο Berkeley και διδακτορικού διπλώματος στη Χημεία και τη Φυσική από το Πανεπιστήμιο Caltech
- Το 1968 ίδρυσε την εταιρεία Intel μαζί με τον Robert Noyce
- Το 1965 παρατήρησε ότι **το πλήθος των τρανζίστορ σε ένα τσιπ υπολογιστών διπλασιάζεται κάθε χρόνο**. Αυτή η τάση έχει γίνει ευρύτερα γνωστή ως **Νόμος του Moore**
- Από το 1975 και έπειτα, **το πλήθος των τρανζίστορ διπλασιάζεται κάθε δύο χρόνια**, ενώ **η απόδοση των μικροεπεξεργαστών διπλασιάζεται κάθε 18 με 24 μήνες**
- Το νόμο του Moore ακολουθούν και **οι μνήμες (τεχνολογίας SRAM, DRAM, HDD, SSD)** του ιεραρχικού συστήματος μνήμης, αλλά με μικρότερους ρυθμούς
- Οι **πωλήσεις των ημιαγωγών** εμφανίζουν και αυτές εκθετική αύξηση
- Ο Νόμος του Moore έχει αποτελέσει την κινητήρια δύναμη πίσω από την απίστευτη πρόοδο που έχει σημειωθεί στη **βιομηχανία των ημιαγωγών** τα 50 τελευταία χρόνια, που οδηγεί στη ψηφιακή εποχή



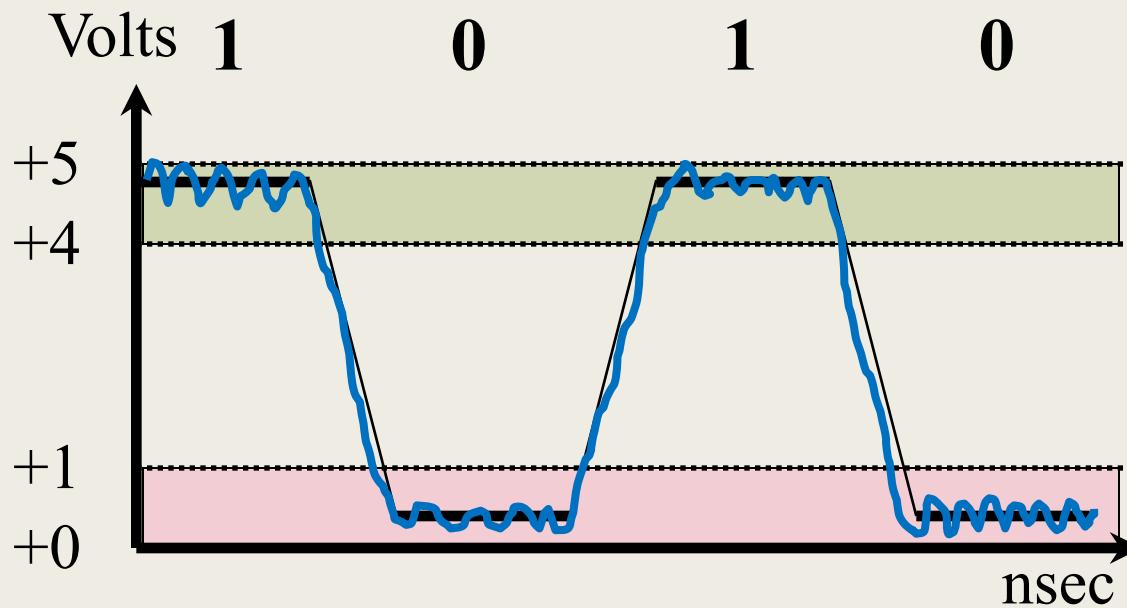
Αναλογικά συστήματα

- Επεξεργάζονται συνεχή ηλεκτρικά σήματα, που μεταβάλλονται σαν συναρτήσεις του χρόνου



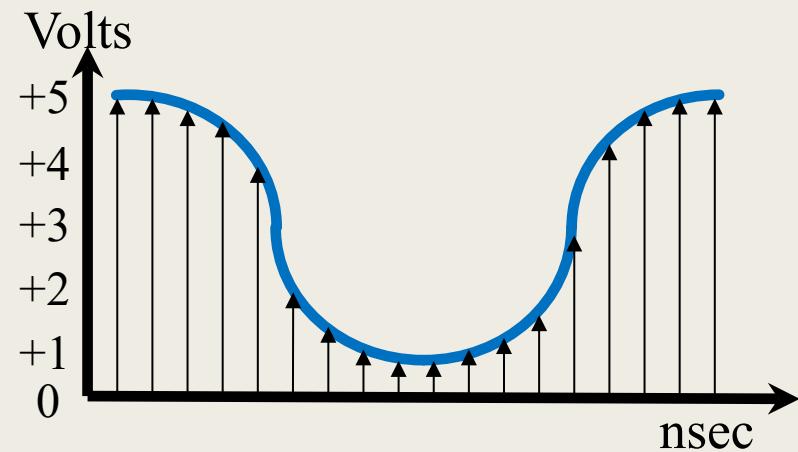
Ψηφιακά συστήματα

- Επεξεργάζονται δυαδικά ηλεκτρικά σήματα
- Τα δυαδικά σήματα λαμβάνουν δύο μόνο τιμές:
 - *To μηδέν (0 ή LOW)*
 - *To ένα (1 ή HIGH)*



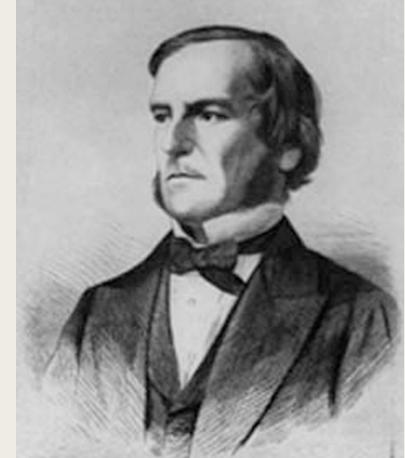
Ψηφιακά συστήματα

- Συστήματα Υπολογιστών
- Συστήματα Επικοινωνιών
- Συστήματα Επεξεργασίας Σήματος
- Συστήματα Αυτομάτου Ελέγχου
 - ψηφιακή επεξεργασία δεδομένων
 - ψηφιακή λήψη δεδομένων
 - ψηφιακή απόκριση
 - αναλογική λήψη δεδομένων
 - μετατροπή της πληροφορίας από αναλογική σε ψηφιακή με Analog to Digital Converter – ADC
(δειγματοληψία + κβαντοποίηση σε ακέραιους αριθμούς 10-16 bit)
 - αναλογική απόκριση
 - μετατροπή της πληροφορίας από ψηφιακή σε αναλογική με Digital to Analog Converter – DAC



George Boole, 1815–1864

- Όντας παιδί εργατών και μην έχοντας την οικονομική δυνατότητα να μορφωθεί μέσω του εκπαιδευτικού συστήματος, ο Boole έμαθε μόνος του μαθηματικά και στη συνέχεια έγινε μέλος του διδακτικού προσωπικού στο Κολέγιο Κουίνς (Ιρλανδία).
- Έγραψε τη μονογραφία **An Investigation of the Laws of Thought (1854)**, όπου παρουσίασε για πρώτη φορά τις δυαδικές μεταβλητές και τις τρεις θεμελιώδεις λογικές πράξεις (logic operations): AND, OR και NOT.
 - Οι δυαδικές μεταβλητές του Boole μπορούσαν να λάβουν τις τιμές TRUE ή FALSE
 - Θεωρούμε ότι είναι συνώνυμοι οι όροι:
 - 1, TRUE, HIGH
 - 0, FALSE, LOW



Μεταφορά πληροφορίας

- Η ποσότητα πληροφορίας D σε μία μεταβλητή διακριτών τιμών με N διαφορετικές καταστάσεις χαρακτηρίζεται από την σχέση
 - $D=\log_2 N$
 - Ο λογάριθμος έχει βάση το 2
 - Η μονάδα μέτρησης του D είναι το **bit**
- Κάθε δυαδική μεταβλητή μεταφέρει $\log_2 2=1$ bit πληροφοριών
- Θεωρητικά, ένα συνεχές σήμα μεταφέρει άπειρη ποσότητα πληροφοριών, αφού μπορεί να πάρει άπειρο πλήθος τιμών
- Στην πράξη, ο θόρυβος και το σφάλμα μέτρησης περιορίζουν τις πληροφορίες που μεταφέρονται στα **10-16 bit**, για τα περισσότερα συνεχή σήματα

Επιλεγμένες ασκήσεις

■ Άσκηση 1.4

Μια αναλογική τάση παίρνει τιμές από 0 έως 5 V. Αν είναι εφικτό να μετρηθεί με ακρίβεια ± 50 mV, πόσα bit πληροφοριών μπορεί να μεταφέρει το μέγιστο;

■ Άσκηση 1.6

Οι Βαβυλώνιοι ανέπτυξαν το εξηνταδικό (sexagesimal, με βάση το 60) αριθμητικό σύστημα πριν από 4000 χρόνια περίπου. Πόσα bit πληροφοριών μεταφέρονται με ένα εξηνταδικό ψηφίο;

Επιλεγμένες ασκήσεις

■ Ασκηση 1.4

Μια αναλογική τάση παίρνει τιμές από 0 έως 5 V. Αν είναι εφικτό να μετρηθεί με ακρίβεια ± 50 mV, πόσα bit πληροφοριών μπορεί να μεταφέρει το μέγιστο;

- Ακρίβεια ± 50 mV σημαίνει ότι το αναλογικό σήμα διαμερίζεται ανά 100 mV, δηλαδή σε 50 διακριτές τιμές από από 0 έως 5 V
- $\log_2 50 = 5.64$ bits

■ Ασκηση 1.6

Οι Βαβυλώνιοι ανέπτυξαν το εξηνταδικό (sexagesimal, με βάση το 60) αριθμητικό σύστημα πριν από 4000 χρόνια περίπου.

Πόσα bit πληροφοριών μεταφέρονται με ένα εξηνταδικό ψηφίο;

- $\log_2 60 = 5.91$ bits

Ερωτήσεις συνεντεύξεων

■ Ερώτηση 1.2

Ένας βασιλιάς παραλαμβάνει 64 χρυσά νομίσματα σε φόρους, αλλά έχει βάσιμες υποψίες ότι ένα από αυτά είναι κάλπικο. Σας καλεί στο παλάτι για να εντοπίσετε το κάλπικο νόμισμα.

Έχετε στη διάθεσή σας μια ζυγαριά παλαιού τύπου με δύο δίσκους στους οποίους μπορείτε να τοποθετείτε νομίσματα.

Πόσες φορές πρέπει να χρησιμοποιήσετε τη ζυγαριά για να βρείτε το ελαφρύτερο, κάλπικο νόμισμα;

- *Επιλέξτε μία από τις απαντήσεις:*
 - (α) 6
 - (β) 5
 - (γ) 4
 - (δ) 3

Ερωτήσεις συνεντεύξεων

■ Ερώτηση 1.2

Ένας βασιλιάς παραλαμβάνει 64 χρυσά νομίσματα σε φόρους, αλλά έχει βάσιμες υποψίες ότι ένα από αυτά είναι κάλπικο. Σας καλεί στο παλάτι για να εντοπίσετε το κάλπικο νόμισμα.

Έχετε στη διάθεσή σας μια ζυγαριά παλαιού τύπου με δύο δίσκους στους οποίους μπορείτε να τοποθετείτε νομίσματα.

Πόσες φορές πρέπει να χρησιμοποιήσετε τη ζυγαριά για να βρείτε το ελαφρύτερο, κάλπικο νόμισμα;

- *Για διαίρεση στα 2, με δύο ζυγαριές A και B*
 - 1η Ζύγιση, Από 32 σε A και B : Έστω ελαφρύτερη η A
 - 2η Ζύγιση. Από 16 σε A και B : Έστω ελαφρύτερη η A
 - 3η Ζύγιση. Από 8 σε A και B : Έστω ελαφρύτερη η A
 - 4η Ζύγιση. Από 4 σε A και B : Έστω ελαφρύτερη η A
 - 5η Ζύγιση. Από 2 σε A και B : Έστω ελαφρύτερη η A
 - 6η Ζύγιση. Από 1 σε A και B : Έστω ελαφρύτερη η A (κάλπικο νόμισμα)

Ερωτήσεις συνεντεύξεων

■ Ερώτηση 1.2

Ένας βασιλιάς παραλαμβάνει 64 χρυσά νομίσματα σε φόρους, αλλά έχει βάσιμες υποψίες ότι ένα από αυτά είναι κάλπικο. Σας καλεί στο παλάτι για να εντοπίσετε το κάλπικο νόμισμα.

Έχετε στη διάθεσή σας μια ζυγαριά παλαιού τύπου με δύο δίσκους στους οποίους μπορείτε να τοποθετείτε νομίσματα.

Πόσες φορές πρέπει να χρησιμοποιήσετε τη ζυγαριά για να βρείτε το ελαφρύτερο, κάλπικο νόμισμα;

- Για διαίρεση στα 3, με δύο ζυγαριές A και B
 - 1η Ζύγιση, Από 22 σε A και B και 20 στην άκρη: Έστω ελαφρύτερη η A
 - 2η Ζύγιση. Από 8 σε A και B και 6 στην άκρη: Έστω ελαφρύτερη η A
 - 3η Ζύγιση. Από 3 σε A και B και 2 στην άκρη : Έστω ελαφρύτερη η A
 - 4η Ζύγιση. Από 1 σε A και B και 1 στην άκρη : Αν $A=B$ τότε κάλπικο αυτό που είναι στην άκρη. Άλλιώς κάλπικο αυτό που είναι το ελαφρύτερο από A και B

(υπάρχουν και άλλες διαιρέσεις π.χ. στην 1^η Ζύγιση 21/21/22, κ.λ.π. που δίνουν το ίδιο αποτέλεσμα)

Ερωτήσεις συνεντεύξεων

■ Ερώτηση 1.2

Ένας βασιλιάς παραλαμβάνει 64 χρυσά νομίσματα σε φόρους, αλλά έχει βάσιμες υποψίες ότι ένα από αυτά είναι κάλπικο. Σας καλεί στο παλάτι για να εντοπίσετε το κάλπικο νόμισμα.

Έχετε στη διάθεσή σας μια ζυγαριά παλαιού τύπου με δύο δίσκους στους οποίους μπορείτε να τοποθετείτε νομίσματα.

Πόσες φορές πρέπει να χρησιμοποιήσετε τη ζυγαριά για να βρείτε το ελαφρύτερο, κάλπικο νόμισμα;

- Για διαίρεση στα 4, με δύο ζυγαριές A και B και στην άκρη τα υπόλοιπα (A περίπτωση worst case scenario)
 - 1η Ζύγιση, Από 16 σε A και B και 32 στην άκρη: Έστω $A=B$, ασχολούμαστε με τα 32 που είναι στην άκρη.
 - 2η Ζύγιση. Από 8 σε A και B και 16 στην άκρη: Έστω $A=B$, ασχολούμαστε με τα 16 που είναι στην άκρη.
 - 3η Ζύγιση. Από 4 σε A και B και 8 στην άκρη : Έστω $A=B$, ασχολούμαστε με τα 8 που είναι στην άκρη.
 - 4η Ζύγιση. Από 2 σε A και B και 4 στην άκρη : Έστω $A=B$, ασχολούμαστε με τα 4 που είναι στην άκρη.
 - 5η Ζύγιση. Από 1 σε A και B και 2 στην άκρη : Έστω $A=B$, ασχολούμαστε με τα 2 που είναι στην άκρη.
 - 6η Ζύγιση. Από 1 σε A και B. Έστω ελαφρύτερη η A (κάλπικο νόμισμα)

Ερωτήσεις συνεντεύξεων

■ Ερώτηση 1.2

Ένας βασιλιάς παραλαμβάνει 64 χρυσά νομίσματα σε φόρους, αλλά έχει βάσιμες υποψίες ότι ένα από αυτά είναι κάλπικο. Σας καλεί στο παλάτι για να εντοπίσετε το κάλπικο νόμισμα.

Έχετε στη διάθεσή σας μια ζυγαριά παλαιού τύπου με δύο δίσκους στους οποίους μπορείτε να τοποθετείτε νομίσματα.

Πόσες φορές πρέπει να χρησιμοποιήσετε τη ζυγαριά για να βρείτε το ελαφρύτερο, κάλπικο νόμισμα;

- Για διαίρεση στα 4, με δύο ζυγαριές A και B και στην άκρη τα υπόλοιπα (B περίπτωση best case scenario)
 - 1η Ζύγιση, Από 16 σε A και B και 32 στην άκρη: Έστω ελαφρύτερη η A.
 - 2η Ζύγιση. Από 4 σε A και B και 8 στην άκρη: Έστω ελαφρύτερη η A.
 - 3η Ζύγιση. Από 1 σε A και B και 2 στην άκρη : Έστω ελαφρύτερη η A (κάλπικο νόμισμα)

Φαίνεται ως πιθανή καλύτερη λύση, αλλά δεν είναι, γιατί δεν εξασφαλίζει ότι ΠΑΝΤΑ με 3 ζυγίσεις βρίσκουμε το σωστό αποτέλεσμα. Μας ενδιαφέρει τι γίνεται στη χειρότερη των περιπτώσεων.

Αριθμητικά Συστήματα

- Δεκαδικό σύστημα αναπαράστασης αριθμών (0-9)

Στήλη των 1
Στήλη των 10
Στήλη των 100
Στήλη των 1000

6598₁₀

Κάθε στήλη ενός δεκαδικού αριθμού έχει δεκαπλάσιο βάρος από την προηγούμενη στήλη.
Ξεκινώντας από τα δεξιά προς τα αριστερά τα βάρη είναι: $10^0, 10^1, 10^2, 10^3, \dots$

- Δυαδικό σύστημα αναπαράστασης αριθμών (0-1)

Στήλη των 1
Στήλη των 2
Στήλη των 4
Στήλη των 8

1101₂

Κάθε στήλη ενός δυαδικού αριθμού έχει διπλάσιο βάρος από την προηγούμενη στήλη.
Ξεκινώντας από τα δεξιά προς τα αριστερά τα βάρη είναι: $2^0, 2^1, 2^2, 2^3, \dots$

Αριθμητικά Συστήματα

- Δεκαδικό σύστημα αναπαράστασης αριθμών (0-9)

ΣΤΗΛΗ ΤΩΝ 1
ΣΤΗΛΗ ΤΩΝ 10
ΣΤΗΛΗ ΤΩΝ 100
ΣΤΗΛΗ ΤΩΝ 1000

$$6598_{10} = 6 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 8 \times 10^0$$

Έξι Πέντε Εννέα Οκτώ
Χιλιάδες Εκατοντάδες Δεκάδες Μονάδες

- Δυαδικό σύστημα αναπαράστασης αριθμών (0-1)

ΣΤΗΛΗ ΤΩΝ 1
ΣΤΗΛΗ ΤΩΝ 2
ΣΤΗΛΗ ΤΩΝ 4
ΣΤΗΛΗ ΤΩΝ 8

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

Μια Μια Μηδέν Μια
Οκτάδα Τετράδα Δυάδες Μονάδα

Δυνάμεις του 2

- | | |
|---------------|--------------------|
| • $2^0 = 1$ | • $2^8 = 256$ |
| • $2^1 = 2$ | • $2^9 = 512$ |
| • $2^2 = 4$ | • $2^{10} = 1024$ |
| • $2^3 = 8$ | • $2^{11} = 2048$ |
| • $2^4 = 16$ | • $2^{12} = 4096$ |
| • $2^5 = 32$ | • $2^{13} = 8192$ |
| • $2^6 = 64$ | • $2^{14} = 16384$ |
| • $2^7 = 128$ | • $2^{15} = 32768$ |

Προσπαθήστε
να θυμάστε
μέχρι το 2^{10}

Παράδειγμα 1.1

- Μετατροπή από το **Δυαδικό** στο **Δεκαδικό** σύστημα
 - Μετατρέψτε τον αριθμό **10110_2** στο δεκαδικό

Παράδειγμα 1.1

- Μετατροπή από το **Δυαδικό** στο **Δεκαδικό** σύστημα
 - Μετατρέψτε τον αριθμό **10110_2** στο δεκαδικό

$$16 \times 1 + 8 \times 0 + 4 \times 1 + 2 \times 1 + 1 \times 0 = \textcolor{blue}{22}_{10}$$

Παράδειγμα 1.2

- Μετατροπή από το Δεκαδικό στο Δυαδικό σύστημα
 - Μετατρέψτε τον αριθμό 84_{10} στο δυαδικό

Μετατροπή από το δεκαδικό στο δυαδικό σύστημα

■ Μέθοδος 1: Από αριστερά προς τα δεξιά

1. Βρίσκω την μεγαλύτερη δύναμη του 2 που χωράει (\leq) στον αριθμό
2. Εάν χωράει, στη στήλη αυτής της δύναμης το αντίστοιχο ψηφίο του δυαδικού αριθμού είναι 1, αλλιώς είναι 0
3. Αφαιρώ τη δύναμη του 2 από τον αριθμό
4. Επαναλαμβάνω

Μετατροπή από το δεκαδικό στο δυαδικό σύστημα

Μέθοδος 1: Από αριστερά προς τα δεξιά

$$84_{10}$$

$$84 - 64 = 20_{10}$$

$$20_{10}$$

$$20 - 16 = 4_{10}$$

$$4_{10}$$

$$4 - 4 = 0_{10}$$

$$0_{10}$$

$$2^6 = 64 < 84$$

$$2^5 = 32 > 20$$

$$2^4 = 16 < 20$$

$$2^3 = 8 > 4$$

$$2^2 = 4 = 4$$

$$2^1 = 2 > 0$$

$$2^0 = 1 > 0$$

$$64 \times 1 -$$

$$32 \times 0$$

$$16 \times 1$$

$$8 \times 0$$

$$4 \times 1$$

$$2 \times 0$$

$$1 \times 0$$

$$= 1010100_2$$

Μετατροπή από το δεκαδικό στο δυαδικό σύστημα

■ Μέθοδος 2: Από δεξιά προς τα αριστερά

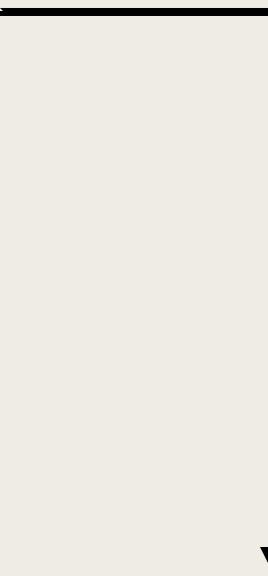
1. Επαναληπτικά διαιρώ με το 2
2. Βάζω το υπόλοιπο (0 ή 1) ως ψηφίο του δυαδικού αριθμού από δεξιά προς τα αριστερά
3. Επαναλαμβάνω

Μετατροπή από το δεκαδικό στο δυαδικό σύστημα

Μέθοδος 2: Από δεξιά προς τα αριστερά (διαίρεση δια 2)

$$\begin{array}{rcl} 84_{10} = & 84/2 = 42 & \text{Υπ. 0} \\ & 42/2 = 21 & \text{Υπ. 0} \\ & 21/2 = 10 & \text{Υπ. 1} \\ & 10/2 = 5 & \text{Υπ. 0} \\ & 5/2 = 2 & \text{Υπ. 1} \\ & 2/2 = 1 & \text{Υπ. 0} \\ & 1/2 = 0 & \text{Υπ. 1} \end{array}$$

= 1010100_2



Εύρος τιμών ενός αριθμού με N ψηφία

■ Δεκαδικός Αριθμός N ψηφίων

- Πόσες τιμές μπορεί να πάρει? 10^N
- Εύρος? $[0, 10^N - 1]$
- Παράδειγμα: Δεκαδικός αριθμός 3 ψηφίων:
 - $10^3 = 1000$ πιθανές τιμές
 - Εύρος: $[0, 999]$

■ Δυαδικός Αριθμός N ψηφίων

- Πόσες τιμές μπορεί να πάρει? 2^N
- Εύρος? $[0, 2^N - 1]$
- Παράδειγμα: Δυαδικός αριθμός 3 ψηφίων:
 - $2^3 = 8$ πιθανές τιμές
 - Εύρος: $[0, 7] = [000_2 \text{ to } 111_2]$

Δεκαεξαδικοί Αριθμοί (hexadecimal)

- Η βάση είναι το 16
- Ευκολότερη αναπαράσταση των Δυαδικών αριθμών

Δεκαεξαδικό	Δεκαδικό	Δυαδικό
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	

Δεκαεξαδικό	Δεκαδικό	Δυαδικό
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	

Δεκαεξαδικοί Αριθμοί (hexadecimal)

- Η βάση είναι το 16
- Ευκολότερη αναπαράσταση των Δυαδικών αριθμών

Δεκαεξαδικό	Δεκαδικό	Δυαδικό
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111

Δεκαεξαδικό	Δεκαδικό	Δυαδικό
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Αριθμητικά Συστήματα

- Δεκαεξαδικό σύστημα αναπαράστασης αριθμών (0-9,A-F)

ΣΤΗΛΗ ΤΩΝ 1
ΣΤΗΛΗ ΤΩΝ 16
ΣΤΗΛΗ ΤΩΝ 256

Κάθε στήλη ενός δεκαεξαδικού αριθμού έχει δεκαεξαδικό βάρος από την προηγούμενη στήλη. Ξεκινώντας από τα δεξιά προς τα αριστερά τα βάρη είναι: **16⁰, 16¹, 16²=256, 16³=4096, ...**

$$2ED_{16} = 2 \times 16^2 + E \times 16^1 + D \times 16^0$$

Δύο
διακοσιο-
πενηνταεξάδες

Δεκατέσσερις
δεκαεξάδες

Δεκατρείς
μονάδες

Παράδειγμα 1.3

- Μετατροπή από το **Δεκεξαδικό** στο **Δεκαδικό** σύστημα
 - Μετατρέψτε τον αριθμό **$2ED_{16}$** στο δεκαδικό

$$2ED_{16} = 2 \times 16^2 + 14 \times 16^1 + 13 \times 16^0 = 749_{10}$$

Δεκαεξαδικό	Δεκαδικό
A	10
B	11
C	12
D	13
E	14
F	15

Παράδειγμα 1.3

- Μετατροπή από το **Δεκαεξαδικό** στο **Δυαδικό** σύστημα
 - Μετατρέψτε τον αριθμό **$2ED_{16}$** στο δυαδικό
- Κάθε δεκαεξαδικό ψηφίο μετατρέπεται σε τέσσερα δυαδικά ψηφία σύμφωνα με τον πίνακα
$$2ED_{16} = 0010\ 1110\ 1101 = \textcolor{blue}{001011101101}_2$$

Δεκαεξαδικό	Δυαδικό
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Δεκαεξαδικό	Δυαδικό
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Παράδειγμα 1.4

- Μετατροπή από το Δυαδικό στο Δεκαεξαδικό σύστημα
 - Μετατρέψτε τον αριθμό 1111010_2 στο δεκαεξαδικό
- Ξεκινώντας από δεξιά προς τα αριστερά χωρίζουμε σε τετράδες δυαδικών ψηφίων
- Η περισσότερο σημαντική τετράδα (στα αριστερά) μπορεί να έχει λιγότερα από τέσσερα ψηφία, οπότε συμπληρώνεται με **μηδενικά στα αριστερά**
- Κάθε τετράδα δυαδικών ψηφίων μετατρέπεται σε ένα δεκαεξαδικό ψηφίο σύμφωνα με τον προηγούμενο πίνακα

$$1111010_2 = 0111_2 \ 1010_2 = 7_{16} \ A_{16} = 7A_{16}$$

Παράδειγμα 1.5

- Μετατροπή από το **Δεκαδικό** στο **Δεκαεξαδικό** σύστημα
 - Μετατρέψτε τον αριθμό **333₁₀** στο δεκαεξαδικό

Αρχικά μετατρέπουμε τον δεκαδικό αριθμό στο δυαδικό και στη συνέχεια μετατρέπουμε το δυαδικό αριθμό στο δεκαεξαδικό (απλούστερη μέθοδος με πιο απλές πράξεις)

$$333_{10} = 333/2 = 166 \text{ Yπ. 1}$$

$$166/2 = 83 \text{ Yπ. 0}$$

$$83/2 = 41 \text{ Yπ. 1}$$

$$41/2 = 20 \text{ Yπ. 1}$$

$$20/2 = 10 \text{ Yπ. 0}$$

$$10/2 = 5 \text{ Yπ. 0}$$

$$5/2 = 2 \text{ Yπ. 1}$$

$$2/2 = 1 \text{ Yπ. 0}$$

$$1/2 = 0 \text{ Yπ. 1} = \color{green}{101001101}_2$$

$$\color{green}{101001101}_2 = 0001\ 0100\ 1101 = \color{red}{14D_{16}}$$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.13 / 1.15

Μετατρέψτε τους ακόλουθους μη προσημασμένους δυαδικούς αριθμούς σε δεκαδικούς και σε δεκαεξαδικούς. Δείξτε αναλυτικά τη μετατροπή.

$$(\beta) \quad 110110_2 \quad (\gamma) \quad 11110000_2$$

■ Άσκηση 1.17

Μετατρέψτε τους ακόλουθους δεκαεξαδικούς αριθμούς σε δεκαδικούς. Δείξτε αναλυτικά τη μετατροπή.

$$(\alpha) \quad A5_{16} \quad (\gamma) \quad FFFF_{16}$$

■ Άσκηση 1.25 / 1.27

Μετατρέψτε τους ακόλουθους δεκαδικούς αριθμούς σε μη προσημασμένους δυαδικούς και στη συνέχεια σε δεκαεξαδικούς.

$$(\beta) \quad 63_{10} \quad (\delta) \quad 845_{10}$$

Byte, nibble και λέξεις

- **Byte** ονομάζεται μία ομάδα των οκτώ bits
 - Αναπαριστά μία από $2^8 = 256$ πιθανές τιμές
 - Χρησιμοποιείται ως μέγεθος αποθήκευσης στη μνήμη
- **Nibble** ονομάζεται μία ομάδα των τεσσάρων bits ή μισού byte
 - Αναπαριστά μία από $2^4 = 16$ πιθανές τιμές
 - Σε ένα δεκαεξαδικό ψηφίο μπορεί να αποθηκευτεί ένα nibble, ενώ σε δύο δεκαεξαδικά ψηφία μπορεί να αποθηκευτεί ένα πλήρες byte
- Οι μικροεπεξεργαστές χειρίζονται δεδομένα σε τεμάχια τα οποία ονομάζονται **λέξεις** (words).
 - Το μέγεθος μιας λέξης εξαρτάται από την αρχιτεκτονική του μικροεπεξεργαστή
 - Στα σύγχρονα υπολογιστικά συστήματα χρησιμοποιούνται μικροεπεξεργαστές των 64 ή των 32 bit.
 - Ενσωματωμένοι επεξεργαστές σε απλές οικιακές συσκευές χρησιμοποιούν λέξεις των 8 ή των 16 bit

Περισσότερο & Λιγότερο Σημαντικά Bit/Byte

- Σε μία ομάδα από bits
 - το πιο αριστερό bit ονομάζεται **περισσότερο σημαντικό bit** (*most significant bit - MSB*)
 - το bit που βρίσκεται στο άλλο άκρο δεξιά λέγεται **λιγότερο σημαντικό bit** (*least significant bit - LSB*)
- Ομοίως, στο εσωτερικό μιας λέξης
 - το πιο αριστερό byte ονομάζεται **περισσότερο σημαντικό byte**
 - το byte που βρίσκεται στο δεξί άκρο λέγεται **λιγότερο σημαντικό byte**

101100

περισσότερο σημαντικό bit λιγότερο σημαντικό bit

(α)

DEAFDAD8

περισσότερο σημαντικό byte λιγότερο σημαντικό byte

(β)

Bytes, bits/sec και διαβαθμίσεις

- Ο όρος **kilo** (χιλιάδα) αντιστοιχεί στο $2^{10} \approx 10^3$
- Ο όρος **mega** (εκατομμύριο) αντιστοιχεί στο $2^{20} \approx 10^6$
- Ο όρος **giga** (δισεκατομμύριο) αντιστοιχεί στο $2^{30} \approx 10^9$
- Η χωρητικότητα της μνήμης συνήθως μετριέται σε **bytes**
 - *Τα 2^{10} bytes είναι ένα kilobyte (1KB) δηλαδή 1024 bytes*
 - *Τα 2^{20} bytes είναι ένα megabyte (1MB) δηλαδή $1024 \times 1024 = 1.048.576$ bytes*
 - *Τα 2^{30} bytes είναι ένα gigabyte (1GB) δηλαδή $1024 \times 1024 \times 1024 = 1.073.741.824$ bytes*
- Η ταχύτητα των καναλιών επικοινωνίας μετριέται σε **bits/sec (bps)**
 - *Τα 2^{10} bps είναι ένα kilobit/sec (1Kbps) δηλαδή 1024 bits/sec*
 - *Τα 2^{20} bps είναι ένα megabit/sec (1Mbps) δηλαδή $1024 \times 1024 = 1.048.576$ bits/sec*
 - *Τα 2^{30} bps είναι ένα gigabit/sec (1Gbps) δηλαδή $1024 \times 1024 \times 1024 = 1.073.741.824$ bits/sec*

Επιλεγμένες ασκήσεις

■ Ασκηση 1.45

Ένα συγκεκριμένο μόντεμ DSL λειτουργεί στα 768 Kbit/sec.
Πόσα Kbyte μπορεί να λάβει σε 1 λεπτό;

Επιλεγμένες ασκήσεις

■ Ασκηση 1.45

Ένα συγκεκριμένο μόντεμ DSL λειτουργεί στα 768 Kbit/sec.

Πόσα Kbyte μπορεί να λάβει σε 1 λεπτό;

- Σε ένα λεπτό μπορεί να λάβει $60 * 768 \text{Kbit} = > 46080 \text{Kbit/min}$
- $1 \text{ Byte} = 8 \text{ bit} = > 46080 \text{Kbit/min} = 5760 \text{Kbyte/min}$

Επιλεγμένες ασκήσεις

■ Ασκηση 1.47

Όταν οι κατασκευαστές σκληρών δίσκων χρησιμοποιούν τους όρους «megabyte» και «gigabyte», εννοούν 10^6 byte και 10^9 byte, αντίστοιχα. Πόσα πραγματικά GB μουσικής μπορείτε να αποθηκεύσετε σε έναν σκληρό δίσκο με χωρητικότητα 50 GB;

Επιλεγμένες ασκήσεις

■ Ασκηση 1.47

Όταν οι κατασκευαστές σκληρών δίσκων χρησιμοποιούν τους όρους «megabyte» και «gigabyte», εννοούν 10^6 byte και 10^9 byte, αντίστοιχα. Πόσα πραγματικά GB μουσικής μπορείτε να αποθηκεύσετε σε έναν σκληρό δίσκο με χωρητικότητα 50 GB;

- $50GB=50.000.000.000 \text{ bytes}$
- $1 \text{ (real) kbyte}=1024 \text{ bytes}=> 50.000.000.000 \text{ bytes} = (50.000.000.000 / 1024) \text{ kbytes}=48.828.125 \text{ Kbytes}$
- $1 \text{ (real) Mbyte}=1024 \text{ Kbyte}=> 48.828.125 \text{ Kbytes} = (48.828.125 / 1024) \text{ Mbytes}=47.683,72 \text{ Mbytes}$
- $1 \text{ (real) Gbyte}=1024 \text{ Mbyte}=> 47.683,72 \text{ Mbytes} = (47.683,72 / 1024) \text{ Gbytes}=46,566 \text{ Gbytes}$

Παράδειγμα 1.6

- Υπολογισμός κατά προσέγγιση δυνάμεων του 2 στο δεκαδικό σύστημα
 - Βρείτε κατά προσέγγιση την τιμή του 2^{24}
 - Χωρίζουμε τον εκθέτη σε δύο μέρη
 - ένα πολλαπλάσιο του δέκα
 - και το υπόλοιπο
- $$2^{24} = 2^{20} \times 2^4$$
- $$2^{20} = 2^{10} * 2^{10} \approx 10^3 * 10^3 = 1000 * 1000 = 1 \text{ εκατομμύριο.}$$
- $$2^4 = 16.$$

Παράδειγμα 1.6

- Υπολογισμός κατά προσέγγιση δυνάμεων του 2 στο δεκαδικό σύστημα
 - Βρείτε κατά προσέγγιση την τιμή του 2^{24}
 - Χωρίζουμε τον εκθέτη σε δύο μέρη
 - ένα πολλαπλάσιο του δέκα
 - και το υπόλοιπο
- $$2^{24} = 2^{20} \times 2^4$$
- $$2^{20} = 2^{10} * 2^{10} \approx 10^3 * 10^3 = 1000 * 1000 = 1 \text{ εκατομμύριο.}$$
- $$2^4 = 16.$$
- Άρα, $2^{24} \approx 16$ εκατομμύρια (καλή προσέγγιση)
- Με ακρίβεια είναι $2^{24} = 16.777.216$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.48

Εκτιμήστε την τιμή του 2^{31} χωρίς τη χρήση αριθμομηχανής

■ Άσκηση 1.49

Μια μνήμη στον μικροεπεξεργαστή Pentium II είναι οργανωμένη ως ορθογώνια διάταξη bit με 2^8 γραμμές και 2^9 στήλες. Εκτιμήστε πόσα bit (σε χιλιάδες) διαθέτει η μνήμη χωρίς να χρησιμοποιήσετε αριθμομηχανή

Επιλεγμένες ασκήσεις

■ Άσκηση 1.48

- Εκτιμήστε την τιμή του 2^{31} χωρίς τη χρήση αριθμομηχανής
- 2 Δις

■ Άσκηση 1.49

- Μια μνήμη στον μικροεπεξεργαστή Pentium II είναι οργανωμένη ως ορθογώνια διάταξη bit με 2^8 γραμμές και 2^9 στήλες. Εκτιμήστε πόσα bit (σε χιλιάδες => Kbyte) διαθέτει η μνήμη χωρίς να χρησιμοποιήσετε αριθμομηχανή

- $2^9 * 2^8 = 2^{17} = 2^{10} * 2^7 \approx 1000 * 2^7$
- $2^9 * 2^8 \text{ bit} \approx 1000 * 2^7 \text{ bit} = 2^7 \text{ Kbit} = 128 \text{ Kbit}$
 $= 16 \text{ Kbyte}$

Ερωτήσεις συνεντεύξεων

■ Ερώτηση 1.3

Ένας καθηγητής, ένας βοηθός διδασκαλίας, ένας φοιτητής σχεδίασης ψηφιακών συστημάτων και ένας πρωτοετής που είναι το αστέρι της ομάδας στίβου του πανεπιστημίου πρέπει να διασχίσουν μια ετοιμόρροπη γέφυρα κατά τη διάρκεια μιας νύχτας χωρίς φεγγάρι. Η γέφυρα είναι τόσο ασταθής που μόνο δύο άτομα μπορούν να τη διασχίζουν ταυτόχρονα και θα πρέπει να έχουν μαζί τους φακό για να βλέπουν.

Οι τέσσερις πρωταγωνιστές έχουν μόνο έναν φακό στη διάθεσή τους, και η απόσταση της γέφυρας από τη μία άκρη έως την άλλη είναι υπερβολικά μεγάλη για να πετούν τον φακό, οπότε κάποιος πρέπει να τον μεταφέρει πίσω στους άλλους.

Το αστέρι του στίβου μπορεί να διασχίσει τη γέφυρα σε 1 λεπτό. Ο φοιτητής σχεδίασης ψηφιακών συστημάτων μπορεί να τη διασχίσει σε 2 λεπτά. Ο βοηθός διδασκαλίας μπορεί να τη διασχίσει σε 5 λεπτά. Ο καθηγητής πάντα αφαιρείται, με αποτέλεσμα να χρειάζεται 10 λεπτά για να διασχίσει τη γέφυρα.

Ποιος είναι ο ταχύτερος χρόνος που απαιτείται για να μεταβούν όλοι οι πρωταγωνιστές μας από τη μία άκρη της γέφυρας στην άλλη;

Ερωτήσεις συνεντεύξεων

■ Ερώτηση 1.3 (Απάντηση)

1. Αρχικά διασχίζουν τη γέφυρα οι 2 πιο γρήγοροι (δρομέας, φοιτητής) σε 2 λεπτά.
2. Ο πιο γρήγορος (δρομέας) επιστρέφει το φακό σε 1 λεπτό
3. Οι δύο πιο αργοί (βοηθός, καθηγητής) διασχίζουν τη γέφυρα σε 10 λεπτά.
4. Το φακό τον επιστρέφει ο φοιτητής (που είναι ο πιο γρήγορος από όσους έχουν διασχίσει τη γέφυρα) σε 2 λεπτά
5. Ο δρομέας και ο φοιτητής διασχίζουν τη γέφυρα σε 2 λεπτά.

Σύνολο: 17 λεπτά.

https://en.wikipedia.org/wiki/Bridge_and_torch_problem

Πρόσθεση δυαδικών αριθμών

- Ακριβώς όπως και στο δεκαδικό σύστημα πρόσθέτουμε τα ψηφία της ίδιας στήλης (ή αλλιώς ίδιου βάρους) μαζί με το bit κρατούμενου (αν υπάρχει)
- Στο **δεκαδικό**

$$\begin{array}{r} \textcolor{red}{11} & \leftarrow \text{κρατούμενο} \\ 4277 \\ + 5499 \\ \hline 9776 \end{array}$$

- Στο **δυαδικό**

Παράγεται bit κρατούμενου:

$$\begin{aligned} 1+1 &= 2_{10} = 10_2 \\ 1+1+1 &= 3_{10} = 11_2 \end{aligned}$$

$$\begin{array}{r} \textcolor{blue}{11} & \leftarrow \text{κρατούμενο} \\ 1011 \\ + 0011 \\ \hline 1110 \end{array}$$

Υπερχείλιση μη προσημασμένων

- Ενας **μη προσημασμένος (unsigned)** δυαδικός αριθμός με **N bit** μπορεί να πάρει τιμές από το διάστημα $[0, 2^N - 1]$
 - Υπάρχει η περίπτωση το αποτέλεσμα μιας πρόσθεσης να χρειάζεται **N+1 bit** για αναπαρασταθεί
 - Σε αυτή την περίπτωση το *bit* που βρίσκεται στην **N+1** θέση **αγνοείται** και έτσι τα υπόλοιπα **N bits** αναπαριστούν ένα λάθος αποτέλεσμα
 - Αυτό το φαινόμενο λέγεται **υπερχείλιση**
 - μπορεί να ανιχθευθεί μέσω του ελέγχου για κάποιο **κρατούμενο εξόδου (carry)**, το οποίο παράγεται στη στήλη του πιο σημαντικού *bit*
- Όταν κάνουμε πράξεις με δυαδικούς αριθμούς των **N bit**, συνήθως είναι ζητούμενο το αποτέλεσμα να αποθηκεύεται επίσης σε αριθμό των **N bit**.

Δυαδικές κωδικοποιήσεις με 4 bit

- Μη προσημασμένοι αριθμοί
- Εύρος τιμών στα 4 bit: [0, $2^4 - 1$]

Δεκαδικός	Μη προσημα σμένος
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Δεκαδικός	Μη προσημα σμένος
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Παράδειγμα 1.7

- Προσθέστε τους ακόλουθους μη προσημασμένους δυαδικούς αριθμούς μεγέθους 4 bit (Παράδειγμα 1.7)

$$\begin{array}{r} 0111 \\ + \quad 0101 \\ \hline \end{array}$$

Παράδειγμα 1.7

- Προσθέστε τους ακόλουθους μη προσημασμένους δυαδικούς αριθμούς μεγέθους 4 bit (Παράδειγμα 1.7)

$$\begin{array}{r} \textcolor{blue}{111} & \leftarrow \text{κρατούμενο} \\ 0111 \\ + \quad 0101 \\ \hline \textcolor{blue}{1100} \end{array}$$

- Επαλήθευση $0111_2 = 7_{10}$, $0101_2 = 5_{10}$, $1100_2 = 12_{10}$

Παράδειγμα 1.8

- Προσθέστε τους ακόλουθους μη προσημασμένους δυαδικούς αριθμούς μεγέθους 4 bit (Παράδειγμα 1.8)

$$\begin{array}{r} 1101 \\ + 0101 \\ \hline \end{array}$$

Παράδειγμα 1.8

- Προσθέστε τους ακόλουθους μη προσημασμένους δυαδικούς αριθμούς μεγέθους 4 bit (Παράδειγμα 1.8)
 - Επαλήθευση $1101_2 = 13_{10}$, $0101_2 = 5_{10}$, $10010_2 = 18_{10}$

$$\begin{array}{r} \textcolor{blue}{11\ 1} & \leftarrow \text{κρατούμενο} \\ 1101 \\ + 0101 \\ \hline 0010 \end{array}$$

Παράδειγμα 1.8

- Προσθέστε τους ακόλουθους μη προσημασμένους δυαδικούς αριθμούς μεγέθους 4 bit (Παράδειγμα 1.8)
 - Επαλήθευση $1101_2 = 13_{10}$, $0101_2 = 5_{10}$, $10010_2 = 18_{10}$

$$\begin{array}{r} \textcolor{blue}{11\ 1} & \leftarrow \text{κρατούμενο (Carry)} \\ 1101 \\ + 0101 \\ \hline \textcolor{red}{1}0010 \end{array}$$

Υπερχείλιση: 2 αντί 18!
(Overflow)

Συνήθως όμως θέλουμε οι αριθμοί που προσθέτουμε και το αποτέλεσμα να έχουν ίδιο αριθμό από bit. Σε αυτή την περίπτωση δεν μπορούμε να αποθηκεύσουμε το 5^o bit στο αποτέλεσμα και έχουμε Υπερχείλιση : Αποτέλεσμα 2 αντί 18!. Αν μπορούσαμε αν έχουμε 5 bit στο αποτέλεσμα, τότε αυτό θα ήταν σωστό!

Προσημασμένοι δυαδικοί αριθμοί

- Μέχρι στιγμής μελετήσαμε **μη προσημασμένους (unsigned)** ακέραιους αριθμούς στο δυαδικό σύστημα αρίθμησης στους οποίους απεικονίζονται μόνο οι **φυσικοί ακέραιοι αριθμοί**.
- Για να απεικονίστουν **αρνητικοί ακέραιοι αριθμοί** στο δυαδικό σύστημα αρίθμησης πρέπει να γίνει χρήση ενός **προσημασμένου (signed) συστήματος αρίθμησης**
- Υπάρχουν αρκετές μέθοδοι για την αναπαράσταση προσημασμένων ακεραίων αριθμών (θετικών και αρνητικών) στο δυαδικό σύστημα. Οι πιο διαδεδομένες σήμερα είναι:
 - **Αριθμοί προσήμου-μεγέθους (sign/magnitude)**
 - **Αριθμοί συμπληρώματος ως προς δύο (two's complement)**

Αριθμοί προσήμου-μεγέθους

- Σε έναν αριθμό προσήμου-μεγέθους των N bit **το πιο σημαντικό bit** του αριθμού υποδηλώνει το πρόσημο του
 - *To '0' στο MSB δηλώνει ότι ο αριθμός είναι **θετικός***
 - *To '1' στο MSB δηλώνει ότι ο αριθμός είναι **αρνητικός***
- Τα υπόλοιπα N-1 bit του αριθμού δηλώνουν το μέγεθος του αριθμού, δηλαδή την **απόλυτη τιμή** του
- Άρα, για έναν αριθμό πρόσημου-μεγέθους N-bit ισχύει
 - *1 bit πρόσημου, N-1 bits μέγεθος (απόλυτη τιμή)*
- Ένας αριθμός προσήμου-μεγέθους των N bit παίρνει τιμές από το κλειστό διάστημα: **$[-(2^{N-1} - 1), 2^{N-1} - 1]$**

Δυαδικές κωδικοποιήσεις με 4 bit

- Προσημασμένοι αριθμοί προσήμου-μεγέθους
- Εύρος τιμών στα 4 bit: $[-(2^3-1), 2^3-1]$

Δεκαδικός	Προσημα σμένος προσήμου μεγέθους
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Δεκαδικός	Προσημα σμένος προσήμου μεγέθους
-0	1000
-1	1001
-2	1010
-3	1011
-4	1100
-5	1101
-6	1110
-7	1111

Παράδειγμα 1.9

- Μετατροπή προσημασμένων δεκαδικών αριθμών σε δυαδικούς αριθμούς προσήμου-μεγέθους
- Οι δεκαδικοί ακέραιοι αριθμοί ± 5 σε αναπαράσταση δυαδικού αριθμού προσήμου-μεγέθους με 4 bit είναι:

Παράδειγμα 1.9

- Μετατροπή προσημασμένων δεκαδικών αριθμών σε δυαδικούς αριθμούς προσήμου-μεγέθους
- Οι δεκαδικοί ακέραιοι αριθμοί ± 5 σε αναπαράσταση δυαδικού αριθμού προσήμου-μεγέθους με 4 bit είναι:
 - Αρχικά βρίσκουμε το μέγεθος τους αριθμού μετατρέποντας τον δεκαδικό αριθμό σε μη προσημασμένο δυαδικό αριθμό
 - $5_{10} = 101_2$

Παράδειγμα 1.9

- Μετατροπή προσημασμένων δεκαδικών αριθμών σε δυαδικούς αριθμούς προσήμου-μεγέθους
- Οι δεκαδικοί ακέραιοι αριθμοί ± 5 σε αναπαράσταση δυαδικού αριθμού προσήμου-μεγέθους με 4 bit είναι:
 - Αρχικά βρίσκουμε το μέγεθος τους αριθμού μετατρέποντας τον δεκαδικό αριθμό σε μη προσημασμένο δυαδικό αριθμό
 - $5_{10} = 101_2$
 - Στη συνέχεια βάζουμε και το bit προσήμου

$$+5 = 0101$$

$$-5 = 1101$$

Αριθμοί προσήμου-μεγέθους

- **Προβλήματα** στην αναπαράσταση προσήμου-μεγέθους
 - Η **πρόσθεση** δεκαδικών αριθμών στην αναπαράσταση προσήμου-μεγέθους **δεν δίνει σωστά αποτελέσματα**
 - Για παράδειγμα, $-5 + 5 = 0$:

Αριθμοί προσήμου-μεγέθους

■ Προβλήματα στην αναπαράσταση προσήμου-μεγέθους

- Η *πρόσθεση* δεκαδικών αριθμών στην αναπαράσταση προσήμου-μεγέθους *δεν δίνει σωστά αποτελέσματα*
- Για παράδειγμα, $-5 + 5 = 0$:

$$\begin{array}{r} 1101 \\ + \quad 0101 \\ \hline \end{array}$$

$\textcolor{red}{1}0010$ (2_{10} αντί για 0 – Λάθος!)

■ Επίσης, υπάρχουν δύο αναπαραστάσεις του 0

- 0000 (+0)
- 1000 (-0)

Αριθμοί συμπληρώματος ως προς 2

- **Δεν έχουν τα προβλήματα των αριθμών προσήμου-μεγέθους**
 - Η **πρόσθεση** δίνει **σωστά αποτελέσματα**
είτε οι αριθμοί είναι μη προσημασμένοι (*unsigned*)
είτε είναι προσημασμένοι (*signed*) σε αναπαράσταση
συμπληρώματος ως προς δύο
 - Υπάρχει **μόνο μία αναπαράσταση για το 0**
 - Το **κρατούμενο εξόδου** που παράγεται κατά την πρόσθεση
αγνοείται χωρίς να απαιτείται άλλη ενέργεια
 - Δεν αλλάζει το πλήθος των ψηφίων του αριθμού μετά την πρόσθεση, αλλά θέλει προσοχή στην **υπερχείλιση**
** Η υπερχείλιση παρότι υπάρχει ως έννοια και στους μη προσημασμένους αλλά και στους προσημασμένους, αντιμετωπίζεται (εντοπίζεται) διαφορετικά.*

Αριθμοί συμπληρώματος ως προς δύο

- Είναι ίδιοι με τους μη προσημασμένους δυαδικούς αριθμούς των N bit με τη μόνη διαφορά ότι η θέση του περισσότερου σημαντικού bit (MSB) έχει βάρος ίσο με -2^{N-1} αντί για 2^{N-1}
- Το **πιο σημαντικό bit** συνεχίζει να δηλώνει το **πρόσημο** (**0 = θετικός, 1 = αρνητικός**)
- Ο μεγαλύτερος θετικός αριθμός στα 4 bit είναι ο αριθμός **+7: 0111**
- Το **0** αναπαρίσταται πάντα με τα ψηφία **όλα-0** (0000)
- Το **-1** αναπαρίσταται πάντα με τα ψηφία **όλα-1** (1111)
- Ο μικρότερος αρνητικός αριθμός στα 4 bit είναι ο αριθμός **-8: 1000**
- Ένας αριθμός συμπληρώματος ως προς δύο των N bit παίρνει τιμές από το κλειστό διάστημα: $[-2^{N-1}, 2^{N-1} - 1]$

Δυαδικές κωδικοποιήσεις με 4 bit

- Προσημασμένοι αριθμοί συμπληρώματος ως προς δύο
- Εύρος τιμών στα N bit: **[-2³, 2³-1]**

Δεκαδικός	Προσημα σμένος συμπλ. ως προς δύο
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Δεκαδικός	Προσημα σμένος συμπλ. ως προς δύο
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111

Αριθμοί συμπληρώματος ως προς δύο

- **Μετατροπή** προσημασμένου αριθμού συμπληρώματος ως προς δύο σε δεκαδικό προσημασμένο ακέραιο αριθμό:
 - Η διαδικασία είναι ίδια με τους μη προσημασμένους δυαδικούς αριθμούς με τη μόνη διαφορά ότι το πιο σημαντικό bit (το bit προσήμου) και έχει βάρος -2^{N-1} αντί για 2^{N-1}
 - $1011 = 1 \times -2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = -5$
- **Υπολογισμός του συμπληρώματος ως προς δύο** ενός αριθμού με αναπαράσταση συμπληρώματος ως προς δύο (δηλαδή από τον αριθμό X οδηγούμαστε στον $-X$)
 - Αντιστροφή όλων των bit του δυαδικού αριθμού
 - Πρόσθεση του '1'
- Για παράδειγμα ο αριθμός $3_{10} = 0011_2$ έχει συμπλήρωμα ως προς δύο το -3_{10} που βρίσκεται ως εξής:
 1. Αντιστροφή bit: 1100
 2. Πρόσθεση +1: +0001

$$1101 = -3_{10}$$

Παράδειγμα 1.10

- Αναπαράσταση ενός αρνητικού αριθμού με χρήση συμπληρώματος ως προς δύο
- Βρείτε πώς μπορεί να αναπαρασταθεί το -2_{10} ως αριθμός συμπληρώματος ως προς δύο με 4 bit.

Παράδειγμα 1.10

- Αναπαράσταση ενός αρνητικού αριθμού με χρήση συμπληρώματος ως προς δύο
- Βρείτε πώς μπορεί να αναπαρασταθεί το -2_{10} ως αριθμός συμπληρώματος ως προς δύο με 4 bit.
 - Αρχικά βρίσκουμε τον δυαδικό αριθμό με το ίδιο μέγεθος
 $+2_{10} = 0010_2$

Παράδειγμα 1.10

- Αναπαράσταση ενός αρνητικού αριθμού με χρήση συμπληρώματος ως προς δύο
- Βρείτε πώς μπορεί να αναπαρασταθεί το -2_{10} ως αριθμός συμπληρώματος ως προς δύο με 4 bit.
 - Αρχικά βρίσκουμε τον δυαδικό αριθμό με το ίδιο μέγεθος
 $+2_{10} = 0010_2$
 - Στη συνέχεια αντιστρέφουμε τα bit του 0010_2 οπότε παράγεται το 1101_2

Παράδειγμα 1.10

- Αναπαράσταση ενός αρνητικού αριθμού με χρήση συμπληρώματος ως προς δύο
- Βρείτε πώς μπορεί να αναπαρασταθεί το -2_{10} ως αριθμός συμπληρώματος ως προς δύο με 4 bit.
 - Αρχικά βρίσκουμε τον δυαδικό αριθμό με το ίδιο μέγεθος
 $+2_{10} = 0010_2$
 - Στη συνέχεια αντιστρέφουμε τα bit του 0010_2 οπότε παράγεται το 1101_2
 - Τέλος, προσθέτουμε το 1 στο προηγούμενο αποτέλεσμα
 $1101_2 + 0001_2 = 1110_2 = -2_{10}$

Παράδειγμα 1.11

- Εύρεση της τιμής των αρνητικών αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Βρείτε τη δεκαδική τιμή του αριθμού 1001_2 , ο οποίος αναπαρίσταται με τη χρήση συμπληρώματος ως προς δύο.
- Μέθοδος 1: Εύρεση συμπληρώματος ως προς 2
 - Αντιστρέφουμε τα *bit* και προσθέτουμε 1
 - $1001_2 = 0110_2$
 - $0110_2 + 0001_2 = 0111_2 = 7_{10}$
 - *Άρα, $1001_2 = -7_{10}$*
- Μέθοδος 2: Μετατροπή στον αντίστοιχο δεκαδικό αριθμό με βάση ότι το βάρος του MSB είναι -2^3 αντί για 2^3
 - $1001 = 1 \times -2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = -7$
(Προσοχή ότι μόνο το MSB έχει αρνητικό βάρος)

Συμπλήρωμα ως προς δύο του 0

- Βρείτε πώς μπορεί να αναπαρασταθεί το -0_{10} ως αριθμός συμπληρώματος ως προς δύο με 4 bit.
 - Αρχικά βρίσκουμε τον δυαδικό αριθμό με το ίδιο μέγεθος
 $+0_{10} = 0000_2$
 - Στη συνέχεια αντιστρέφουμε τα bit του 0000_2 οπότε παράγεται το 1111_2
 - Τέλος, προσθέτουμε το 1 στο προηγούμενο αποτέλεσμα
 $1111_2 + 0001_2 = \underline{1}0000_2 = -0_{10}$
- Δεν υπάρχει ξεχωριστή αναπαράσταση για το -0_{10} με αναπαράσταση συμπληρώματος ως προς δύο
- Το μηδέν θεωρείται θετικό επειδή στο bit του προσήμου του υπάρχει το ψηφίο 0
- Συνεπώς οι θετικοί αριθμοί που μπορούν να αναπαρασταθούν με συμπλήρωμα ως προς δύο είναι κατά ένας λιγότεροι σε σχέση με τους αντίστοιχους αρνητικούς αριθμούς

Συμπλήρωμα ως προς δύο του -2^{N-1}

- Βρείτε εάν μπορεί να αναπαρασταθεί το $+2^{N-1}_{10}$ ως αριθμός συμπληρώματος ως προς δύο με 4 bit ($N=4$, ισχύει για κάθε N).

Συμπλήρωμα ως προς δύο του -2^{N-1}

- Βρείτε εάν μπορεί να αναπαρασταθεί το $+2^{N-1}_{10}$ ως αριθμός συμπληρώματος ως προς δύο με 4 bit (N=4, ισχύει για κάθε N).
 - Αρχικά βρίσκουμε τον αριθμό συμπληρώματος ως προς δύο του $-2^3_{10} = 1000_2$
 - Στη συνέχεια αντιστρέφουμε τα bit του 1000_2 οπότε παράγεται το 0111_2
 - Τέλος, προσθέτουμε το 1 στο προηγούμενο αποτέλεσμα
 $0111_2 + 0001_2 = 1000_2 = -2^3_{10}$
- Δεν υπάρχει δυνατότητα αναπαράστασης για το $+2^3_{10}$ λόγω του φαινομένου της υπερχείλισης των προσημασμένων αριθμών συμπληρώματος ως προ δύο
 - Προσθέσαμε δύο θετικούς αριθμούς και το αποτέλεσμα ήταν ένας αρνητικός αριθμός

Πρόσθεση προσημασμένων αριθμών

- Η πρόσθεση προσημασμένων αριθμών με αναπαράσταση συμπληρώματος ως προς δύο υλοποιείται ακριβώς όπως και στους μη προσημασμένους αριθμούς με τις ακόλουθες διαφοροποιήσεις ως προς την ερμηνεία του αποτελέσματος:
 - Σε αντίθεση με τους μη προσημασμένους αριθμούς η παραγωγη ενός κρατούμενου εξόδου από τη στήλη του πιο σημαντικού bit **δεν σημαίνει πάντοτε υπερχείλιση**
 - Η πρόσθεση δύο **ετερόσημων αριθμών** δεν παράγει ποτέ υπερχείλιση
 - Η πρόσθεση δύο **ομόσημων αριθμών** των N bits μπορεί να προκαλέσει υπερχείλιση αν το αποτέλεσμα είναι μεγαλύτερο του $2^{N-1} - 1$ ή μικρότερο του -2^{N-1}
- Αν δύο ομόσημοι αριθμοί προστεθούν και το αποτέλεσμα έχει αντίθετο πρόσημο, τότε παρουσιάζεται **υπερχείλιση**
- **Προσοχή:** Δεν ταυτίζονται οι υπερχειλίσεις που παράγονται κατά την πρόσθεση μεταξύ προσημασμένων και μη προσημασμένων αριθμών

Παράδειγμα 1.12

- Πρόσθεση αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Υπολογίστε τα αθροίσματα $(\alpha) -2_{10} + 1_{10}$ και $(\beta) -7_{10} + 7_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο, των 4 bit.

Παράδειγμα 1.12

- Πρόσθεση αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Υπολογίστε τα αθροίσματα (α) $-2_{10} + 1_{10}$ και (β) $-7_{10} + 7_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο.
 - $\alpha) -2_{10} + 1_{10} = 1110_2 + 0001_2 = 1111_2 = -1_{10}$
 - $\beta) -7_{10} + 7_{10} = 1001_2 + 0111_2 = 10000_2 = 0000_2 = 0_{10}$

Το κρατούμενο εξόδου (πέμπτο bit) αγνοείται, οπότε μένει το σωστό αποτέλεσμα (με 4 bit), δηλαδή το 0000_2 .

Παράδειγμα 1.14

- Πρόσθεση αριθμών συμπληρώματος ως προς δύο με υπερχείλιση
- Υπολογίστε το άθροισμα $4_{10} + 5_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο με 4 bit

Παράδειγμα 1.14

- Πρόσθεση αριθμών συμπληρώματος ως προς δύο με υπερχείλιση
- Υπολογίστε το άθροισμα $4_{10} + 5_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο με 4 bit
 - $4_{10} + 5_{10} = 0100_2 + 0101_2 = 1001_2 = -7_{10}$
 - Το αποτέλεσμα υπερχειλίζει το εύρος τιμών των θετικών αριθμών συμπληρώματος ως προς δύο με 4 bit (που είναι το $+7_{10}$,) παράγοντας ένα εσφαλμένο αρνητικό αποτέλεσμα, το -7_{10}
 - Εάν χρησιμοποιούσαμε αριθμούς τουλάχιστον των 5 bits τότε το αποτέλεσμα θα ήταν σωστό
 - $4_{10} + 5_{10} = 00100_2 + 00101_2 = 01001_2 = 9_{10}$
 - **Προσοχή:** Εάν θεωρήσουμε ότι οι αριθμοί των 4 bit είναι μη προσημασμένοι, το αποτέλεσμα είναι σωστό χωρίς υπερχείλιση!
 - $4_{10} + 5_{10} = 0100_2 + 0101_2 = 1001_2 = 9_{10}$

Το ατύχημα του πύραυλου Ariane 5

- Ο πύραυλος Ariane 5, που κόστισε 7 δισεκατομμύρια δολάρια και εκτοξεύτηκε στις 4 Ιουνίου 1996, απέκλινε από την πορεία του 40 δευτερόλεπτα μετά από την εκτόξευση, κόπηκε στα δύο και εξερράγη.
- Η αποτυχία προκλήθηκε όταν ο υπολογιστής που έλεγχε τον πύραυλο **υπερχείλισε** το εύρος (16 bit) των προσημασμένων τιμών του και κατέρρευσε.
- Ο εν λόγω κώδικας είχε ελεγχθεί διεξοδικά στον πύραυλο Ariane 4. Όμως, ο Ariane 5 διέθετε πιο γρήγορη μηχανή η οποία παρήγαγε μεγαλύτερες τιμές για τον υπολογιστή ελέγχου, προκαλώντας έτσι την υπερχείλιση



Αφαίρεση αριθμών συμπληρώματος ως προς δύο

■ **Μέθοδος 1:**

Για να πραγματοποιήσουμε την αφαίρεση υπολογίζουμε το συμπλήρωμα ως προς δύο του αφαιρετέου και το προσθέτουμε στον μειωτέο

■ **Μέθοδος 2: Εναλλακτική μέθοδος**

Για να πραγματοποιήσουμε την αφαίρεση αντιστρέφουμε τα bit του αφαιρετέου και το προσθέτουμε στο μειωτέο ξεκινώντας με κρατούμενο εισόδου 1, αντί για 0

Παράδειγμα 1.13

- Αφαίρεση αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Υπολογίστε τα (α) $5_{10} - 3_{10}$ και (β) $3_{10} - 5_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο με 4 bit.
- **Μέθοδος 1:**

Παράδειγμα 1.13

- Αφαίρεση αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Υπολογίστε τα (α) $5_{10} - 3_{10}$ και (β) $3_{10} - 5_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο με 4 bit.
- **Μέθοδος 1:**
 - (α) Ο μειωτέος είναι $5_{10} = 0101_2$
Ο αφαιρετέος είναι $3_{10} = 0011_2$
 - Υπολογίζουμε το συμπλήρωμα του αφαιρετέου ως προς δύο και παίρνουμε $-3_{10} = 1101_2$
 - Εκτελούμε πρόσθεση αντί για αφαίρεση
 - $5_{10} + (-3_{10}) = 0101_2 + 1101_2 = \textcolor{red}{10010}_2 = 2_{10}$
 - Το κρατούμενο στην πιο σημαντική θέση αγνοείται

Παράδειγμα 1.13

- Αφαίρεση αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Υπολογίστε τα (α) $5_{10} - 3_{10}$ και (β) $3_{10} - 5_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο με 4 bit.
- **Μέθοδος 1:**
 - (β) Ο μειωτέος είναι $3_{10} = 0011_2$
Ο αφαιρετέος είναι $5_{10} = 0101_2$
 - Υπολογίζουμε το συμπλήρωμα του αφαιρετέου ως προς δύο και παίρνουμε $-5_{10} = 1011_2$
 - Εκτελούμε πρόσθεση αντί για αφαίρεση
 - $3_{10} + (-5_{10}) = 0011_2 + 1011_2 = 1110_2 = -2_{10}$

Παράδειγμα 1.13

- Αφαίρεση αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Υπολογίστε τα (α) $5_{10} - 3_{10}$ και (β) $3_{10} - 5_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο με 4 bit.
- **Μέθοδος 2:**
 - (α) Ο μειωτέος είναι $5_{10} = 0101_2$
Ο αφαιρετέος είναι $3_{10} = 0011_2$
 - Αντιστρέφουμε τα bit του αφαιρετέου και παίρνουμε 1100_2
 - Εκτελούμε πρόσθεση αντί για αφαίρεση με κρατούμενο εισόδου 1
 - $5_{10} + (-3_{10}) = 0101_2 + 1100_2 + 1 = 0101_2 + 1100_2 + 0001 = 2_{10}$
 - Το κρατούμενο στην πιο σημαντική θέση αγνοείται

Παράδειγμα 1.13

- Αφαίρεση αριθμών με αναπαράσταση συμπληρώματος ως προς δύο
- Υπολογίστε τα (α) $5_{10} - 3_{10}$ και (β) $3_{10} - 5_{10}$ χρησιμοποιώντας αριθμούς συμπληρώματος ως προς δύο με 4 bit.
- **Μέθοδος 2:**

(β) Ο μειωτέος είναι $3_{10} = 0011_2$

Ο αφαιρετέος είναι $5_{10} = 0101_2$

- Αντιστρέφουμε τα bit του αφαιρετέου και παίρνουμε 1010_2
- Εκτελούμε πρόσθεση αντί για αφαίρεση με κρατούμενο εισόδου 1
- $3_{10} + (-5_{10}) = 0011_2 + 1010_2 + 1 = 0011_2 + 1010_2 + 0001 = 1110_2 = -2_{10}$

Επέκταση προσήμου/μηδενός

- Ένας προσημασμένος αριθμός συμπληρώματος ως προς δύο μπορεί να επεκταθεί ώστε να έχει περισσότερα bits
 - Αυτή η διαδικασία λέγεται **επέκταση προσήμου** (sign extension)
Τα επιπλέον bits που θα προστεθούν θα αντιγράφονται από το bit προσήμου
 - Έστω ότι θέλουμε να προσθέσουμε 4 παραπάνω bit αριστερά:
 $3 = 0011 = 00000011$
 $-5 = 1011 = 11111011$
- Ένας μη προσημασμένος αριθμός μπορεί να επεκταθεί ώστε να έχει περισσότερα bits
 - Αυτή η διαδικασία λέγεται **επέκταση μηδενός** (zero extension)
 - Τα επιπλέον bits που θα προστεθούν θα είναι 0
 - Έστω ότι θέλουμε να προσθέσουμε 4 παραπάνω bit αριστερά

$$3 = 0011 = 00000011$$

$$11 = 1011 = 00001011$$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.58 – Λύση

Εκτελέστε τις ακόλουθες προσθέσεις μη προσημασμένων δεκαεξαδικών αριθμών. Αναφέρετε αν το άθροισμα θα προκαλέσει υπερχείλιση ή όχι στην περίπτωση που το αποτέλεσμα έχει 8 bit (δύο δεκαεξαδικά ψηφία).

$$(\gamma) \quad AB_{16} + 3E_{16}$$

$$(\delta) \quad 8F_{16} + AD_{16}$$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.58 – Λύση

Εκτελέστε τις ακόλουθες προσθέσεις μη προσημασμένων δεκαεξαδικών αριθμών. Αναφέρετε αν το άθροισμα θα προκαλέσει υπερχείλιση ή όχι στην περίπτωση που το αποτέλεσμα έχει 8 bit (δύο δεκαεξαδικά ψηφία).

$$(\gamma) \quad AB_{16} + 3E_{16}$$

$$(\delta) \quad 8F_{16} + AD_{16}$$

Κρατούμενο 11111
(γ) 10101011
+ 00111110

11101001
(171+62=233)

ΔΕΝ υπάρχει υπερχείλιση

Κρατούμενο 1111
(δ) 10001111
+ 10101101

100111100(=60)
(143+173=316)

Carry →
Υπάρχει υπερχείλιση

Επιλεγμένες ασκήσεις

■ Άσκηση 1.61 – Λύση

Μετατρέψτε τους παρακάτω δεκαδικούς αριθμούς σε δυαδικούς αριθμούς συμπληρώματος ως προς δύο των 6 bit και αφαιρέστε τους. Αναφέρετε αν η διαφορά θα προκαλέσει υπερχείλιση ή όχι στην περίπτωση που το αποτέλεσμα έχει 6 bit.

$$(\gamma) \quad -28_{10} - 3_{10}$$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.61 – Λύση

Μετατρέψτε τους παρακάτω δεκαδικούς αριθμούς σε δυαδικούς αριθμούς συμπληρώματος ως προς δύο των 6 bit και αφαιρέστε τους. Αναφέρετε αν η διαφορά θα προκαλέσει υπερχείλιση ή όχι στην περίπτωση που το αποτέλεσμα έχει 6 bit.

$$(γ) \quad -28_{10} - 3_{10}$$

$$28_{10} = 011100_2$$

$$\begin{aligned}-28_{10} &= 100011_2 + 1 \\ &= 100100_2\end{aligned}$$

$$3_{10} = 000011_2$$

$$\begin{aligned}-3_{10} &= 111100_2 + 1 \\ &= 111101_2\end{aligned}$$

$$-28_{10} - 3_{10} = (-28_{10}) + (-3_{10})$$

$$\begin{array}{r} 111 \\ 100100_2 \\ + 111101_2 \\ \hline \end{array}$$

Overflow?

OXI

$$\underline{\hspace{2cm}}$$

$$1100001$$

ΔΕΝ υπάρχει υπερχείλιση

Επιλεγμένες ασκήσεις

■ Άσκηση 1.61 – Λύση

Μετατρέψτε τους παρακάτω δεκαδικούς αριθμούς σε δυαδικούς αριθμούς συμπληρώματος ως προς δύο των 6 bit και αφαιρέστε τους. Αναφέρετε αν η διαφορά θα προκαλέσει υπερχείλιση ή όχι στην περίπτωση που το αποτέλεσμα έχει 6 bit.

(δ) $-16_{10} - 21_{10}$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.61 – Λύση

Μετατρέψτε τους παρακάτω δεκαδικούς αριθμούς σε δυαδικούς αριθμούς συμπληρώματος ως προς δύο των 6 bit και αφαιρέστε τους. Αναφέρετε αν η διαφορά θα προκαλέσει υπερχείλιση ή όχι στην περίπτωση που το αποτέλεσμα έχει 6 bit.

$$(δ) \quad -16_{10} - 21_{10}$$

$$16_{10} = 010000_2$$

$$\begin{array}{r} -16_{10} = 101111_2 + 1 \\ \hline 110000_2 \end{array}$$

$$21_{10} = 010101_2$$

$$\begin{array}{r} -21_{10} = 101010_2 + 1 \\ \hline = 101011_2 \end{array}$$

$$-16_{10} - 21_{10} = (-16_{10}) + (-21_{10})$$

$$\begin{array}{r} 110000_2 \\ + 101011_2 \\ \hline \end{array}$$

Overflow?
NAI

1011011

Υπάρχει υπερχείλιση

Πράξεις αριθμών συμπληρώματος ως προς δύο

- Υπερχείλιση μπορεί να έχουμε ΜΟΝΟ όταν προσθέτουμε ομόσημους αριθμούς.
 - An to αποτέλεσμα έχει διαφορετικό πρόσημο από τους αριθμούς που προσθέτουμε τότε έχουμε υπερχείλιση
- Παράδειγμα, υπολογίστε την ακόλουθη πράξη (στα 3 bit)
 - $-4_{10} + -4_{10}$

Πράξεις αριθμών συμπληρώματος ως προς δύο

- Υπερχείλιση μπορεί να έχουμε ΜΟΝΟ όταν προσθέτουμε ομόσημους αριθμούς.
 - Αν το αποτέλεσμα έχει διαφορετικό πρόσημο από τους αριθμούς που προσθέτουμε τότε έχουμε υπερχείλιση
 - Παράδειγμα, υπολογίστε την ακόλουθη πράξη (στα 3 bit)
 - $-4_{10} + -4_{10}$
 - Το $4_{10} = 100 \Rightarrow -4_{10} = 011 + 1 = 100$ (Μπορούμε να το υπολογίσουμε και απευθείας)
- $$\begin{array}{r} 100_2 \\ 100_2 \\ \hline \textcolor{red}{1000}_2 \end{array}$$
- Αντί για -8_{10} έχουμε 0_{10}
Υπερχείλιση (Overflow)

Δυαδικές κωδικοποιήσεις με 4 bit

- Binary Coded Decimal (BCD) κωδικοποίηση

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

Επιλεγμένες ασκήσεις

■ Άσκηση 1.64

Σε ένα δυαδικά κωδικοποιημένο δεκαδικό (**binary coded decimal, BCD**) σύστημα, χρησιμοποιούνται 4 bit για την αναπαράσταση ενός δεκαδικού ψηφίου από το 0 έως το 9.

Για παράδειγμα, το 37_{10} γράφεται ως 00110111_{BCD} .

- (α) Γράψτε τον αριθμό 289_{10} στο σύστημα BCD.
- (β) Μετατρέψτε τον αριθμό 100101010001_{BCD} σε δεκαδικό.
- (γ) Μετατρέψτε τον αριθμό 01101001_{BCD} σε δυαδικό.
- (δ) Εξηγήστε γιατί το σύστημα BCD ενδέχεται να είναι ένας χρήσιμος τρόπος αναπαράστασης αριθμών.
 - Εύκολη μετατροπή από το BCD στο δεκαδικό σύστημα
 - Ακριβής αναπαράσταση του διψήφιου δεκαδικού κλασματικού μέρους με ένα byte στις χρηματοοικονομικές εφαρμογές

Επιλεγμένες ασκήσεις

■ Άσκηση 1.64 - Λύση

Σε ένα δυαδικά κωδικοποιημένο δεκαδικό (**binary coded decimal, BCD**) σύστημα, χρησιμοποιούνται 4 bit για την αναπαράσταση ενός δεκαδικού ψηφίου από το 0 έως το 9.

Για παράδειγμα, το 37_{10} γράφεται ως 00110111_{BCD} .

(α) Γράψτε τον αριθμό 289_{10} στο σύστημα BCD.

0010 1000 1001

2 8 9

Επιλεγμένες ασκήσεις

■ Άσκηση 1.64 - Λύση

Σε ένα δυαδικά κωδικοποιημένο δεκαδικό (**binary coded decimal, BCD**) σύστημα, χρησιμοποιούνται 4 bit για την αναπαράσταση ενός δεκαδικού ψηφίου από το 0 έως το 9.

Για παράδειγμα, το 37_{10} γράφεται ως 00110111_{BCD} .

(β) Μετατρέψτε τον αριθμό 100101010001_{BCD} σε δεκαδικό.

$$1001=9_{10}, \quad 0101=5_{10}, \quad 0001=1_{10} \Rightarrow 951_{10}$$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.64 - Λύση

Σε ένα δυαδικά κωδικοποιημένο δεκαδικό (**binary coded decimal, BCD**) σύστημα, χρησιμοποιούνται 4 bit για την αναπαράσταση ενός δεκαδικού ψηφίου από το 0 έως το 9.

Για παράδειγμα, το 37_{10} γράφεται ως 00110111_{BCD} .

(γ) Μετατρέψτε τον αριθμό 01101001_{BCD} σε δυαδικό.

Αρχικά μετατρέπουμε το BCD σε δεκαδικό

$0110=6_{10}$, $1001=9_{10}$, οπότε έχουμε τον 69_{10}

Κατόπιν μετατρέπουμε το δεκαδικό σε δυαδικό με όποιο τρόπο θέλουμε

$$69_{10} = 64 + 4 + 1 = 1 \cdot 2^6 + 1 \cdot 2^2 + 1 \cdot 2^0 = 1000101_2$$

Επιλεγμένες ασκήσεις

■ Άσκηση 1.65

Σε ένα δυαδικά κωδικοποιημένο δεκαδικό (**binary coded decimal, BCD**) σύστημα, χρησιμοποιούνται 4 bit για την αναπαράσταση ενός δεκαδικού ψηφίου από το 0 έως το 9.

Για παράδειγμα, το 37_{10} γράφεται ως 00110111_{BCD} .

Εξηγήστε τα μειονεκτήματα του συστήματος BCD σε σύγκριση με τις δυαδικές αναπαραστάσεις αριθμών

Επιλεγμένες ασκήσεις

■ Άσκηση 1.65

Σε ένα δυαδικά κωδικοποιημένο δεκαδικό (**binary coded decimal, BCD**) σύστημα, χρησιμοποιούνται 4 bit για την αναπαράσταση ενός δεκαδικού ψηφίου από το 0 έως το 9.

Για παράδειγμα, το 37_{10} γράφεται ως 00110111_{BCD} .

Εξηγήστε τα μειονεκτήματα του συστήματος BCD σε σύγκριση με τις δυαδικές αναπαραστάσεις αριθμών

- Δεν λειτουργεί σωστά η πρόσθεση
- Σε 1 byte στο BCD σύστημα αποθηκεύονται 100 αριθμοί (0-99), ενώ στο δυαδικό σύστημα 256 αριθμοί

Επιλεγμένες ασκήσεις

■ Άσκηση 1.66

Ένας ιπτάμενος δίσκος πέφτει σε ένα χωράφι με καλαμπόκια στη Νεμπράσκα. Το FBI, που καλείται να ερευνήσει τα συντρίμμια, βρίσκει ένα τεχνικό εγχειρίδιο το οποίο περιέχει μια εξίσωση στο Αρειανό αριθμητικό σύστημα:

$$325 + 42 = 411$$

Αν η εξίσωση είναι σωστή, πόσα δάχτυλα σε κάθε ένα από τα δύο χέρια θα περιμένατε να έχουν οι Αρειανοί;

*Σκεφτείτε πως χρησιμοποιούμε εμείς τα δάκτυλα για πράξεις και τι σημαίνει αυτό για το αριθμητικό σύστημα που χρησιμοποιούμε.

Επιλεγμένες ασκήσεις

■ Άσκηση 1.66

Ένας ιπτάμενος δίσκος πέφτει σε ένα χωράφι με καλαμπόκια στη Νεμπράσκα. Το FBI, που καλείται να ερευνήσει τα συντρίμμια, βρίσκει ένα τεχνικό εγχειρίδιο το οποίο περιέχει μια εξίσωση στο Αρειανό αριθμητικό σύστημα:

$$325 + 42 = 411$$

Αν η εξίσωση είναι σωστή, πόσα δάκτυλα σε κάθε ένα από τα δύο χέρια θα περιμένατε να έχουν οι Αρειανοί;

*Σκεφτείτε πως χρησιμοποιούμε εμείς τα δάκτυλα για πράξεις και τι σημαίνει αυτό για το αριθμητικό σύστημα που χρησιμοποιούμε.

6αδικό σύστημα :3 δάκτυλα σε κάθε χέρι, ψηφία 0-5

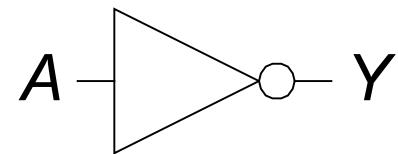
Λογικές Πύλες

- Οι λογικές πύλες είναι απλά ψηφιακά κυκλώματα που δέχονται μία η παραπάνω δυαδικές εισόδους και παράγουν μία δυαδική έξοδο
- Η σχέση μεταξύ εισόδων και εξόδου μπορεί να περιγραφτεί από έναν πίνακα αλήθειας ή μία εξίσωση Boole
- Ο πίνακας αλήθειας έχει δύο μέρη, αριστερά οι είσοδοι και δεξιά οι αντίστοιχοι έξοδοι
- Ο πίνακας αληθείας έχει 2^N γραμμές ώστε να περιέχει όλους τους δυνατούς συνδυασμούς των τιμών των N εισόδων
- Η εξίσωση Boole είναι μια μαθηματική παράσταση με δυαδικές μεταβλητές και λογικούς τελεστές που προκύπτει από τον πίνακα αλήθειας

Πύλη NOT

- Η πύλη NOT παίρνει μία τιμή και παράγει την αντίστροφη τιμή στην έξοδο
 - Για είσοδο '1' (TRUE) παράγει έξοδο '0' (FALSE)
 - Για είσοδο '0' (FALSE) παράγει έξοδο '1' (TRUE)
- Αν η είσοδος είναι A και η έξοδος Y τότε το σχηματικό της, η εξίσωση Boole και ο πίνακας αλήθειας φαίνονται δεξιά
- Ο κύκλος στο τέλος της πύλης ονομάζεται **ψυσαλίδα** και δηλώνει αντιστροφή της εξόδου
- Η γραμμή πάνω από το A στην εξίσωση Boole προφέρεται **NOT**
- Η πύλη NOT είναι επίσης γνωστή ως **αντιστροφέας (inverter)**

NOT



$$Y = \overline{A}$$

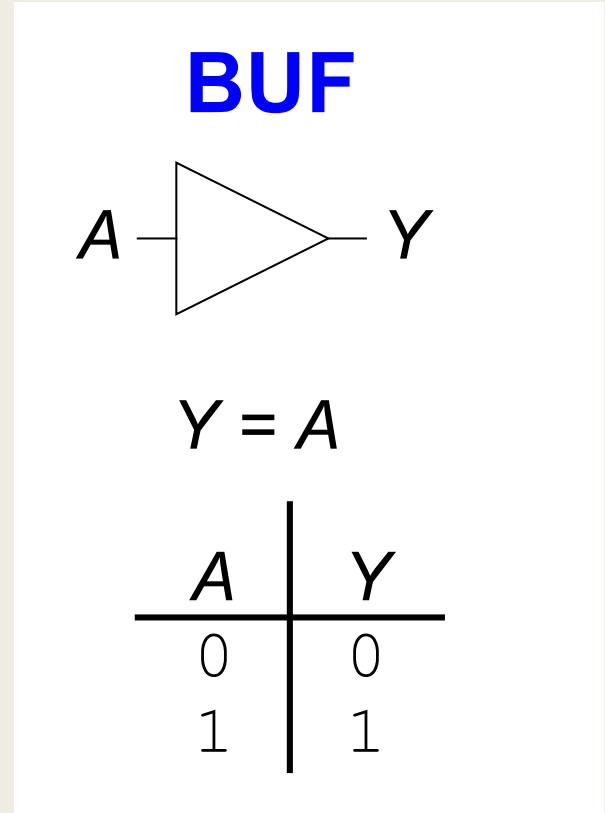
A	Y
0	1
1	0

Συμβολισμοί Πύλης NOT

- $Y = \sim A$ (SystemVerilog)
- $Y = !A$
- $Y = \text{not } A$ (VHDL)
- $Y = A'$
- $Y = \neg A$
- $Y = \bar{A}$ (βιβλίο)

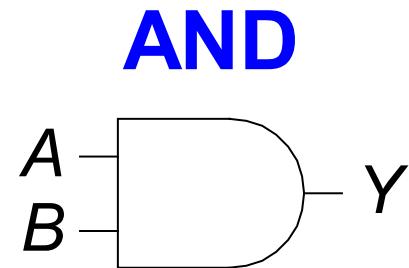
Απομονωτής (Buffer)

- Ο απομονωτής ή buffer αντιγράφει την είσοδο στην έξοδο
 - Για είσοδο '1' (TRUE) παράγει έξοδο '1' (TRUE)
 - Για είσοδο '0' (FALSE) παράγει έξοδο '0' (FALSE)
- Αν η είσοδος είναι A και η έξοδος Y τότε το σχηματικό της, η εξίσωση Boole και ο πίνακας αλήθειας φαίνονται δεξιά
- Στο σχηματικό διαφέρει από την πύλη NOT ως προς τη φυσαλίδα
- Ο απομονωτής δεν αλλάζει τη λογική τιμή, αλλά ενισχύει το ρεύμα του σήματος, ώστε να διοχετεύει μεγάλες ποσότητες ρεύματος σε έναν κινητήρα ή να διασυνδέει την έξοδο του με εισόδους πολλών πυλών



Πύλη AND (2 εισόδων)

- Η πύλη AND παράγει έξοδο Y με τιμή ‘1’ (TRUE), αν και μόνο αν, τόσο το A όσο και το B έχουν την τιμή ‘1’ (TRUE). Διαφορετικά, η έξοδος έχει την τιμή ‘0’ (FALSE)
- Η εξίσωση Boole διαβάζεται:
 - το Y ισούται με το A και το B

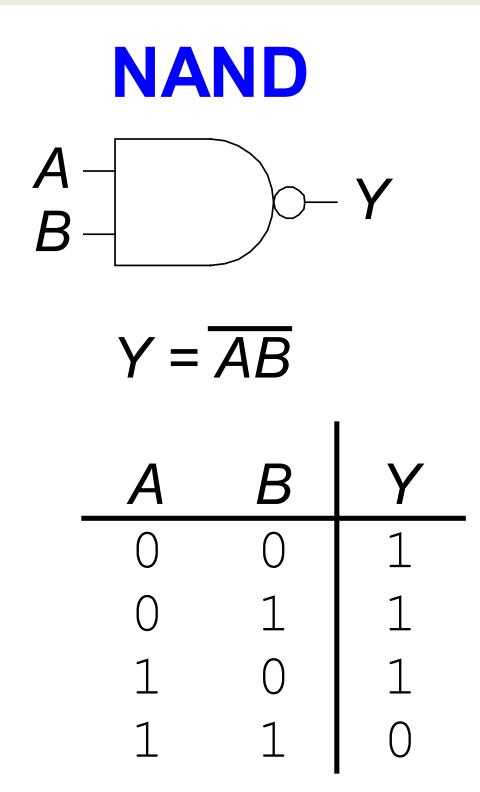


$$Y = AB$$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

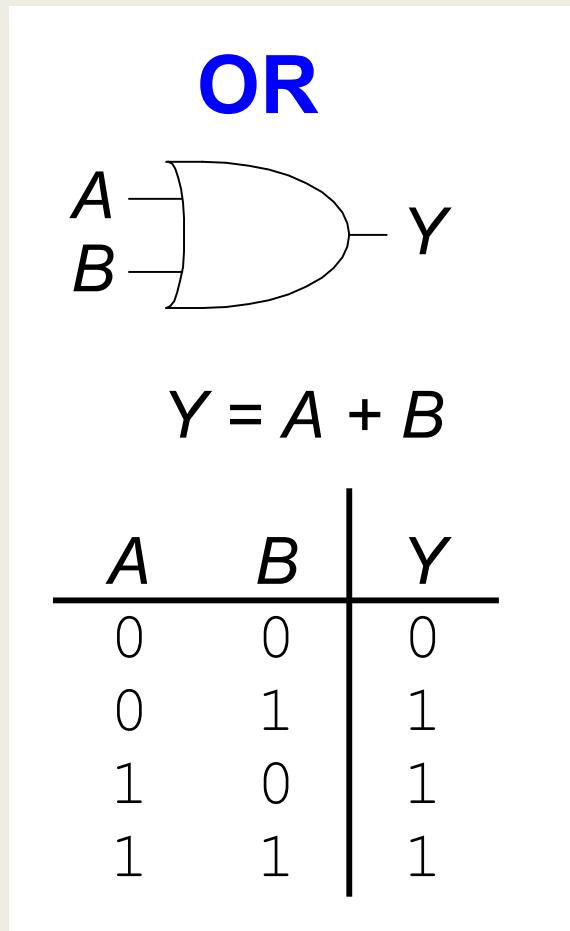
Πύλη NAND (2 εισόδων)

- Η πύλη NAND εκτελεί την πράξη NOT AND
 - Στο σχηματικό διαφέρει από την πύλη AND ως προς τη φυσαλίδα που έχει στην έξοδο
- Η πύλη NAND παράγει έξοδο πάντα με τιμή ‘1’ (TRUE), με εξαίρεση την περίπτωση που και οι δύο είσοδοι έχουν την τιμή TRUE



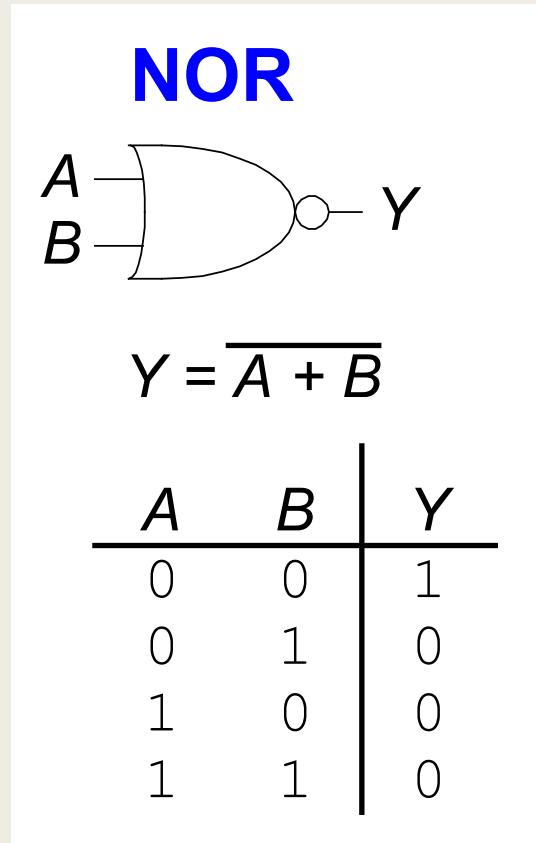
Πύλη OR (2 εισόδων)

- Η πύλη OR παράγει έξοδο Y με τιμή ‘1’ (TRUE), αν είτε το A είτε το B (είτε και τα δύο) έχουν την τιμή ‘1’ (TRUE). Διαφορετικά, η έξοδος έχει την τιμή ‘0’ (FALSE)
- Η εξίσωση Boole διαβάζεται:
 - το Y ισούται με το A ή το B



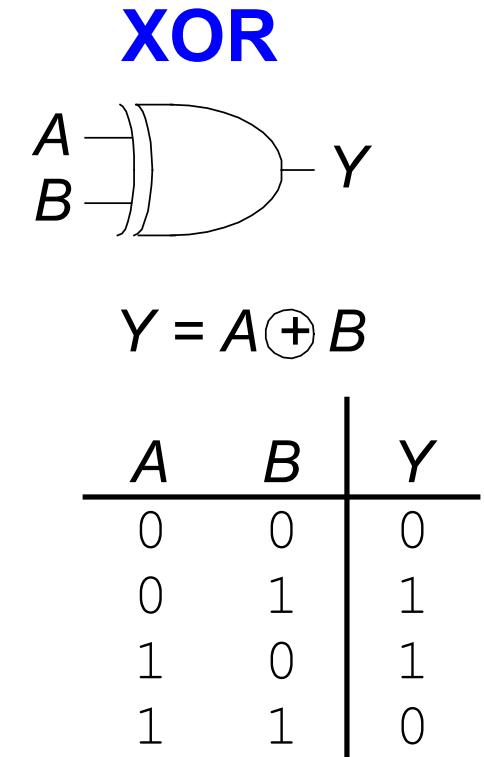
Πύλη NOR (2 εισόδων)

- Η πύλη NOR εκτελεί την πράξη NOT OR
 - Στο σχηματικό διαφέρει από την πύλη OR ως προς τη φυσαλίδα που έχει στην έξοδο
- Η πύλη NOR παράγει έξοδο με τιμή ‘1’ (TRUE) αν ούτε το A ούτε το B δεν έχουν την τιμή TRUE, δηλαδή αν και τα δύο έχουν την τιμή ‘0’ (FALSE)
- Η NOR παράγει έξοδο ‘1’ (TRUE) όταν και οι 2 είσοδοι είναι ‘0’ και η AND παράγει έξοδο ‘1’ (TRUE) όταν και οι 2 είσοδοι είναι ‘1’



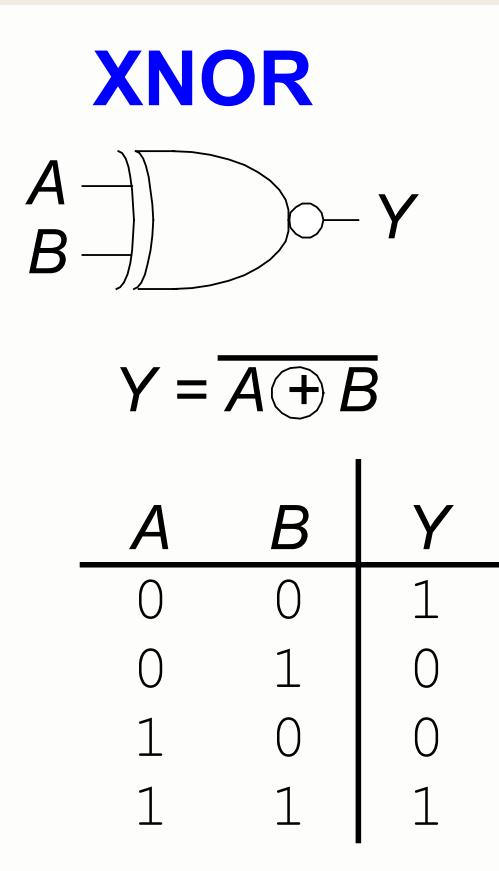
Πύλη XOR (2 εισόδων)

- Η **πύλη XOR (exclusive OR)** παράγει έξοδο με τιμή '1' (TRUE) αν το A ή το B, αλλά όχι και τα δύο ταυτόχρονα, έχουν την τιμή '1' (TRUE)
- Η πράξη XOR συμβολίζεται με το \oplus δηλαδή το συν μέσα σε έναν κύκλο
- Επίσης, υλοποιεί την αλγεβρική πράξη
$$Y = (A+B)\text{mod}2, A,B \in \{0,1\}$$
- Μερικές φορές θα την διαβάσετε και ως **EOR**
- Έχει τιμή 1 όταν οι είσοδοι είναι διαφορετικές



Πύλη XNOR (2 εισόδων)*

- Η πύλη XNOR (exclusive NOR) παράγει έξοδο με τιμή ‘1’ (TRUE) αν το A και το B ταυτόχρονα έχουν την τιμή ‘1’ (TRUE) ή την τιμή ‘0’ (FALSE)
- Η πύλη XNOR εκτελεί την αντίστροφη πράξη μίας πύλης XOR
 - Στο σχηματικό διαφέρει από την πύλη XOR ως προς τη φυσαλίδα που έχει στην έξοδο
- Μία πύλη XNOR με δύο εισόδους ονομάζεται ενίστε πύλη ισότητας (equality gate) επειδή η έξοδος της έχει την τιμή ‘1’ (TRUE) όταν οι είσοδοι είναι ίσες

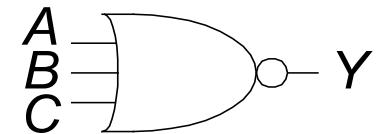


*Παράδειγμα 1.15

Παράδειγμα 1.16

- Πύλες NOR με περισσότερες από δύο εισόδους
- Μία πύλη NOR με N εισόδους παράγει έξοδο με τιμή ‘1’ (TRUE) μόνο αν όλες οι είσοδοι έχουν την τιμή ‘0’ (FALSE)

NOR3



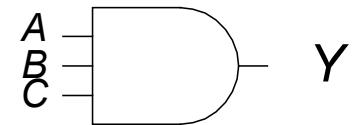
$$Y = \overline{A+B+C}$$

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Παράδειγμα 1.17

- Πύλες AND με περισσότερες από δύο εισόδους
- Μία πύλη AND με N εισόδους παράγει έξοδο με τιμή ‘1’ (TRUE) μόνο αν όλες οι είσοδοι έχουν την τιμή ‘1’ (TRUE)

AND3



$$Y = ABC$$

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Πύλες περιπτής και άρτιας ισοτιμίας

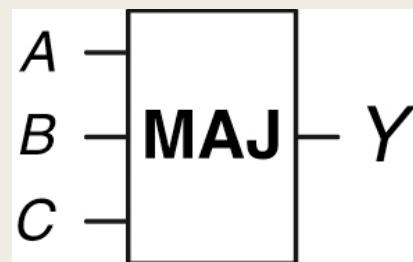
- Μια πύλη XOR με N εισόδους ονομάζεται ενίστε **πύλη περιπτής ισοτιμίας (odd parity gate)** και παράγει έξοδο με τιμή '1' (TRUE) αν ένα **περιπτό πλήθος** εισόδων έχουν την τιμή '1' (TRUE)
- Μια πύλη XNOR με N εισόδους ονομάζεται ενίστε **πύλη άρτιας ισοτιμίας (even parity gate)** και παράγει έξοδο με τιμή '1' (TRUE) αν ένα **άρτιο πλήθος εισόδων** έχουν την τιμή '1' (TRUE)
- Υλοποιούνται με δένδρα από N-1 πύλες XOR δύο εισόδων
 - Στην περίπτωση της πύλης XNOR, η πύλη δύο εισόδων στην έξοδο είναι πύλη XNOR αντί για XOR
- Χρησιμοποιούνται στην κωδικοποίηση και την κρυπτογραφία

Επιλεγμένες ασκήσεις

■ Ασκηση 1.73

Μια πύλη πλειοψηφίας (**majority gate**) παράγει έξοδο με τιμή '1' (TRUE), αν και μόνο αν, περισσότερες από τις μισές είσοδοι έχουν την τιμή '1' (TRUE)

- Συμπληρώστε τον πίνακα αληθείας για την πύλη πλειοψηφίας με τρεις εισόδους

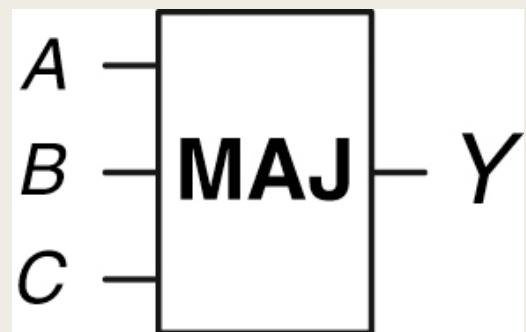


■ Άσκηση 1.73

Μια **πύλη πλειοψηφίας (majority gate)** παράγει έξοδο με τιμή '1' (TRUE), αν και μόνο αν, περισσότερες από τις μισές είσοδοι έχουν την τιμή '1' (TRUE)

- Συμπληρώστε τον πίνακα αληθείας για την πύλη πλειοψηφίας με τρεις εισόδους

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

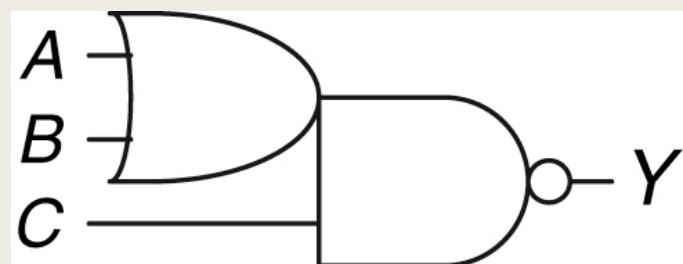


Επιλεγμένες ασκήσεις

■ Ασκηση 1.75

Μια πύλη **OR-AND-INVERT(OAI)** παράγει έξοδο με τιμή '0' (FALSE) αν το C έχει την τιμή '1' (TRUE) και το A ή το B έχει την τιμή '1' (TRUE). Διαφορετικά παράγει έξοδο με τιμή '1' (TRUE).

- Συμπληρώστε τον πίνακα αληθείας για την πύλη OR-AND-INVERT (OAI)

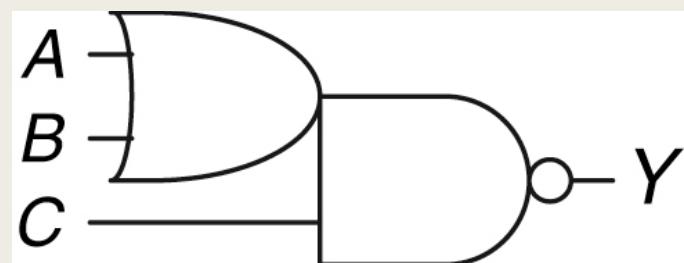


■ Άσκηση 1.75

Μια πύλη **OR-AND-INVERT(OAI)** παράγει έξοδο με τιμή '0' (FALSE) αν το C έχει την τιμή '1' (TRUE) και το A ή το B έχει την τιμή '1' (TRUE). Διαφορετικά παράγει έξοδο με τιμή '1' (TRUE).

- Συμπληρώστε τον πίνακα αληθείας για την πύλη OR-AND-INVERT (OAI)

A	B	C	Y
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	1
1	1	1	0



Επιλεγμένες ασκήσεις

■ Άσκηση 1.76

Υπάρχουν 16 διαφορετικοί πίνακες αληθείας για εξισώσεις (συναρτήσεις) Boole με δύο μεταβλητές.

- Παραθέστε όλους τους πίνακες.
- Δώστε ένα σύντομο περιγραφικό όνομα.

■ Άσκηση 1.77

Πόσοι διαφορετικοί πίνακες αληθείας υπάρχουν για συναρτήσεις Boole με N μεταβλητές;

Επιλεγμένες ασκήσεις

■ Άσκηση 1.76

Υπάρχουν 16 διαφορετικοί πίνακες αληθείας για εξισώσεις (συναρτήσεις) Boole με δύο μεταβλητές.

- Παραθέστε όλους τους πίνακες.
- Δώστε ένα σύντομο περιγραφικό όνομα.

■ Άσκηση 1.77

Πόσοι διαφορετικοί πίνακες αληθείας υπάρχουν για συναρτήσεις Boole με N μεταβλητές;

- 2^{2^N}

A	B	Y	A	B	Y	A	B	Y	A	B	Y
0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	0	1	0	0	1	1	0	1	1
1	0	0	1	0	0	1	0	0	1	0	0
1	1	0	1	1	0	1	1	0	1	1	0
Zero		A NOR B		$\bar{A}B$		NOT A					
A	B	Y	A	B	Y	A	B	Y	A	B	Y
0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	0	1	0	0	1	1	0	1	1
1	0	1	1	0	1	1	0	1	1	0	1
1	1	0	1	1	0	1	1	0	1	1	0
$A\bar{B}$		NOT B		XOR		NAND					
A	B	Y	A	B	Y	A	B	Y	A	B	Y
0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	0	1	0	0	1	1	0	1	1
1	0	0	1	0	0	1	0	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1
AND		XNOR		\bar{B}		$\bar{A} + B$					
A	B	Y	A	B	Y	A	B	Y	A	B	Y
0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	0	1	0	0	1	1	0	1	1
1	0	0	1	0	0	1	0	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1
A		$A + \bar{B}$		OR		One					

Αριθμητικά Κυκλώματα

- Τα **αριθμητικά κυκλώματα** αποτελούν τα κεντρικά δομικά στοιχεία των υπολογιστών
- Οι υπολογιστές και τα στοιχεία ψηφιακής λογικής εκτελούν πολλές αριθμητικές λειτουργίες ή πράξεις, όπως:
 - *πρόσθεση*
 - *αφαίρεση*
 - *συγκρίσεις*
 - *ολισθήσεις*
 - *πολλαπλασιασμό*
 - *διαίρεση*

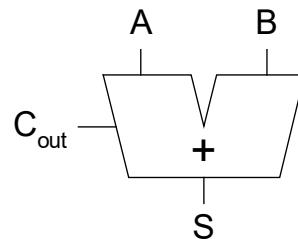
Στην ενότητα αυτή θα επικεντρωθούμε στα αριθμητικά κυκλώματα που εκτελούν τις πράξεις της **πρόσθεσης** και της **αφαίρεσης**

Αθροιστές του ενός bit

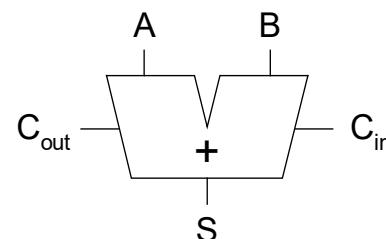
- Προσθέτουν τα ψηφία της ίδιας στήλης (ή αλλιώς ίδιου βάρους) μαζί με το bit κρατούμενου (εάν υπάρχει)
- Υπάρχουν δύο αθροιστές του ενός bit
 - Ο **ημιαθροιστής (half-adder)**, όταν **δεν** υπάρχει κρατούμενο εισόδου
 - Ο **πλήρης αθροιστής (full-adder)**, όταν υπάρχει **κρατούμενο εισόδου C_{in} (carry in)**
- Παράγουν στην έξοδο
 - Το **άθροισμα S (sum)**
 - Το **κρατούμενο εξόδου C_{out} (carry out)**
- Σε έναν **αθροιστή πολλών bit**, το κρατούμενο εξόδου C_{out} μεταφέρεται ως κρατούμενο εισόδου C_{in} , στην επόμενη στήλη στα αριστερά (στο επόμενο βάρος), όταν υπάρχει
- Δεν διαφοροποιείται η πράξη της πρόσθεσης στο δυαδικό σύστημα μεταξύ μη προσημασμένων αριθμών και προσημασμένων αριθμών σε αναπαράσταση συμπληρώματος ως προς δύο
 - αλλάζει μόνο η ερμηνεία των αριθμών και η εμφάνιση της υπερχείλισης

Αθροιστές του ενός bit

Half
Adder



Full
Adder



A	B	C _{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$

$$C_{out} = AB$$

C _{in}	A	B	C _{out}	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

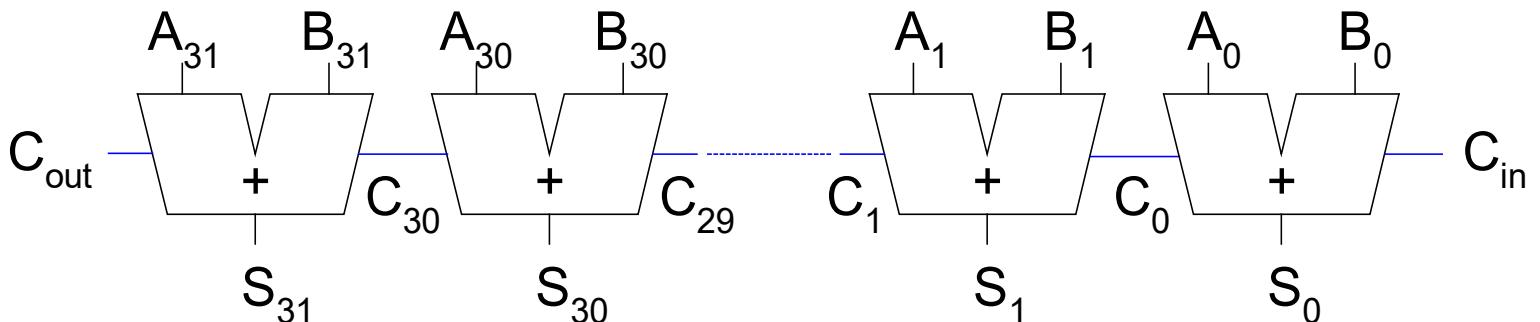
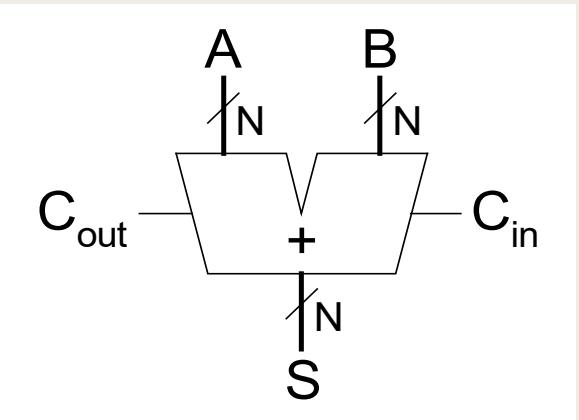
$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

Αθροιστές των N bit

■ Αθροιστής κυμάτωσης κρατούμενου (ripple-carry adder)

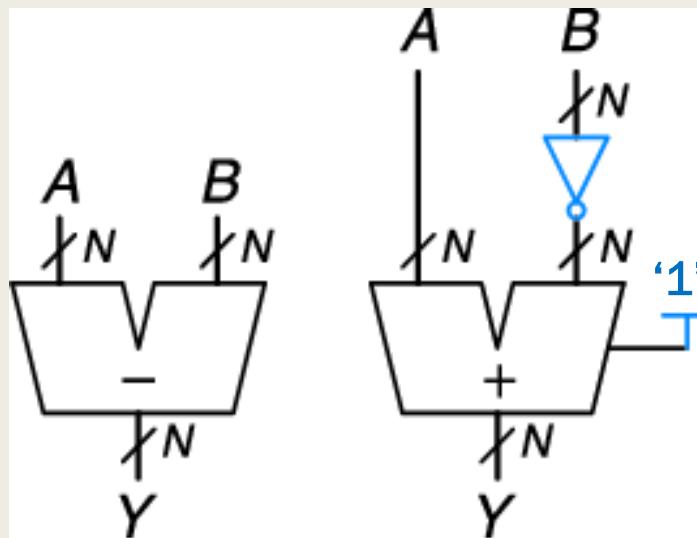
- N πλήρεις αθροιστές του ενός bit συνδεδεμένοι αλυσιδωτά
- Το κρατούμενο διαδίδεται μέσω της αλυσίδας κρατουμένου
- Η έξοδος C_{out} του ενός βάρους χρησιμεύει ως είσοδος C_{in} του επόμενου βάρους
- Γενικά είναι αργός (χρησιμοποιείται μόνο στα FPGA)



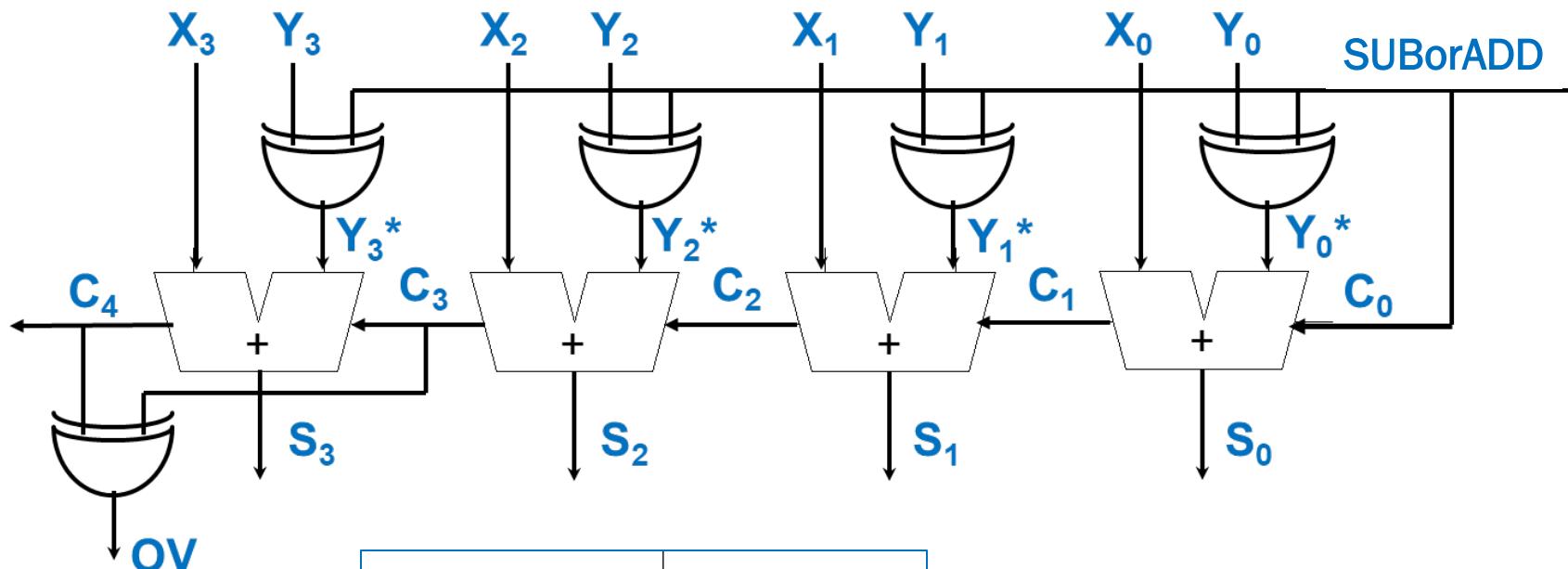
Αφαιρέτες των N bit

■ Αφαιρέτης κυμάτωσης κρατούμενου (ripple-carry subtracter)

- N πλήρεις αθροιστές του ενός bit συνδεδεμένοι αλυσιδωτά
- Το κρατούμενο διαδίδεται μέσω της αλυσίδας κρατουμένου
- Η έξοδος C_{out} του ενός βάρους χρησιμεύει ως είσοδος C_{in} του επόμενου βάρους
- Αντιστρέφονται τα ψηφία του αφαιρετέου
- Η είσοδος C_{in} του μικρότερου βάρους αρχικοποιείται στο 1
- Γενικά είναι αργός (χρησιμοποιείται μόνο στα FPGA)



Αθροιστής/αφαιρέτης με επιλογή και υπερχείλιση προσημασμάτων



C_3	X_3	Y_3^*	C_4	S_3
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

OV

$X_3 = Y_3^* = 1 \& S_3 = 0$

$X_3 = Y_3^* = 0 \& S_3 = 1$

4 bit

Πρόσθεση $SUBorADD = 0$

$C_3 C_2 C_1 C_0 = 0$

$X_3 X_2 X_1 X_0 + Y_3 Y_2 Y_1 Y_0$

$\Theta_4 S_3 S_2 S_1 S_0$

Αφαίρεση $SUBorADD = 1$

$C_3 C_2 C_1 C_0 = 1$

$X_3 X_2 X_1 X_0 + Y_3 Y_2 Y_1 Y_0$

$\Theta_4 S_3 S_2 S_1 S_0$

Τάση Τροφοδοσίας και Γείωση

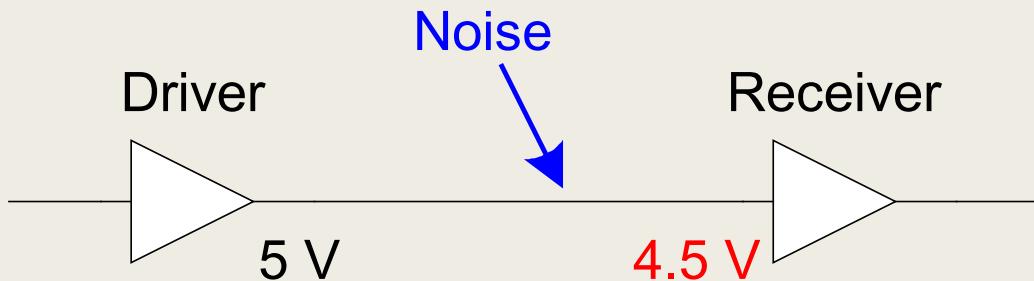
- Η χαμηλότερη τάση σε ένα σύστημα ονομάζεται **γείωση** (ground) ή **GND**
 - Θεωρούμε ότι η χαμηλότερη τάση είναι OV
- Η υψηλότερη τάση σε ένα σύστημα, δηλαδή η τάση τροφοδοσίας, προέρχεται από την τροφοδοσία και συμβολίζεται με **V_{DD}**
 - Στις δεκαετίες του 1970 και 1980 είχε την τιμή 5V
 - Κάθως η τεχνολογία των τσιπ εξελίσσεται η υψηλότερη τάση μειώνεται συνεχώς (από τα 5V στο 1V)

Λογικά επίπεδα

- Οι μεταβλητές (όπως η τάση στην έξοδο ενός ηλεκτρονικού κυκλώματος) αναπαρίστανται με συνεχείς φυσικές τιμές
- Ένα ψηφιακό σύστημα όμως χρησιμοποιεί μεταβλητές που παίρνουν **διακριτές τιμές**
- Επομένως ένας κατασκευαστής ψηφιακών συστημάτων πρέπει να βρεί μία αντιστοίχιση **που να συνδέει τις συνεχείς τιμές με τις διακριτές τιμές**
- Για παράδειγμα: Η αντιστοίχιση ενός δυαδικού σήματος A με την τάση σε μία έξοδο μίας πύλης (κατά τη θετική λογική)
 - Τα 0V αντιστοιχούν στο A=0 (*LOW*)
 - Τα 5V αντιστοιχούν στο A=1 (*HIGH*)
- Λόγω ανοχής στον θόρυβο, θα πρέπει και κάποιες κοντινές τάσεις να αντιστοιχούν στο A=0 ή A=1
 - π.χ. τα 4,9V στα 5V αντιστοιχούν στο A=1
- Τί συμβαίνει όμως με ποιο ενδιάμεσες τιμές τάσεων (π.χ. 3,2V);

Τι είναι ο θόρυβος (noise);

- Η ύπαρξη θορύβου είναι αρκετά συχνό φαινόμενο ανάμεσα στον οδηγό και στον δέκτη
- Οτιδήποτε μπορεί να επηρεάσει ή να υποβαθμίσει ένα σήμα
 - Π.χ., αντίσταση, θόρυβος τάσης τροφοδοσίας, ηλεκτρομαγνητική αλληλεπίδραση μεταξύ γειτονικών συρμάτων, κλπ.
- Για παράδειγμα: Η πύλη «οδηγός» έχει έξοδο 5V. Λόγω της αντίστασης απ' το μακρύ σύρμα που συνδέει την έξοδο του οδηγού με την είσοδο του δέκτη τελικά ο δέκτης θα λάβει 4.5V

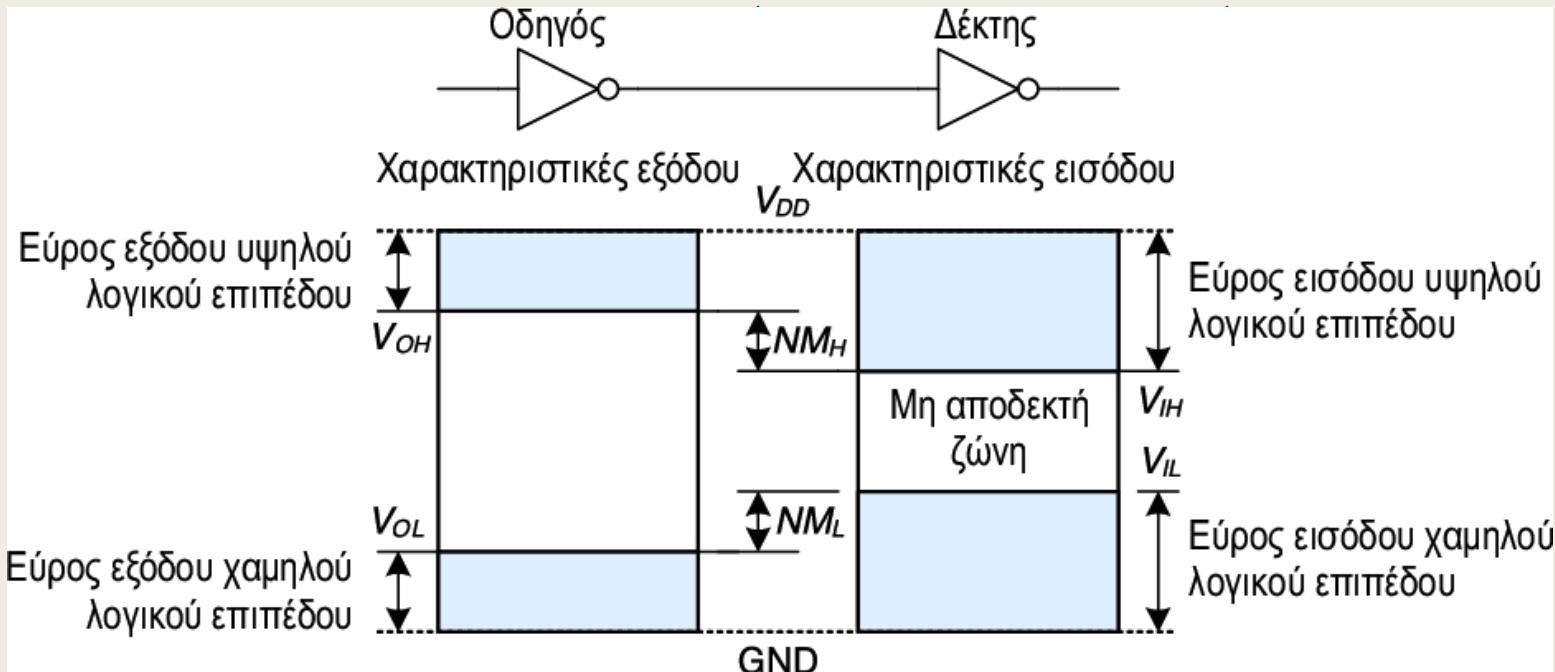


Λογικά Επίπεδα

- Η αντιστοίχηση μιας συνεχούς μεταβλητής σε μία διακριτή δυαδική μεταβλητή πραγματοποιείται με τον ορισμό των λογικών επιπέδων
- Ορίζονται διαφορετικά λογικά επίπεδα για εισόδους και εξόδους, ώστε να παρέχεται προστασία από τον θόρυβο
 - V_{OH} : ελάχιστη τάση στην έξοδο ενός οδηγού που αντιστοιχεί στην τιμή '1' (HIGH)
 - V_{OL} : μέγιστη τάση στην έξοδο ενός οδηγού που αντιστοιχεί στην τιμή '0' (LOW)
 - V_{IH} : ελάχιστη τάση στην είσοδο ενός δέκτη που αντιστοιχεί στην τιμή '1' (HIGH)
 - V_{IL} : μέγιστη τάση στην είσοδο ενός δέκτη που αντιστοιχεί στην τιμή '0' (LOW)



Περιθώρια Θορύβου (noise margins)



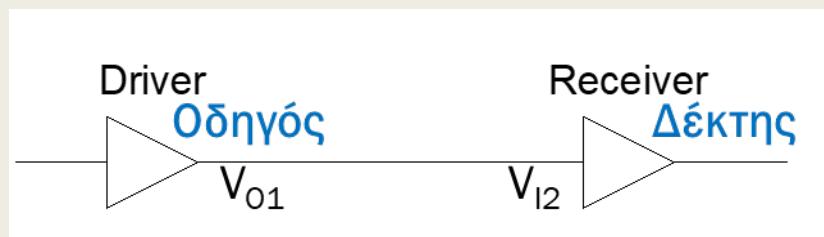
Υψηλό Περιθώριο Θορύβου: $NM_H = V_{OH} - V_{IH}$

Χαμηλό Περιθώριο Θορύβου : $NM_L = V_{IL} - V_{OL}$

Περιθώρια Θορύβου

- Για να ερμηνευθεί σωστά η έξοδος του οδηγού στην είσοδο του δέκτη, πρέπει να επιλέξουμε τέτοιες τιμές λογικών επιπέδων ώστε να ισχύουν οι ανισότητες:
 - $V_{OL} < V_{IL}$ και $V_{OH} > V_{IH}$
- Άρα, ακόμα και αν η τάση στην έξοδο του οδηγού έχει μεταβληθεί από κάποια ποσότητα θορύβου, η είσοδος του δέκτη θα εξακολουθεί να είναι σε θέση να ανιχνεύει το σωστό λογικό επίπεδο.
- Το **περιθώριο θορύβου (noise margin)** είναι εκείνη η ποσότητα του θορύβου που θα μπορούσε να προστεθεί στην έξοδο του οδηγού στη χειρότερη περίπτωση, η οποία επιτρέπει ωστόσο στο σήμα να ερμηνευθεί σωστά ως μια έγκυρη είσοδος του δέκτη
- Αν για κάποιον λόγο, π.χ. την παρουσία θορύβου ή την ύπαρξη κάποιου ελαττωματικού εξαρτήματος, η είσοδος του δέκτη έχει τιμή που εμπίπτει στη **μη αποδεκτή ζώνη (forbidden zone)** μεταξύ V_{IL} και V_{IH} , τότε η συμπεριφορά της πύλης είναι απρόβλεπτη.

Παράδειγμα 1.18



- Θεωρήστε το κύκλωμα της Εικόνας.
 - Το V_{O1} είναι η τάση εξόδου του οδηγού, ενώ το V_{I2} είναι η τάση εισόδου του δέκτη
 - Και οι δύο buffer έχουν $V_{DD} = 5V$, $V_{IL} = 1,35V$, $V_{IH} = 3,15V$, $V_{OL} = 0,33V$ και $V_{OH} = 3,84V$.
 - Ποια είναι τα περιθώρια θορύβου για το χαμηλό και το υψηλό λογικό επίπεδο του buffer;
 - Μπορεί το κύκλωμα να ανέχεται 1 V θορύβου μεταξύ των V_{O1} και V_{I2} ;

Παράδειγμα 1.18



- Θεωρήστε το κύκλωμα της Εικόνας.
 - Το V_{O1} είναι η τάση εξόδου του οδηγού, ενώ το V_{I2} είναι η τάση εισόδου του δέκτη
 - Και οι δύο buffer έχουν $V_{DD} = 5V$, $V_{IL} = 1,35V$, $V_{IH} = 3,15V$, $V_{OL} = 0,33V$ και $V_{OH} = 3,84V$.
 - Ποια είναι τα περιθώρια θορύβου για το χαμηλό και το υψηλό λογικό επίπεδο του buffer;
 - Μπορεί το κύκλωμα να ανέχεται 1 V θορύβου μεταξύ των V_{O1} και V_{I2} ;
- Τα περιθώρια θορύβου του αντιστροφέα είναι τα εξής:
 - $NM_L = V_{IL} - V_{OL} = 1,35V - 0,33V = 1,02V$
 - $NM_H = V_{OH} - V_{IH} = 3,84V - 3,15V = 0,69V$
 - Το κύκλωμα μπορεί να ανεχθεί 1V θορύβου όταν η έξοδος είναι LOW ($NM_L = 1,02V$), αλλά όχι όταν η έξοδος είναι HIGH ($NM_H = 0,69V$).
 - Εάν $V_{O1} = V_{OH} = 3,84V$, τότε εξαιτίας του θορύβου $V_{I2} = 3,84V - 1V = 2,84V < V_{IH} = 3,15V$, άρα ο δέκτης ενδέχεται να μην ανιχνεύσει μια σωστή είσοδο HIGH, αλλά μία λανθασμένη είσοδο LOW

Λογικές οικογένειες

- Από το 1970 μέχρι και το 1990 υπήρχαν τέσσερις λογικές οικογένειες που κυριάρχησαν στην αγορά
 - Λογική τρανζίστορ-τρανζίστορ(*Transistor-Transistor Logic, TTL*)
 - Λογική συμπληρωματικού ημιαγωγού οξειδίου μετάλλου (*Complementary Metal-Oxide-Semiconductor Logic, CMOS*)
 - Λογική TTL χαμηλής τάσης (*Low Voltage TTL Logic, LV-TTL*)
 - Λογική CMOS χαμηλής τάσης(*Low Voltage CMOS Logic, LVC-MOS*).

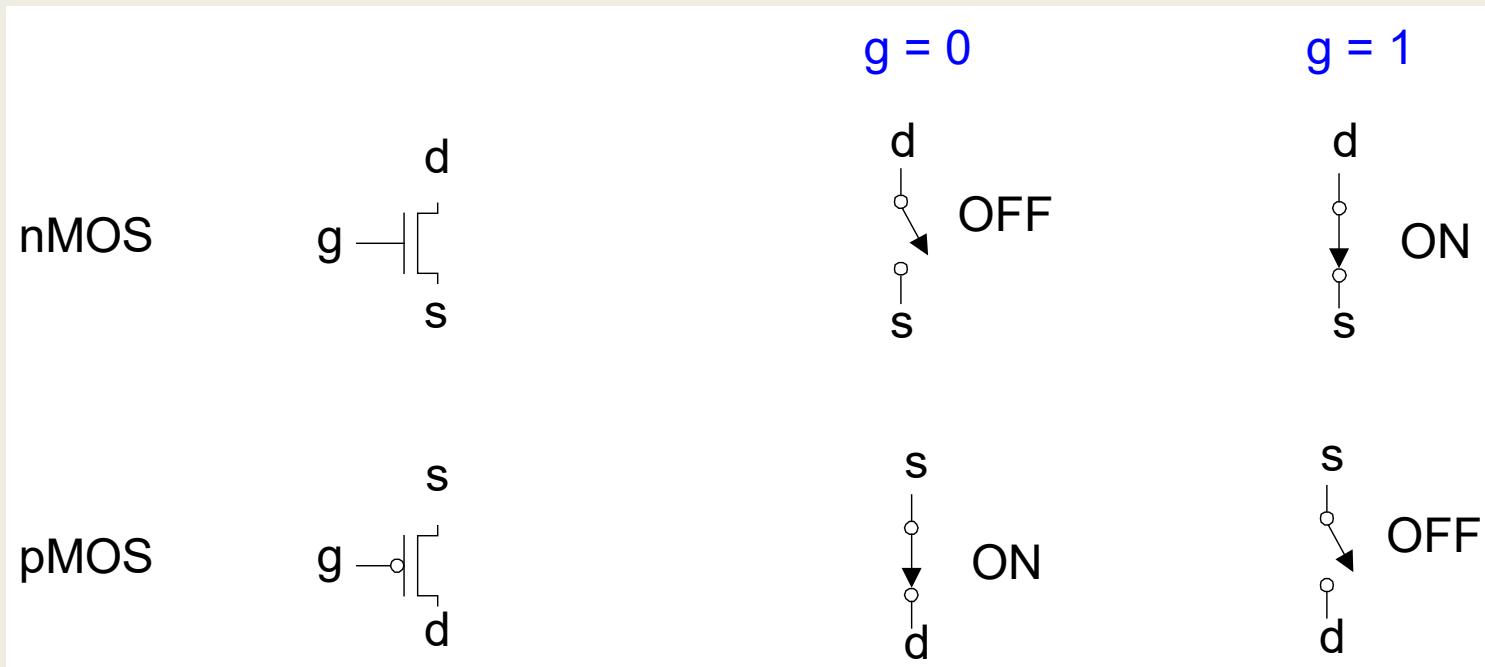
Λογική οικογένεια	V_{DD}	V_{IL}	V_{IH}	V_{OL}	V_{OH}
TTL	5(4.75-5.25)	0.8	2.0	0.4	2.4
CMOS	5(4.5-6)	1.35	3.15	0.33	3.84
LV-TTL	3.3(3-3.6)	0.8	2.0	0.4	2.4
LVC-MOS	3.3(3-3.6)	0.9	1.8	0.36	2.7

Τρανζίστορ

- Οι πρώτοι ηλεκτρονικοί υπολογιστές χρησιμοποιούσαν ηλεκτρονόμους ή λυχνίες κενού.
- Οι σύγχρονοι υπολογιστές χρησιμοποιούν ηλεκτρικούς διακόπτες οι οποίοι είναι επίσης γνώστοι σαν τρανζίστορ
 - *Τρανζίστορ διπολικής επαφής*
 - *Τρανζίστορ φαινομένου πεδίου ημιαγωγών μετάλλου οξειδίου (MOSFET - metal-oxide-semiconductor field-effect transistor)*
- Τα τρανζίστορ MOSFET χρησιμοποιούνται σαν δομικά στοιχεία σχεδόν σε όλα τα ψηφιακά συστήματα.

Τρανζίστορ MOSFET ως διακόπτης

- Οι λογικές πύλες κατασκευάζονται από τρανζίστορ φαινομένου πεδίου ημιαγωγών μετάλλου οξειδίου (MOSFET) που λειτουργούν ως διακόπτες
- Υπάρχουν δύο είδη τρανζίστορ MOSFET: nMOS και pMOS
 - Η τάση στην **πύλη (gate)** ρυθμίζει τη ροή του ρεύματος μεταξύ της **πηγής (source)** και της **υποδοχής (drain)**

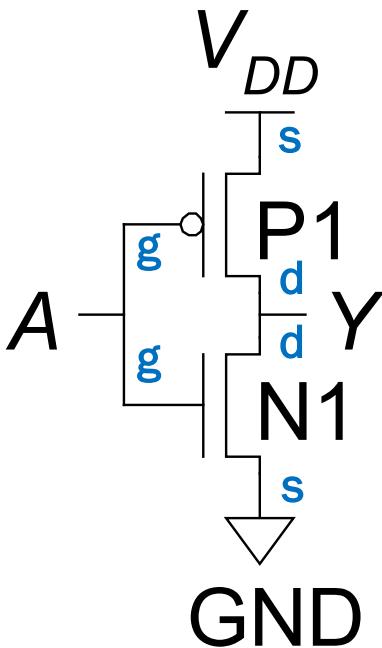


Λειτουργία τρανζίστορ MOSFET

- Το τρανζίστορ nMOS
 - δεν άγει (είναι OFF) όταν η πύλη (gate) είναι '0'
(έχει σαν τάση αντίστοιχη της λογικής τιμής '0')
 - άγει (είναι ON) όταν η πύλη (gate) είναι '1'
(έχει σαν τάση αντίστοιχη της λογικής τιμής '1')
- Το τρανζίστορ pMOS
 - δεν άγει (είναι OFF) όταν η πύλη είναι '1'
 - άγει (είναι ON) όταν η πύλη είναι '0'
- Όταν το τρανζίστορ δεν άγει (είναι OFF) έχει πολύ μεγάλη αντίσταση αλλά όχι άπειρη
 - Υπάρχει ένα μικρό ρεύμα διαρροής I_{DD}
- Όταν το τρανζίστορ άγει (είναι ON) μεταφέρει «καλά» από την πηγή (source) στην υποδοχή (drain)
 - εάν είναι nMOS το '0', συνεπώς συνδέεται η πηγή (s) στο GND
 - εάν είναι pMOS το '1', συνεπώς συνδέεται η πηγή (s) στο V_{DD}

Πύλη NOT σε τεχνολογία CMOS

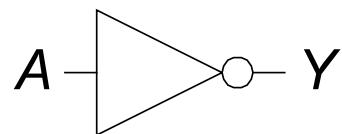
- Στην τεχνολογία **CMOS (complementary MOS)** χρησιμοποιούνται **μαζί** και τα δύο είδη τρανζίστορ MOSFET, δηλαδή **nMOS και pMOS**
- Πώς δημιουργείται μία **πύλη NOT** σε τεχνολογία CMOS;
- Σχηματικό διάγραμμα:
 - Στο τρανζίστορ nMOS (*N1*) συνδέεται η πηγή (*s*) στο GND (τρίγωνο)
 - Στο τρανζίστορ pMOS (*P1*) συνδέεται η πηγή (*s*) στο V_{DD} (γραμμή)
 - Οι πύλες (*g*) των *N1* και *P1* συνδέονται στην είσοδο *A*
 - Οι υποδοχές (*d*) των *N1* και *P1* συνδέονται στην έξοδο *Y*



CMOS Πύλη NOT

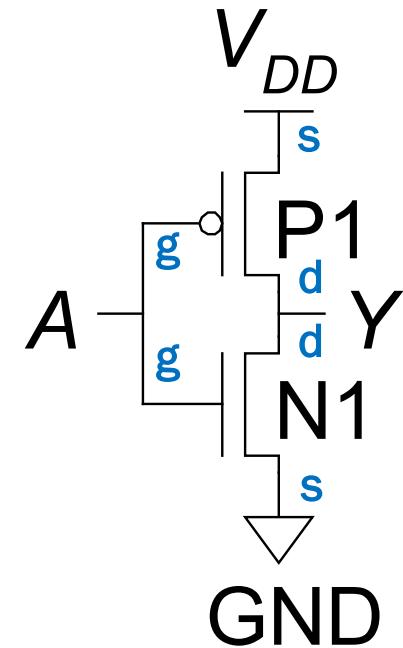
A	P1	N1	Y
0	ON	OFF	1
1	OFF	ON	0

NOT

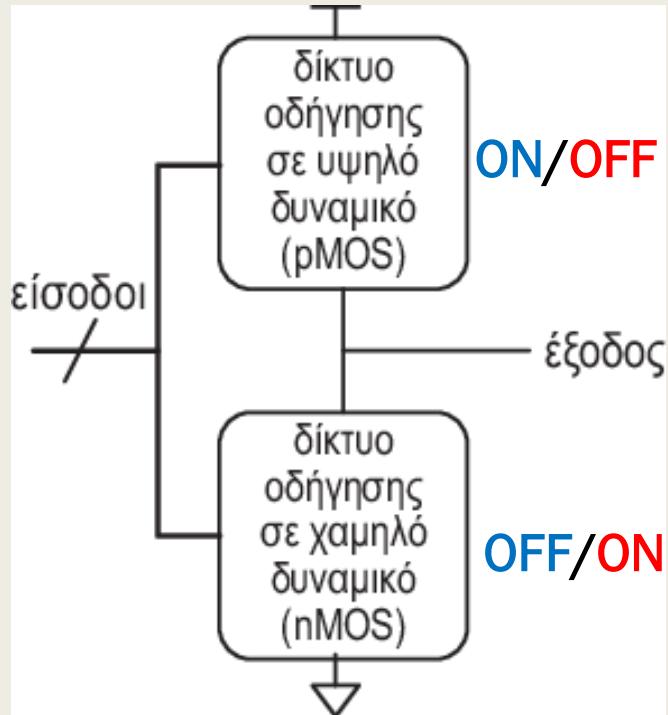


$$Y = \overline{A}$$

A	Y
0	1
1	0



Γενική μορφή λογικής πύλης CMOS



ON/OFF

OFF/ON

Δίκτυο οδήγησης της εξόδου σε υψηλό δυναμικό (ώστε όταν είναι ON να είναι η έξοδος '1') με τρανζίστορ pMOS. Το δίκτυο pMOS τοποθετείται ανάμεσα στην τάση V_{DD} και την έξοδο

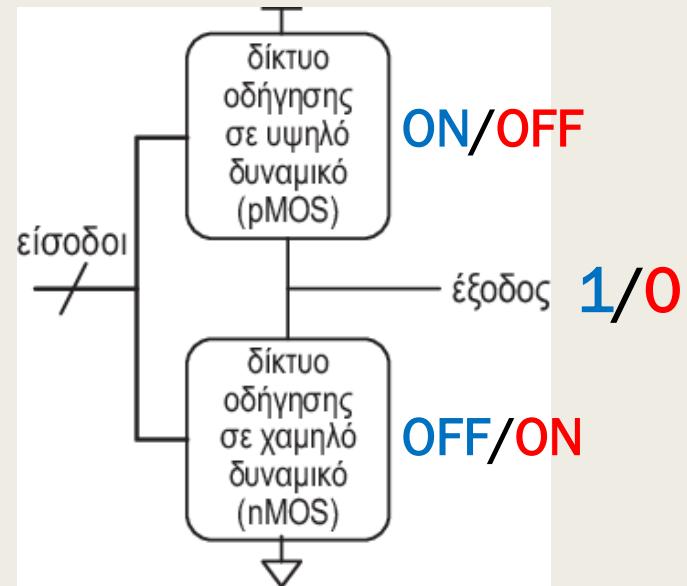
1/0

Δίκτυο οδήγησης της εξόδου σε χαμηλό δυναμικό (ώστε όταν είναι ON να είναι η έξοδος '0') με τρανζίστορ nMOS. Το δίκτυο nMOS τοποθετείται ανάμεσα στην έξοδο και την τάση OV (GND)

- Τα δίκτυα αποτελούνται από τρανζίστορ συνδεδεμένα παράλληλα ή σε σειρά.
 - Όταν τα τρανζίστορ είναι συνδεδεμένα παράλληλα, το δίκτυο είναι ON αν οποιοδήποτε από τα τρανζίστορ είναι ON.
 - Όταν τα τρανζίστορ είναι συνδεδεμένα σε σειρά, το δίκτυο είναι ON μόνο αν όλα τα τρανζίστορ είναι ON.

Γενική μορφή λογικής πύλης CMOS

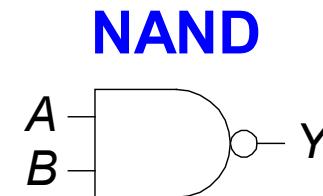
- Σε μια CMOS πύλη που λειτουργεί σωστά σε κάθε δεδομένη στιγμή, το ένα από τα δύο δίκτυα πρέπει να είναι ON και το άλλο OFF, έτσι ώστε η έξοδος να μην βραχυκυκλώνει ή να παραμένει μετέωρη.



- Κανόνας των *συμπληρωμάτων αγωγιμότητας*
 - Όταν τα τρανζίστορ nMOS είναι συνδεδεμένα σε *σειρά (πράξη AND)*, τα τρανζίστορ pMOS πρέπει να είναι συνδεδεμένα παράλληλα
 - Όταν τα τρανζίστορ nMOS είναι συνδεδεμένα *παράλληλα (πράξη OR)*, τα τρανζίστορ pMOS πρέπει να είναι συνδεδεμένα σε σειρά.
- Ξεκινάμε τη σχεδίαση πάντα από το δίκτυο nMOS υλοποιώντας τις απαιτούμενες πράξεις AND και OR
 - *Η έξοδος πάντα αντιστρέφεται (πράξη NOT) εκ κατασκευής*

CMOS πύλη NAND-2

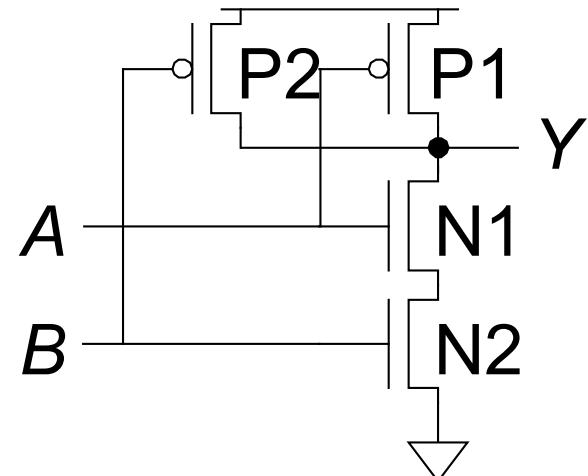
- Σχηματικό διάγραμμα:
 - Στο δίκτυο nMOS τα 2 τρανζίστορ συνδέονται στη σειρά, ώστε να γίνει η πράξη AND
 - Στο δίκτυο pMOS τα 2 τρανζίστορ συνδέονται παράλληλα, λόγω του κανόνα των συμπληρωμάτων αγωγιμότητας



$$Y = \overline{AB}$$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

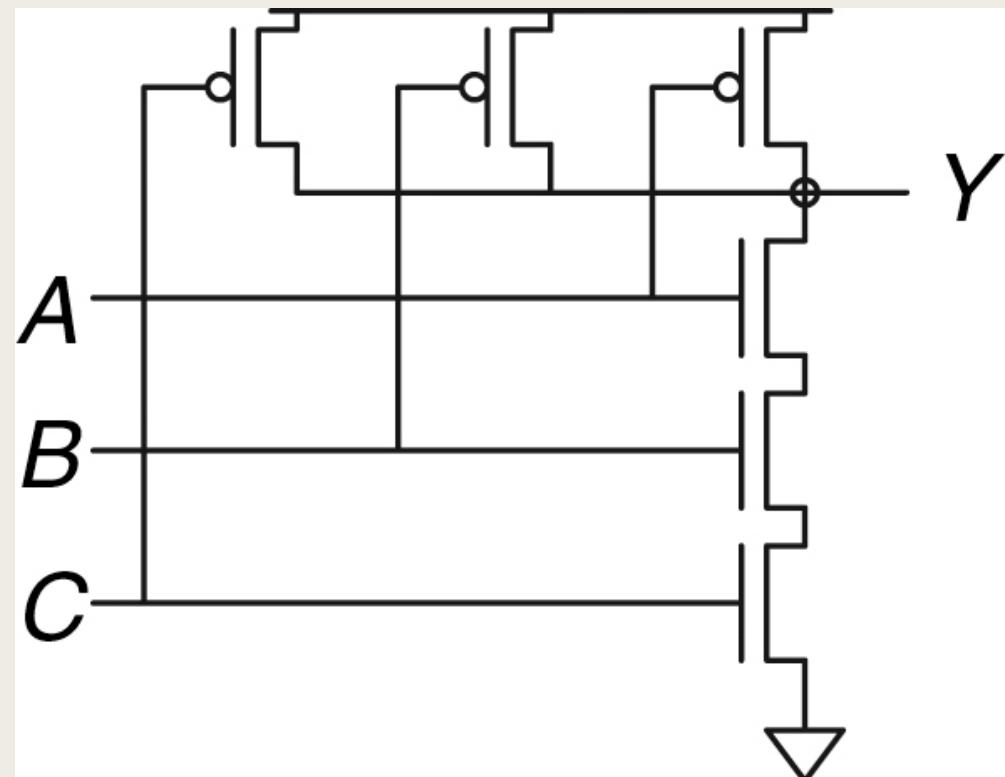
A	B	P1	P2	N1	N2	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0



CMOS πύλη NAND-3*

- Σχηματικό διάγραμμα:
 - Στο δίκτυο *n*MOS τα 3 τρανζίστορ συνδέονται στη σειρά (πράξη AND)
 - Στο δίκτυο *p*MOS τα 3 τρανζίστορ συνδέονται παράλληλα

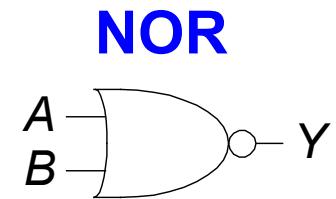
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



*Παράδειγμα 1.20

CMOS πύλη NOR-2*

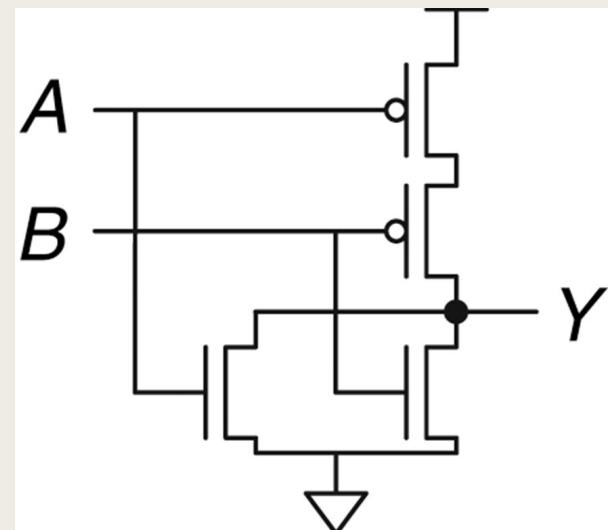
- Σχηματικό διάγραμμα:
 - Στο δίκτυο nMOS τα 2 τρανζίστορ συνδέονται παράλληλα, ώστε να γίνει η πράξη OR
 - Στο δίκτυο pMOS τα 2 τρανζίστορ συνδέονται σε σειρά, λόγω του κανόνα των συμπληρωμάτων αγωγιμότητας



$$Y = \overline{A + B}$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

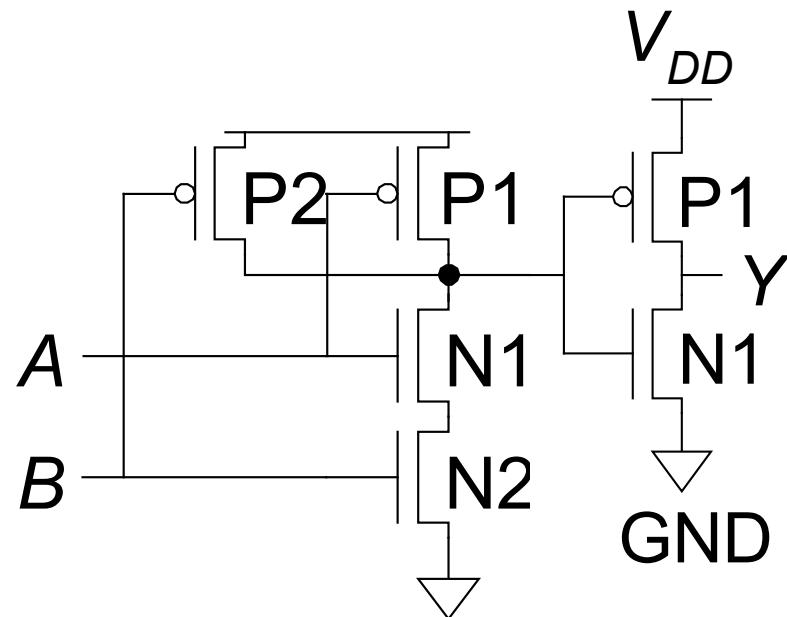
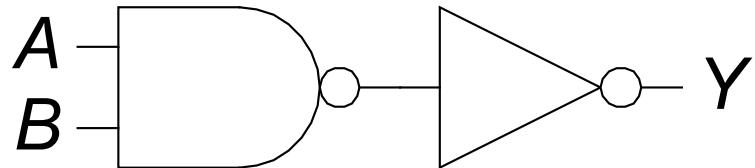
A	B	P1	P2	N1	N2	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	0
1	1	OFF	OFF	ON	ON	0



*Παράδειγμα 1.21

CMOS Πύλη AND-2*

- Πώς θα κατασκευάσετε μια πύλη AND?



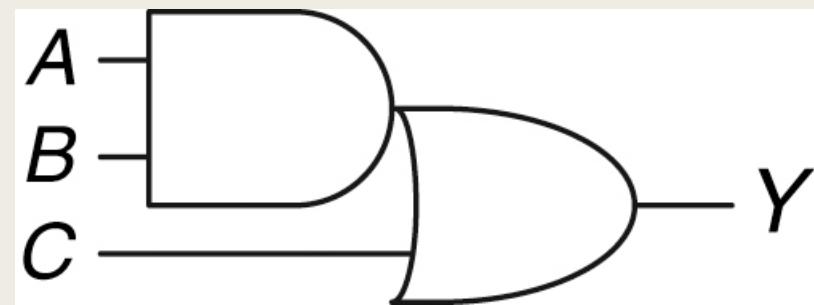
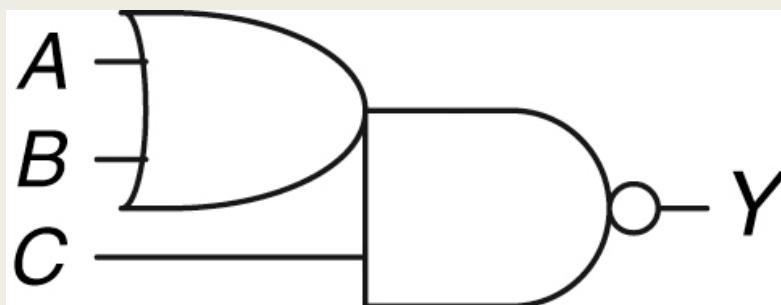
*Παράδειγμα 1.22

Επιλεγμένες ασκήσεις

■ Άσκηση 1.84

Σχεδιάστε ένα σχηματικό διάγραμμα για τις ακόλουθες πύλες CMOS. Χρησιμοποιήστε τον ελάχιστο δυνατό αριθμό τρανζίστορ.

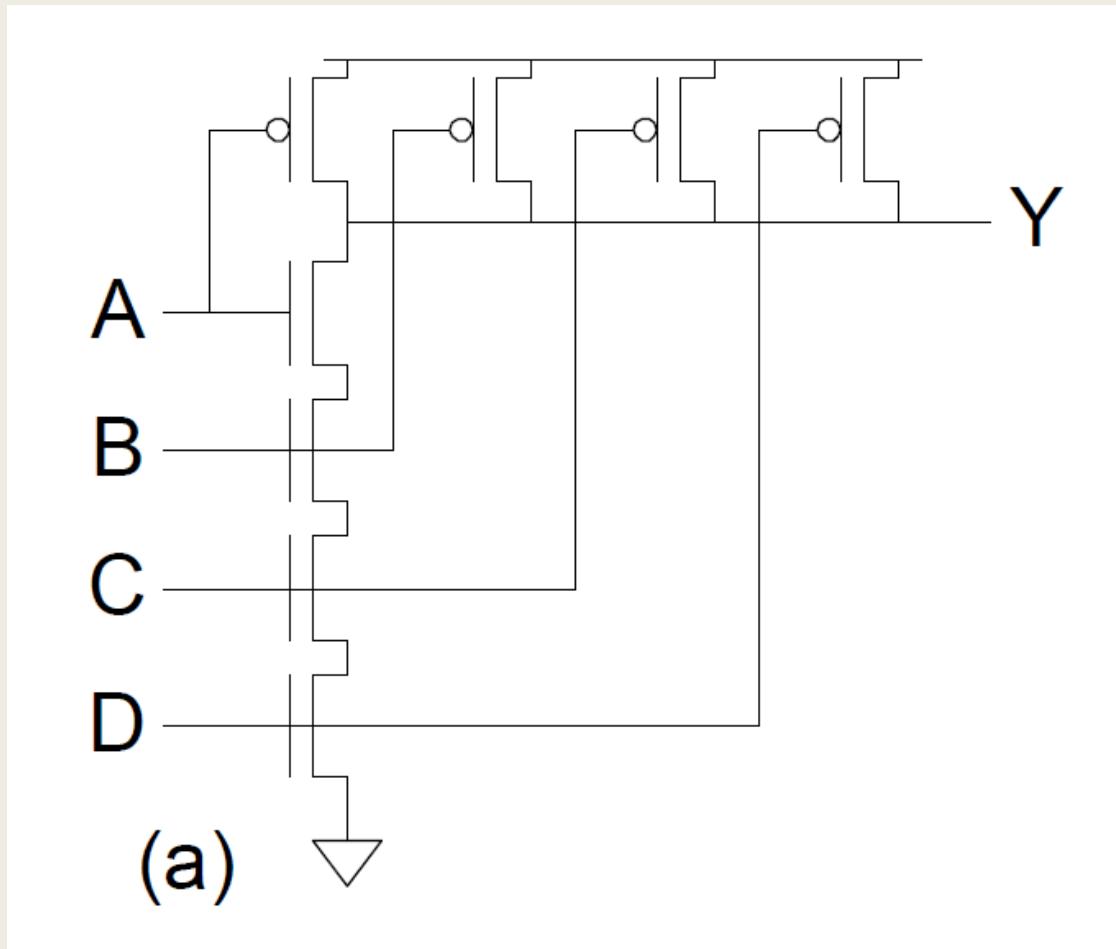
- (α) Πύλη NAND-4 με τέσσερις εισόδους
- (β) Πύλη OR-AND-INVERT με τρεις εισόδους
- (γ) Πύλη AND-OR με τρεις εισόδους



■ Άσκηση 1.84

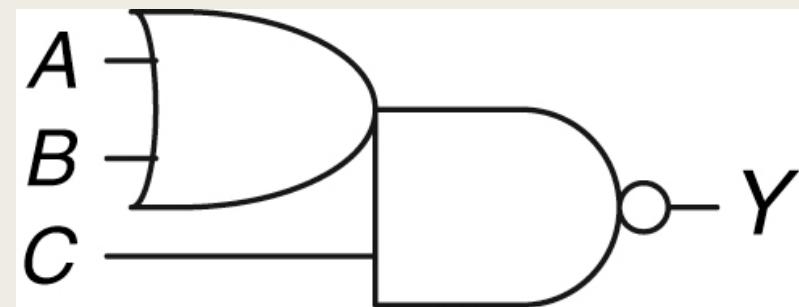
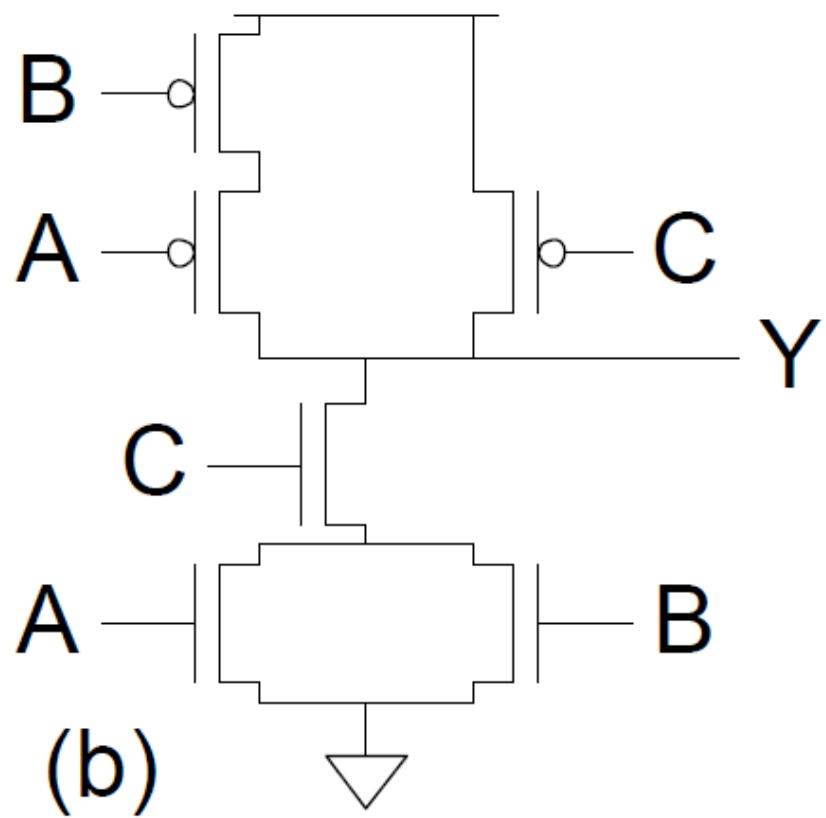
Σχεδιάστε ένα σχηματικό διάγραμμα για τις ακόλουθες πύλες CMOS. Χρησιμοποιήστε τον ελάχιστο δυνατό αριθμό τρανζίστορ.

- (α) Πύλη NAND-4 με τέσσερις εισόδους



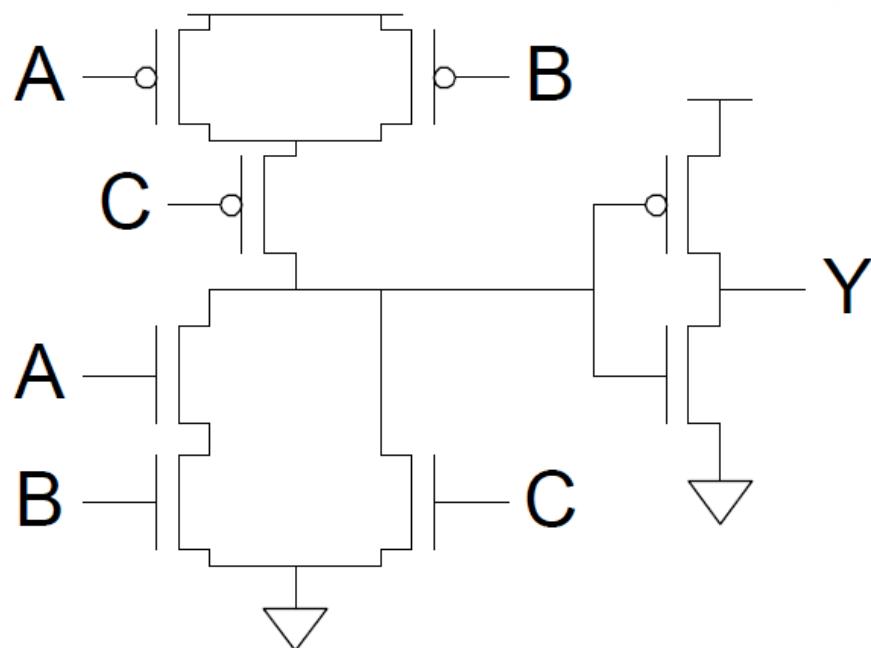
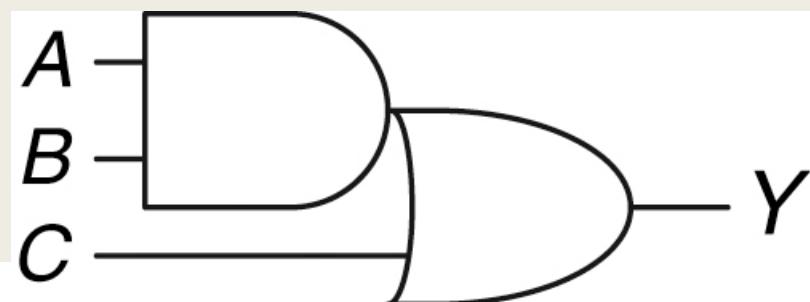
■ Άσκηση 1.84

Σχεδιάστε ένα σχηματικό διάγραμμα για τις ακόλουθες πύλες CMOS. Χρησιμοποιήστε τον ελάχιστο δυνατό αριθμό τρανζίστορ.
 - (β) Πύλη OR-AND-INVERT με τρεις εισόδους



■ Άσκηση 1.84

Σχεδιάστε ένα σχηματικό διάγραμμα για τις ακόλουθες πύλες CMOS. Χρησιμοποιήστε τον ελάχιστο δυνατό αριθμό τρανζίστορ.
 - (γ) Πύλη AND-OR με τρεις εισόδους



(c)

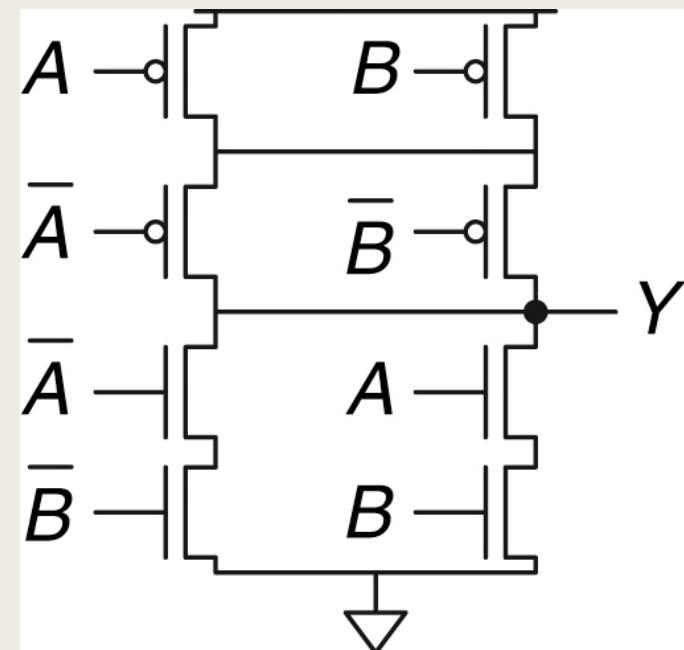
Επιλεγμένες ασκήσεις

■ Άσκηση 1.87

Συμπληρώστε τον πίνακα αληθείας για τη συνάρτηση που εκτελείται από την πύλη της Εικόνας

- Ο πίνακας θα πρέπει να έχει δύο εισόδους, τα A και B .
- Ποιο είναι το όνομα της συνάρτησης;

A	B	Y
0	0	
0	1	
1	0	
1	1	



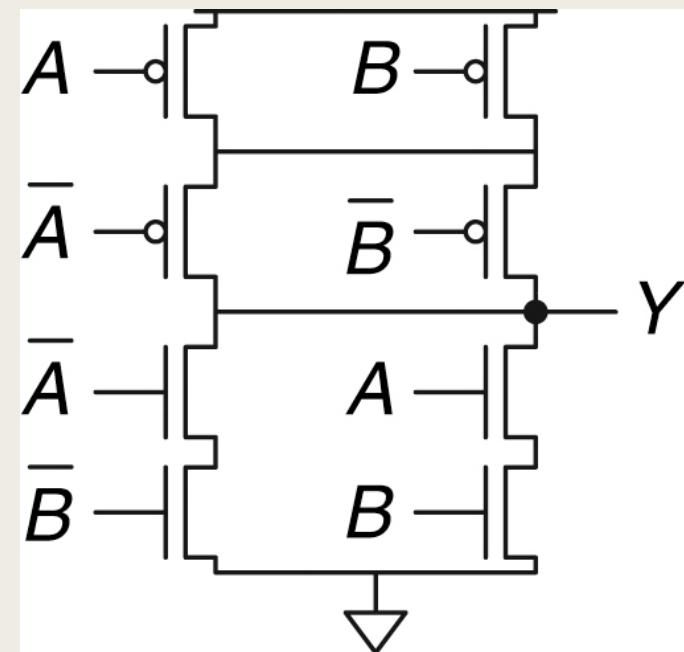
Επιλεγμένες ασκήσεις

■ Άσκηση 1.87

Συμπληρώστε τον πίνακα αληθείας για τη συνάρτηση που εκτελείται από την πύλη της Εικόνας

- Ο πίνακας θα πρέπει να έχει δύο εισόδους, τα A και B .
- Ποιο είναι το όνομα της συνάρτησης; **XOR**

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



Κατανάλωση Ισχύος

- Η ποσότητα της ενέργειας που καταναλώνεται ανά μονάδα χρόνου λέγεται **κατανάλωση ισχύος**
- Τα ψηφιακά συστήματα καταναλώνουν **δυναμική** και **στατική** ισχύς
 - **Δυναμική ισχύς:** η κατανάλωση ισχύος καθώς τα σήματα μεταβάλλονται μεταξύ του 0 και 1

$$P_{\text{dynamic}} = \frac{1}{2} C V_{DD}^2 f$$

όπου C η μέση χωρητικότητα, V_{DD} η τάση τροφοδοσίας και f η συχνότητα λειτουργίας του τσιπ

- **Στατική ισχύς:** η κατανάλωση ισχύος όταν τα τρανζίστορ είναι OFF λόγω του **ρεύματος διαρροής I_{DD}** στο τσιπ

$$P_{\text{static}} = I_{DD} V_{DD}$$

Παράδειγμα 1.23

- Ένα κινητό τηλέφωνο διαθέτει μπαταρία των 6 watt-hour (W-hr) και λειτουργεί στα 1,2 V
- Υποθέστε ότι, όταν το κινητό τηλέφωνο βρίσκεται σε χρήση, λειτουργεί στα 300 MHz, η μέση ποσότητα χωρητικότητας στο τσιπ η οποία εναλλάσσεται σε οποιαδήποτε δεδομένη στιγμή είναι ίση με 10 nF (10^{-8} Farad) και εκπέμπει επίσης 3 W ισχύος από την κεραία του.
- Όταν το τηλέφωνο δεν βρίσκεται σε χρήση, η δυναμική ενέργεια μειώνεται σχεδόν στο μηδέν επειδή είναι απενεργοποιημένη η επεξεργασία του σήματος. Όμως, το τηλέφωνο αντλεί 40 mA του ρεύματος ηρεμίας ανεξάρτητα από το αν χρησιμοποιείται ή όχι.
- Υπολογίστε τη διάρκεια ζωής της μπαταρίας του τηλεφώνου (α) αν δεν χρησιμοποιείται, και (β) αν χρησιμοποιείται συνεχώς.

Παράδειγμα 1.23 (Λύση-1/2)

$$P_{\text{static}} = I_{DD} V_{DD}$$

- Η στατική ισχύς είναι $P_{\text{static}} = (40 \text{ mA})(1,2 \text{ V}) = 48 \text{ mW}$.
- (α) Αν το τηλέφωνο δεν χρησιμοποιείται, τότε αυτή είναι και η μόνη κατανάλωση ισχύος, άρα η διάρκεια ζωής της μπαταρίας είναι $(6 \text{ Whr})/(0.048 \text{ W}) = 125$ ώρες (περίπου 5 μέρες).

Παράδειγμα 1.23 (Λύση2/2)

$$P_{\text{dynamic}} = \frac{1}{2} C V_{DD}^2 f$$

- Η δυναμική ισχύς είναι $P_{\text{dynamic}} = (0,5)(10^{-8} \text{ F})(1,2 \text{ V})^2(3 \times 10^8 \text{ Hz}) = 2,16 \text{ W}$.
- (β) Αν το τηλέφωνο χρησιμοποιείται, τότε η συνολική ενεργή ισχύς που καταναλώνεται (η δυναμική ισχύς μαζί με τη στατική ισχύ και την ισχύ εκπομπής) είναι $2,16 \text{ W} + 0,048 \text{ W} + 3 \text{ W} = 5,2 \text{ W}$
- Άρα η διάρκεια ζωής της μπαταρίας είναι $6 \text{ W-hr}/5,2 \text{ W} = 1,15 \text{ ώρες}$.

Άσκηση επανάληψης - 1

Ένας επεξεργαστής των 4-bit μετά το τέλος των αριθμητικών του πράξεων (πρόσθεση ή αφαίρεση σε συμπλήρωμα ως προς 2) ενημερώνει και τις αντίστοιχες σημαίες N (Negative), Z (Zero), C (Carry), V (Overflow). N=1, όταν το αποτέλεσμα είναι ένας αρνητικός προσημασμένος αριθμός. Z=1, όταν το αποτέλεσμα είναι μηδέν. C=1, όταν υπάρχει στο αποτέλεσμα υπερχείλιση μη προσημασμένων αριθμών. V=1, όταν υπάρχει στο αποτέλεσμα υπερχείλιση προσημασμένων αριθμών.

Ποιες είναι οι τιμές των σημαιών NZCV μετά την εκτέλεση της πράξης A-B, όπου A=0x8 και B=0x8; (Τα A και B δίδονται στο 16-δικό σύστημα αρίθμησης). Επιλέξτε τη σωστή απάντηση

- A)NZCV="1001"
- B)NZCV="0111"
- Γ)NZCV="1001"
- Δ)NZCV="1010"
- Ε)Δεν ξέρω/Δεν απαντώ

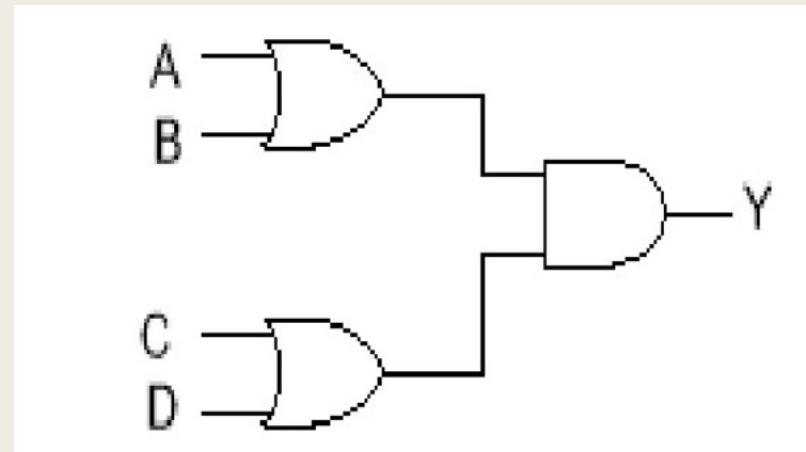
Άσκηση επανάληψης - 1

Ένας επεξεργαστής των 4-bit μετά το τέλος των αριθμητικών του πράξεων (πρόσθεση ή αφαίρεση σε συμπλήρωμα ως προς 2) ενημερώνει και τις αντίστοιχες σημαίες N (Negative), Z (Zero), C (Carry), V (Overflow). N=1, όταν το αποτέλεσμα είναι ένας αρνητικός προσημασμένος αριθμός. Z=1, όταν το αποτέλεσμα είναι μηδέν. C=1, όταν υπάρχει στο αποτέλεσμα υπερχείλιση μη προσημασμένων αριθμών. V=1, όταν υπάρχει στο αποτέλεσμα υπερχείλιση προσημασμένων αριθμών.

Ποιες είναι οι τιμές των σημαιών NZCV μετά την εκτέλεση της πράξης A-B, όπου A=0x8 και B=0x8; (Τα A και B δίδονται στο 16-δικό σύστημα αρίθμησης). Επιλέξτε τη σωστή απάντηση

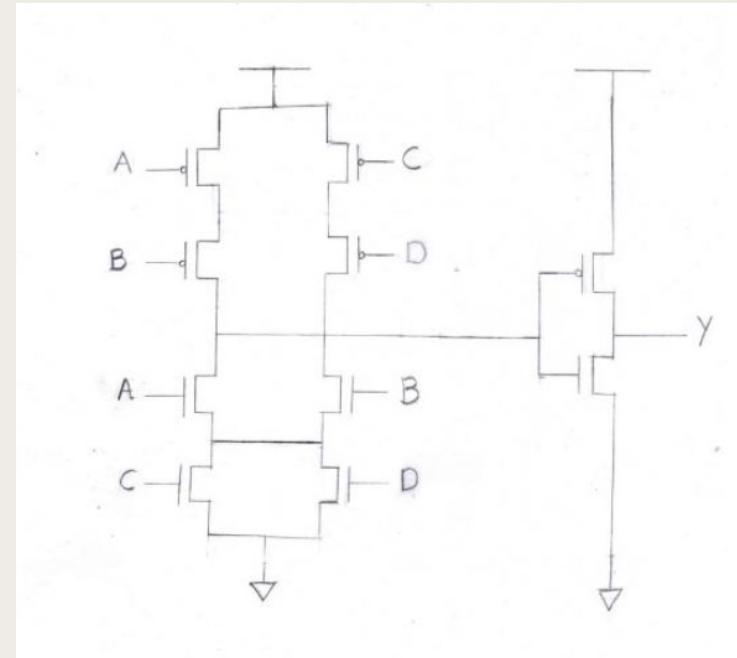
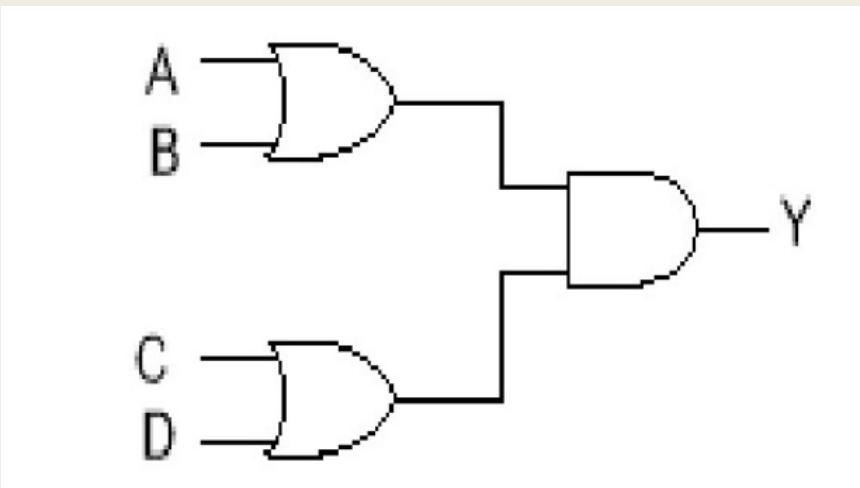
- A)NZCV="1001"
- B)NZCV="0111" (σωστό)
- Γ)NZCV="1001"
- Δ)NZCV="1010"
- Ε)Δεν ξέρω/Δεν απαντώ

Άσκηση επανάληψης - 2



Βρείτε το κύκλωμα CMOS που υλοποιεί το ανωτέρω κύκλωμα

Άσκηση επανάληψης - 2





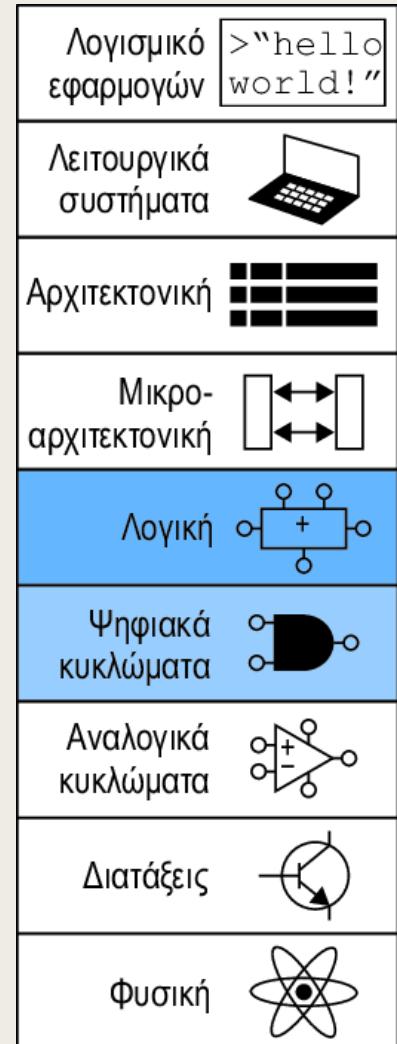
Κεφάλαιο 2

Σχεδίαση συνδυαστικής λογικής

Γιώργος Παπαδημητρίου, Αντώνης Πασχάλης,
Βασίλης Καρακώστας

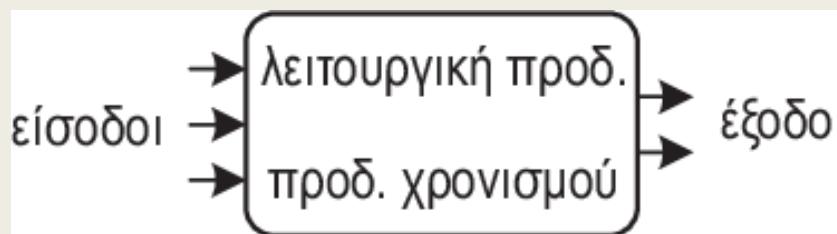
Περιεχόμενα κεφαλαίου 2

- Ψηφιακά κυκλώματα
- Εξισώσεις Boole
- Άλγεβρα Boole
- Σχηματικά διαγράμματα
- Πολυεπίπεδη συνδυαστική λογική
- Μη αποδεκτη τιμή X / Μετέωρη τιμή Z
- Χάρτες Karnaugh
- Κώδικες BCD και Gray
- Πολυπλέκτες / Αποκωδικοποιητές
- Χρονισμός συνδυαστικής λογικής



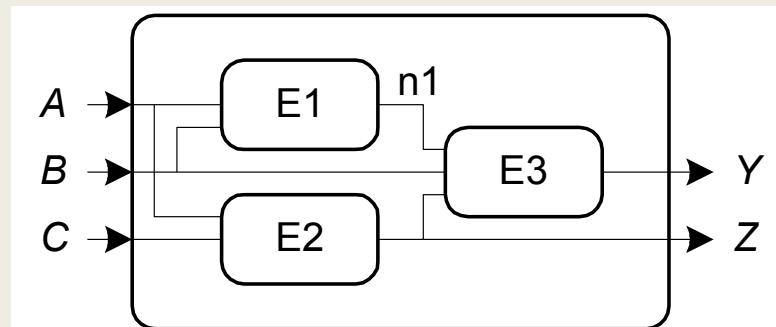
Ψηφιακά κυκλώματα

- Ένα **ψηφιακό κύκλωμα** είναι ένα δίκτυο στοιχείων που επεξεργάζεται μεταβλητές οι οποίες παίρνουν διακριτές τιμές
- Το κύκλωμα με βάση την αρχή της τμηματικότητας είναι ένα **μαύρο κουτί** με:
 - ένα ή περισσότερα **τερματικά σημεία εισόδου** (*input terminals*) που παίρνουν διακριτές τιμές,
 - ένα ή περισσότερα **τερματικά σημεία εξόδου** (*output terminals*) που παίρνουν διακριτές τιμές,
 - μια **λειτουργική προδιαγραφή** (*functional specification*) που περιγράφει τη σχέση μεταξύ εισόδων και εξόδων,
 - μια **προδιαγραφή χρονισμού** (*timing specification*) που περιγράφει την καθυστέρηση με την οποία αποκρίνονται οι έξοδοι στη μεταβολή των εισόδων.



Ψηφιακά κυκλώματα

- Ένα **ψηφιακό κύκλωμα** αποτελείται από **κόμβους** και **στοιχεία**
 - Το **στοιχείο (element)** (E_1, E_2, E_3) είναι και αυτό ψηφιακό κύκλωμα με εισόδους, εξόδους, και προδιαγραφή
 - Ο **κόμβος (node)** είναι ένα σύρμα, του οποίου η τάση μεταφέρει μια μεταβλητή που παίρνει διακριτές τιμές
- Οι κόμβοι ταξινομούνται ως **κόμβοι εισόδου**, **κόμβοι εξόδου** ή **εσωτερικοί κόμβοι**
 - Οι **κόμβοι εισόδου** (A, B, C) δέχονται τιμές από τον έξω κόσμο
 - Οι **κόμβοι εξόδου** (Y, Z) στέλνουν τιμές στον έξω κόσμο
 - Τα σύρματα που δεν είναι είσοδοι ή έξοδοι ονομάζονται **εσωτερικοί κόμβοι** (n_1)



Τύποι ψηφιακών κυκλωμάτων

- Τα ψηφιακά κυκλώματα ταξινομούνται σε **συνδυαστικά** (combinational) ή **ακολουθιακά** (sequential) κυκλώματα.
- **Συνδυαστική Λογική**
 - Οι έξοδοι ενός συνδυαστικού κυκλώματος εξαρτώνται **μόνο από τις τρέχουσες τιμές** των εισόδων
 - Οι **λογικές πύλες** είναι τα πιο απλά συνδυαστικά κυκλώματα
 - Δεν έχουν **μνήμη** (ούτε καταστάσεις)
- **Ακολουθιακή Λογική**
 - Οι έξοδοι ενός ακολουθιακού κυκλώματος εξαρτώνται **και από τις τρέχουσες και από τις προηγούμενες τιμές** των εισόδων
 - Έχουν μνήμη (αποθηκεύουν καταστάσεις)

Κανόνες συνδυαστικής σύνθεσης

- Οι κανόνες της **συνδυαστικής σύνθεσης** (combinational composition) μας λένε πώς μπορούμε να κατασκευάσουμε ένα μεγάλο συνδυαστικό κύκλωμα από μικρότερα στοιχεία συνδυαστικών κυκλωμάτων (αρχή της ιεραρχίας)
 - *To πιο μικρό στοιχείο είναι η λογική πύλη*
- Ένα ψηφιακό κύκλωμα είναι συνδυαστικό αν αποτελείται από αλληλοσυνδεόμενα στοιχεία κυκλωμάτων τέτοια ώστε:
 - *κάθε στοιχείο κυκλώματος είναι και το ίδιο συνδυαστικό.*
 - *κάθε κόμβος του κυκλώματος είτε είναι είσοδος, είτε συνδέεται σε ακριβώς μία έξοδο.*
 - *το κύκλωμα δεν περιέχει κυκλικές διαδρομές, δηλαδή κάθε διαδρομή μέσω των στοιχείων του κυκλώματος περνάει από κάθε κόμβο του το πολύ μία φορά.*

Κανόνες συνδυαστικής σύνθεσης

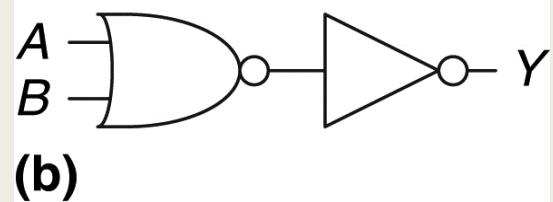
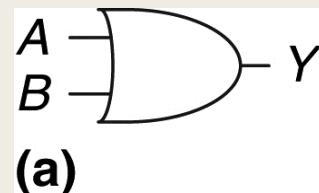
- Οι κανόνες της συνδυαστικής σύνθεσης μπορεί να είναι επαρκείς, αλλά δεν είναι αυστηρά υποχρεωτικοί
- Υπάρχουν ορισμένα κυκλώματα που, παρότι δεν υπάκουν τους συγκεκριμένους κανόνες, εξακολουθούν να είναι συνδυαστικά
- Προϋπόθεση είναι όμως να ικανοποιούν τη συνθήκη ότι
 - *οι έξοδοι εξαρτώνται μόνο από τις τρέχουσες τιμές των εισόδων*

Συνδυαστικά κυκλώματα

- Η λειτουργική προδιαγραφή ενός συνδυαστικού κυκλώματος εκφράζει τις τιμές των εξόδων σε σχέση με τις τρέχουσες τιμές των εισόδων
 - Είναι ένας πίνακας αλήθειας ή μία συνάρτηση (εξίσωση) Boolean
 - Για παράδειγμα, η συνάρτηση F της πύλης OR: $Y = F(A, B) = A + B$
- Μία συνάρτηση F μπορεί να έχει πολλές διαφορετικές υλοποιήσεις
 - Για παράδειγμα, η συνάρτηση F της πύλης OR μπορεί να υλοποιηθεί
 - (α) με την χρήση μιας πύλης OR
 - (β) με την χρήση μιας πύλης NOR και μίας πύλης NOT



$$Y = F(A, B) = A \cdot B$$

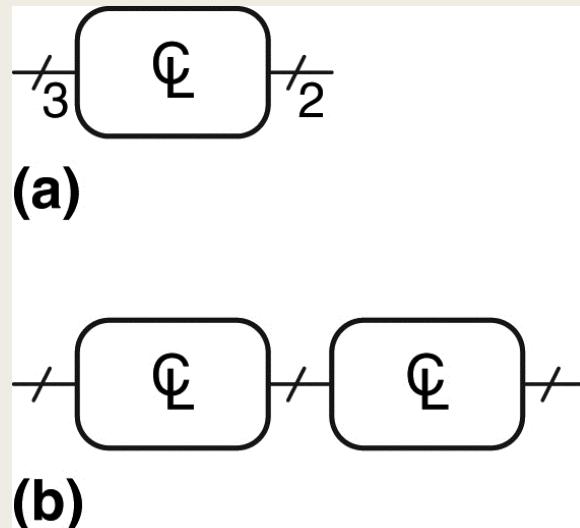


Συνδυαστικά κυκλώματα

- Ανάμεσα στις **πολλές διαφορετικές υλοποιήσεις** μίας συνάρτησης F , εμείς επιλέγουμε ποια υλοποίηση θα χρησιμοποιήσουμε ανάλογα με:
 - *Τα δομικά στοιχεία που έχουμε στη διάθεσή μας*
 - *Τους σχεδιαστικούς περιορισμούς που υπάρχουν, όπως:*
 - Επιφάνεια υλοποίησης
 - Ταχύτητα
 - Κατανάλωση ισχύος
 - Χρόνος σχεδίασης

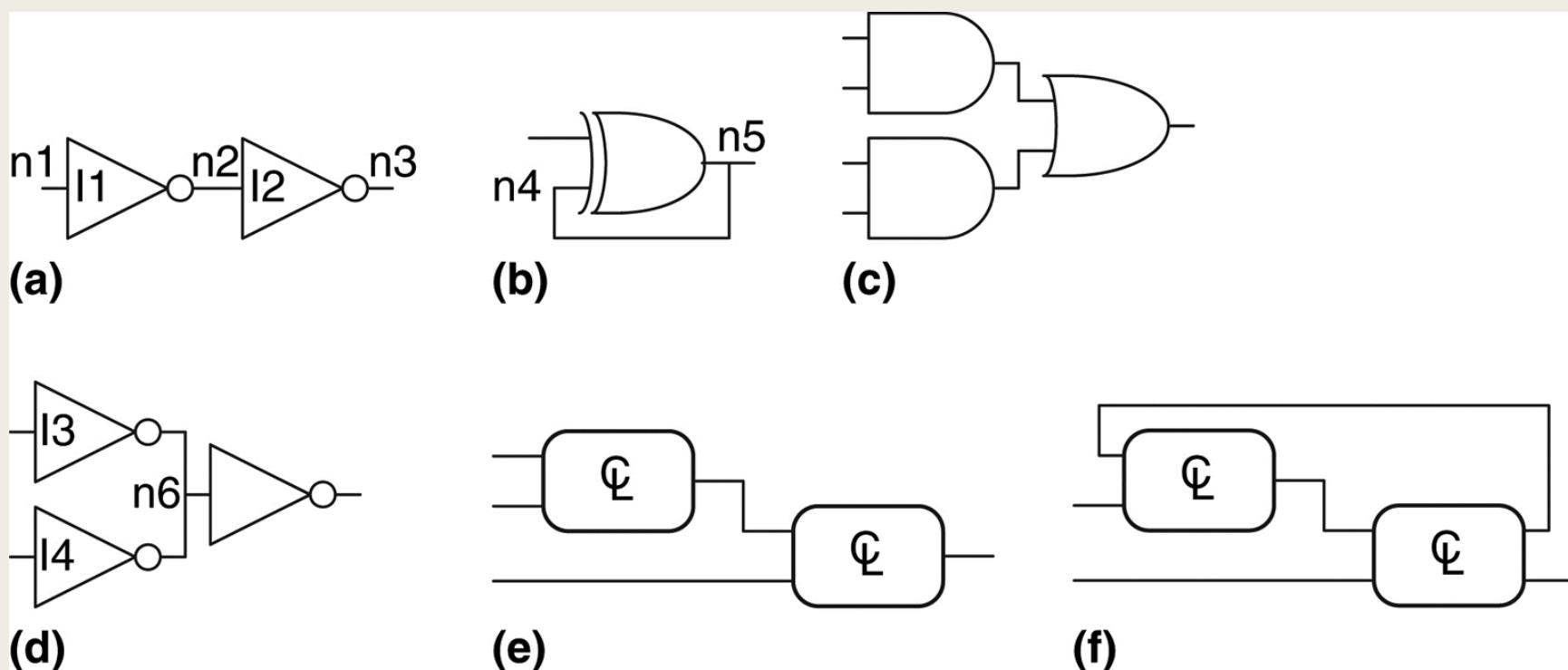
Δίαυλος (Bus)

- Σε μια προσπάθεια να απλοποιήσουμε τα διαγράμματα, συχνά χρησιμοποιούμε μία γραμμή με μια κάθετο (/) που την τέμνει, καθώς και έναν αριθμό δίπλα από αυτή για να συμβολίσουμε έναν **δίαυλο** (bus), δηλαδή μια δέσμη πολλών σημάτων
 - Ο αριθμός καθορίζει το **πλήθος των σημάτων (των bit)** στον δίαυλο
 - Αν το πλήθος των bit δεν παίζει ρόλο ή δεν είναι προφανές από τα συμφραζόμενα, η κάθετος ενδέχεται να μην συνοδεύεται από αριθμό



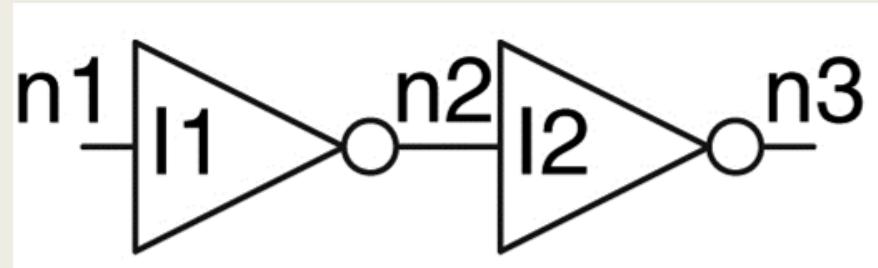
Παράδειγμα 2.1

- Ποια από τα παρακάτω κυκλώματα είναι συνδυαστικά?



Παράδειγμα 2.1 (συνέχεια)

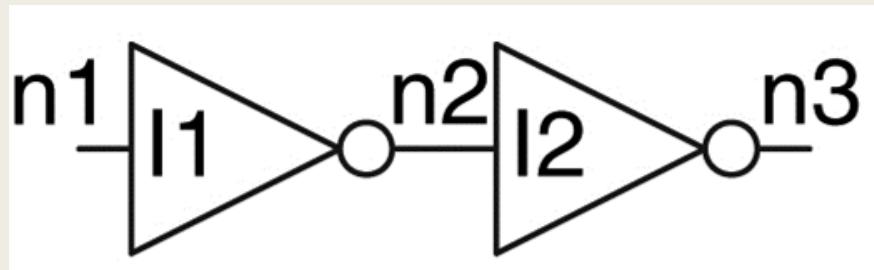
- Το κύκλωμα (α)



Παράδειγμα 2.1 (συνέχεια)

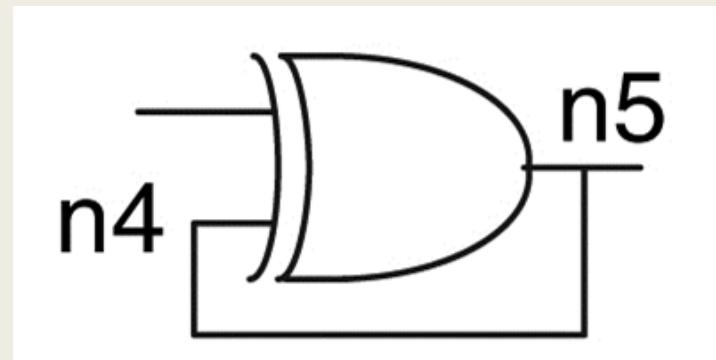
- Το κύκλωμα (α) **είναι συνδυαστικό**

- Έχει κατασκευαστεί από δύο στοιχεία συνδυαστικών κυκλωμάτων (αντιστροφείς I_1 και I_2)
- Διαθέτει τρεις κόμβους: n_1 , n_2 και n_3
- Ο κόμβος n_1 είναι είσοδος του κυκλώματος και του αντιστροφέα I_1
- Ο κόμβος n_2 είναι εσωτερικός κόμβος, ο οποίος αποτελεί την έξοδο του I_1 και την είσοδο του I_2
- Ο κόμβος n_3 είναι η έξοδος του κυκλώματος και του αντιστροφέα I_2



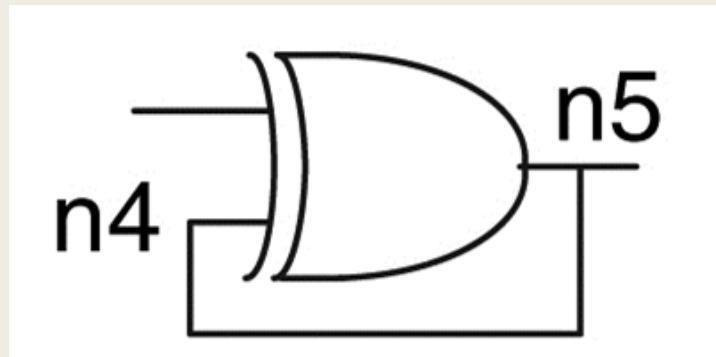
Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (β)



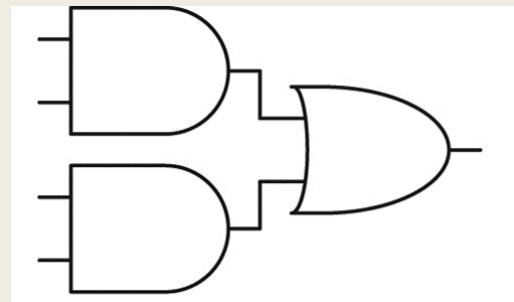
Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (β) **δεν είναι συνδυαστικό**
 - Υπάρχει κυκλική διαδρομή: Η έξοδος της πύλης XOR ανατροφοδοτεί μια από τις εισόδους της
 - Επομένως, υπάρχει μια κυκλική διαδρομή που ξεκινάει από τον κόμβο $n4$, διέρχεται από την πύλη XOR και καταλήγει στον κόμβο $n5$, ο οποίος επιστρέφει στον κόμβο $n4$.



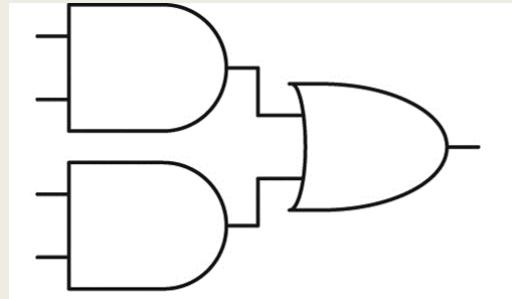
Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (γ)



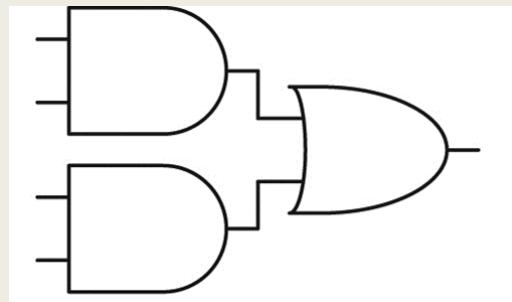
Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (γ) **είναι συνδυαστικό**

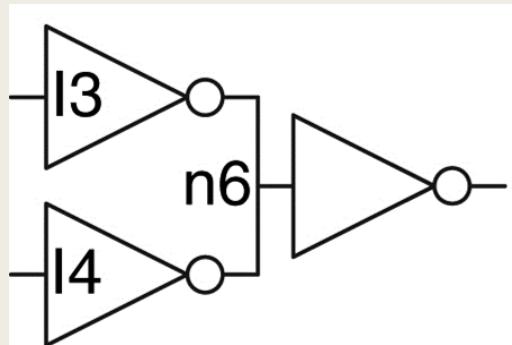


Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (γ) **είναι συνδυαστικό**

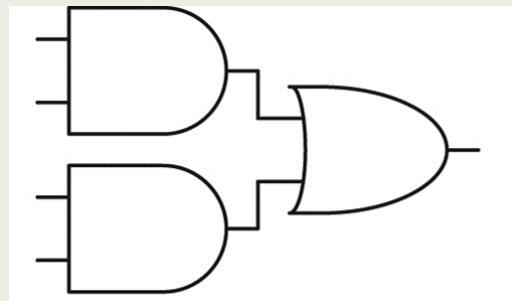


- Το κύκλωμα (δ)



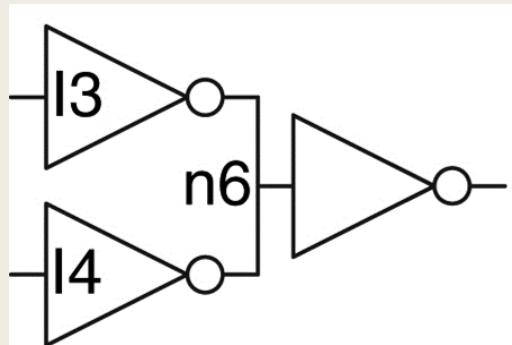
Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (γ) **είναι συνδυαστικό**



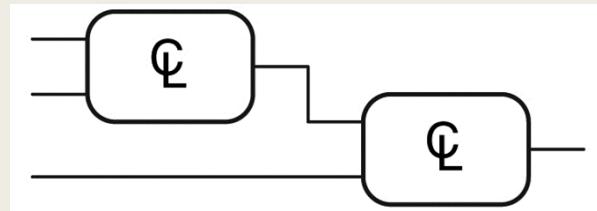
- Το κύκλωμα (δ) **δεν είναι συνδυαστικό**

- Ο κόμβος $n6$ συνδέεται στις εξόδους **και των δύο αντιστροφέων** $I3$ και $I4$



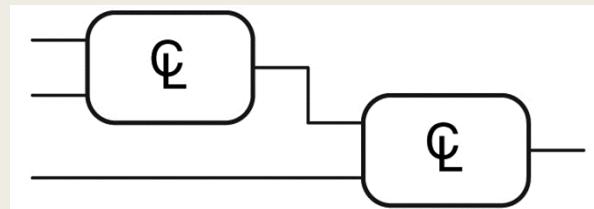
Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (ε)



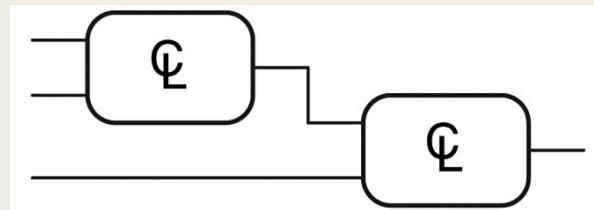
Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (ε) **είναι συνδυαστικό**
 - αποτελεί παράδειγμα του πώς δύο συνδυαστικά κυκλώματα συνδέονται μεταξύ τους για να σχηματίσουν ένα μεγαλύτερο συνδυαστικό κύκλωμα

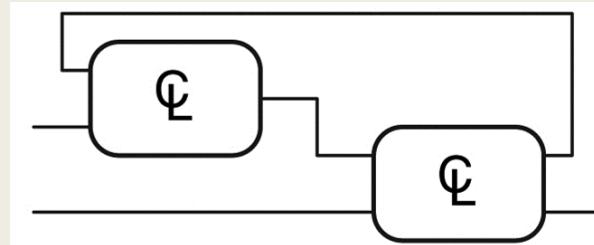


Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (ε) **είναι συνδυαστικό**
 - αποτελεί παράδειγμα του πώς δύο συνδυαστικά κυκλώματα συνδέονται μεταξύ τους για να σχηματίσουν ένα μεγαλύτερο συνδυαστικό κύκλωμα

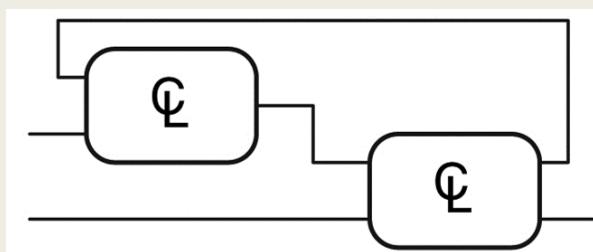
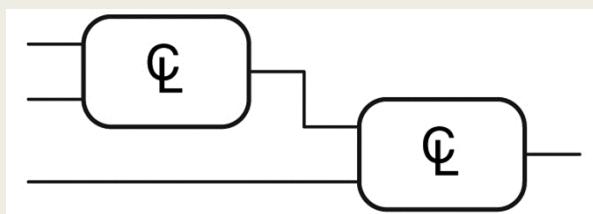


- Το κύκλωμα (σ)



Παράδειγμα 2.1 (συνέχεια)

- Το κύκλωμα (ε) **είναι συνδυαστικό**
 - αποτελεί παράδειγμα του πώς δύο συνδυαστικά κυκλώματα συνδέονται μεταξύ τους για να σχηματίσουν ένα μεγαλύτερο συνδυαστικό κύκλωμα
- Το κύκλωμα (στ) **μπορεί να είναι ή να μην είναι συνδυαστικό**
 - Περιέχει μια κυκλική διαδρομή που διέρχεται από τα δύο στοιχεία
 - Ανάλογα τις συναρτήσεις των δομικών στοιχείων, μπορεί να είναι ή να μην είναι συνδυαστικό κύκλωμα



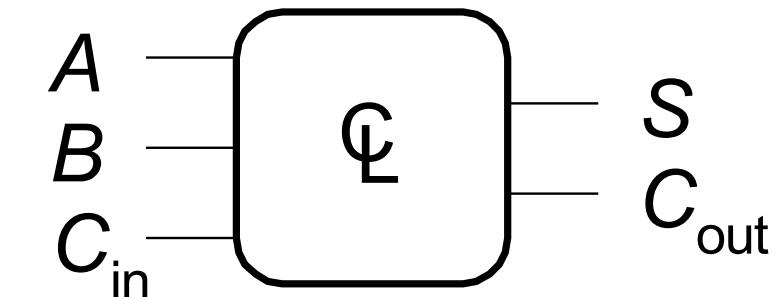
Περίπλοκα κυκλώματα

- Μεγάλα κυκλώματα, όπως οι μικροεπεξεργαστές, ενδέχεται να είναι ιδιαίτερα περίπλοκα
 - σε τέτοιες περιπτώσεις χρησιμοποιούμε τις αρχές διαχείρισης της πολυπλοκότητας
- Η θεώρηση ενός κυκλώματος ως μαύρου κουτιού με καλώς ορισμένη διασύνδεση και λειτουργία αποτελεί εφαρμογή της «**αφαίρεσης**» και της **τμηματικότητας**
- Η κατασκευή του κυκλώματος με χρήση μικρότερων στοιχείων κυκλωμάτων αποτελεί εφαρμογή της **ιεραρχίας**.
- Οι κανόνες της συνδυαστικής σύνθεσης συνιστούν εφαρμογή της **πειθαρχίας**

Εξισώσεις Boole

- Η λειτουργική προδιαγραφή ενός συνδυαστικού κυκλώματος συνήθως εκφράζεται με τη μορφή ενός **πίνακα αληθείας** ή μιας **εξίσωσης Boole**
- Οι εξισώσεις Boole περιέχουν **μεταβλητές** που παίρνουν είτε την τιμή '**1**' (**TRUE**) είτε την τιμή '**0**' (**FALSE**), όρα είναι ιδανικές για την περιγραφή της ψηφιακής λογικής
- Παράδειγμα: **Πλήρης Αθροιστής (Full Adder)**

C_{in}	A	B	C_{out}	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



$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

Ορολογία

- **Συμπλήρωμα** (complement) μιας μεταβλητής A είναι η αντίστροφή της, \bar{A} ($\bar{A} = \text{NOT } A$)
 - $\bar{A}, \bar{B}, \bar{C}$
- **Λεκτικά** (literals) ονομάζονται η μεταβλητή ή το συμπλήρωμά της
 - $A, \bar{A}, B, \bar{B}, C, \bar{C}$
- Αναφερόμαστε στο A ως την **αληθινή μορφή** της μεταβλητής A και στο \bar{A} ως τη **συμπληρωματική μορφή** της μεταβλητής A
 - *Εάν $A = \text{TRUE}$, τότε $\bar{A} = \text{FALSE}$*
 - *Εάν $A = \text{FALSE}$, τότε $\bar{A} = \text{TRUE}$*

Ορολογία

- **Λογικό γινόμενο** (ή απλώς γινόμενο) ή **όρος** (implicant) ονομάζεται το γινόμενο των λεκτικών (η λογική πράξη AND με ένα ή περισσότερα λεκτικά)
 - $AB\bar{C}$, $\bar{A}C$, BC
- **Ελαχιστόρος** (minterm) ονομάζεται το **γινόμενο** που περιλαμβάνει **όλες** τις μεταβλητές της εισόδου
 - $AB\bar{C}$, $A\bar{B}\bar{C}$, ABC
- **Λογικό άθροισμα** (ή απλώς **άθροισμα**) ονομάζεται το άθροισμα των λεκτικών (η λογική πράξη OR με ένα ή περισσότερα λεκτικά)
 - $A+B+\bar{C}$, $\bar{A}+C$, $B+C$
- **Μεγιστόρος** (maxterm) ονομάζεται το **άθροισμα** που περιλαμβάνει **όλες** τις μεταβλητές της εισόδου
 - $(A+\bar{B}+C)$, $(\bar{A}+B+\bar{C})$, $(\bar{A}+\bar{B}+C)$

Προτεραιότητα πράξεων

- Η σειρά των πράξεων **παίζει ρόλο** κατά την ερμηνεία εξισώσεων Boole
 - Στις εξισώσεις Boole, η πράξη **NOT** έχει την υψηλότερη προτεραιότητα
 - Ακολουθεί η πράξη **AND**
 - Τη χαμηλότερη προτεραιότητα έχει η πράξη **OR**
 - Για παράδειγμα
$$Y = A + BC \text{ μεταφράζεται σε } Y = A \text{ OR } (B \text{ AND } C)$$
 - Αλλαγή στην προτεραιότητα επιτυγχάνεται με τη χρήση παρενθέσεων
$$Y = (A + B)C \text{ μεταφράζεται σε } Y = (A \text{ OR } B) \text{ AND } C$$

Πίνακας αλήθειας – Ελαχιστόροι

- Για να βρούμε την εξίσωση Boole για ένα ψηφιακό κύκλωμα πρέπει πρώτα να σχεδίασουμε τον πίνακα αλήθειας του
- Ο πίνακας αληθείας ενός ψηφιακού κυκλώματος με N εισόδους περιέχει 2^N γραμμές, μία για κάθε πιθανό συνδυασμό τιμών των εισόδων του
 - Κάθε γραμμή συσχετίζεται με ένα **δυαδικό αριθμό των N bit**, σε αύξουσα σειρά, που αντιστοιχεί με τιμές των μεταβλητών της εισόδου
 - Κάθε γραμμή συσχετίζεται με έναν **ελαχιστόρο** που είναι '**1**' ή **TRUE** για τις τιμές των μεταβλητών της εισόδου της συγκεκριμένης σειράς
 - Ο ελαχιστόρος ονομάζεται m_i ($i=0, \dots, 2^N-1$), όπου i είναι η δεκαδική τιμή του αντίστοιχου δυαδικού αριθμού των N bit της γραμμής

A B	Ελαχιστόρος	Όνομα
0 0	$\bar{A}\bar{B}$	m_0
0 1	$\bar{A}B$	m_1
1 0	$A\bar{B}$	m_2
1 1	AB	m_3

Κανονική μορφή αθροίσματος γινομένων

- Μπορούμε να βρούμε την εξίσωση Boole από οποιονδήποτε πίνακα αληθείας **αθροίζοντας (πράξη OR) καθέναν από τους ελαχιστόρους (πράξη AND)** για τους οποίους η έξοδος έχει τιμή ‘1’ (TRUE)
- Άρα, η εξίσωση Boole σε κανονική μορφή **αθροίσματος γινομένων** είναι **το άθροισμα (OR) των γινομένων (AND)**
 - Γράφεται και με τον **συμβολισμό σίγμα**, ως άθροισμα ελαχιστώρων

A	B	Y	Ελαχι- στόρος	Ονομασία ελαχιστόρου
0	0	0	$\bar{A} \bar{B}$	m_0
0	1	1	$\bar{A} B$	m_1
1	0	0	$A \bar{B}$	m_2
1	1	1	$A B$	m_3

$$Y = F(A, B) = \bar{A}B + AB = \Sigma(m_1, m_3) = \Sigma(1, 3)$$

Πίνακας αλήθειας – Μεγιστόροι

- Για να βρούμε την εξίσωση Boole για ένα ψηφιακό κύκλωμα πρέπει πρώτα να σχεδίασουμε τον πίνακα αληθείας του
- Ο πίνακας αληθείας ενός ψηφιακού κυκλώματος με N εισόδους περιέχει 2^N γραμμές, μία για κάθε πιθανό συνδυασμό τιμών των εισόδων του
 - Κάθε γραμμή συσχετίζεται με ένα **δυαδικό αριθμό των N bit**, σε αύξουσα σειρά, που αντιστοιχεί με τιμές των μεταβλητών της εισόδου
 - Κάθε γραμμή συσχετίζεται με έναν **μεγιστόρο** που είναι '**Ο**' ή **FALSE** για τις τιμές των μεταβλητών της εισόδου της συγκεκριμένης σειράς
 - Ο μεγιστόρος ονομάζεται M_i ($i=0.., 2^N-1$), όπου i είναι η δεκαδική τιμή του αντίστοιχου δυαδικού αριθμού των N bit της γραμμής

A B	Μεγιστόρος	Όνομα
0 0	$A + B$	M_0
0 1	$A + \bar{B}$	M_1
1 0	$\bar{A} + B$	M_2
1 1	$\bar{A} + \bar{B}$	M_3

Κανονική μορφή γινομένου αθροισμάτων

- Μπορούμε να βρούμε την εξίσωση Boole από οποιονδήποτε πίνακα αληθείας **πολλαπλασιάζοντας (πράξη AND) καθέναν από τους μεγιστόρους (πράξη OR)** για τους οποίους η έξοδος έχει τιμή 'Ο' (FALSE)
- Άρα, η εξίσωση Boole σε κανονική μορφή γινομένου αθροισμάτων είναι **το γινόμενο (AND) των αθροισμάτων (OR)**
 - Γράφεται και με τον **συμβολισμό πι**, ως γινόμενο μεγιστόρων

A	B	Y	Μεγι- στόρος	Όνομασία μεγιστόρου
0	0	0	$A + B$	M_0
0	1	1	$A + \bar{B}$	M_1
1	0	0	$\bar{A} + B$	M_2
1	1	1	$\bar{A} + \bar{B}$	M_3

$$Y = F(A, B) = (A + B)(\bar{A} + B) = \Pi(M_0, M_2) = \Pi(0, 2)$$

Παράδειγμα 2.3

- Ο Γιάννης έχει αποφασίσει να πάει για πικνίκ.
- Σκέφτεται ότι **δεν πρόκειται να περάσει καλά**
 - αν βρέξει ($R = 1$) ή
 - αν υπάρχουν μυρμήγκια ($A = 1$)
- Σχεδιάστε ένα κύκλωμα που θα παράγει ως έξοδο E την τιμή TRUE ($E=1$) **μόνο αν ο Γιάννης απολαμβάνει το πικνίκ του**

A	R	E
0	0	
0	1	
1	0	
1	1	

Παράδειγμα 2.3

- Ο Γιάννης έχει αποφασίσει να πάει για πικνίκ.
- Σκέφτεται ότι **δεν πρόκειται να περάσει καλά**
 - αν βρέξει ($R = 1$) ή
 - αν υπάρχουν μυρμήγκια ($A = 1$)
- Σχεδιάστε ένα κύκλωμα που θα παράγει ως έξοδο E την τιμή TRUE ($E=1$) **μόνο αν ο Γιάννης απολαμβάνει το πικνίκ του**

A	R	E
0	0	1
0	1	0
1	0	0
1	1	0

Παράδειγμα 2.3

- Κανονική μορφή αθροίσματος γινομένου

A	R	E	Ελαχιστόρος
0	0	1	
0	1	0	
1	0	0	
1	1	0	

- Κανονική μορφή γινομένου αθροισμάτων

A	R	E	Μεγιστόρος
0	0	1	
0	1	0	
1	0	0	
1	1	0	

Παράδειγμα 2.3

- Κανονική μορφή αθροίσματος γινομένου

A	R	E	Ελαχιστόρος
0	0	1	$\bar{A} \bar{R}$
0	1	0	$\bar{A} R$
1	0	0	$A \bar{R}$
1	1	0	$A R$

- Κανονική μορφή γινομένου αθροισμάτων

A	R	E	Μεγιστόρος
0	0	1	
0	1	0	
1	0	0	
1	1	0	

Παράδειγμα 2.3

- Κανονική μορφή αθροίσματος γινομένου

A	R	E	Ελαχιστόρος	$E = \bar{A} \bar{R} = \Sigma(0)$
0	0	1	$\bar{A} \bar{R}$	
0	1	0	$\bar{A} R$	
1	0	0	$A \bar{R}$	
1	1	0	$A R$	

- Κανονική μορφή γινομένου αθροισμάτων

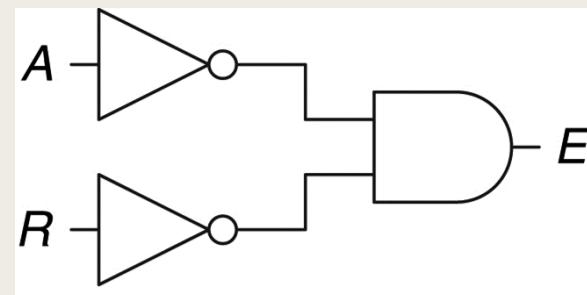
A	R	E	Μεγιστόρος
0	0	1	
0	1	0	
1	0	0	
1	1	0	

Παράδειγμα 2.3

- Κανονική μορφή αθροίσματος γινομένου

A	R	E	Ελαχιστόρος
0	0	1	$\bar{A} \bar{R}$
0	1	0	$\bar{A} R$
1	0	0	$A \bar{R}$
1	1	0	$A R$

$$E = \bar{A} \bar{R} = \Sigma(0)$$



- Κανονική μορφή γινομένου αθροισμάτων

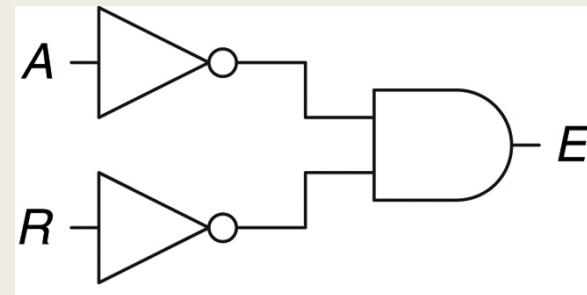
A	R	E	Μεγιστόρος
0	0	1	
0	1	0	
1	0	0	
1	1	0	

Παράδειγμα 2.3

- Κανονική μορφή αθροίσματος γινομένου

A	R	E	Ελαχιστόρος
0	0	1	$\bar{A} \bar{R}$
0	1	0	$\bar{A} R$
1	0	0	$A \bar{R}$
1	1	0	$A R$

$$E = \bar{A} \bar{R} = \Sigma(0)$$



- Κανονική μορφή γινομένου αθροισμάτων

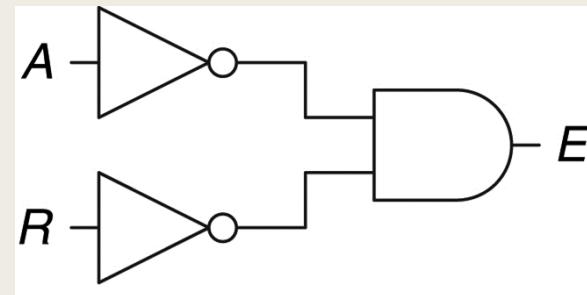
A	R	E	Μεγιστόρος
0	0	1	$A + R$
0	1	0	$A + \bar{R}$
1	0	0	$\bar{A} + R$
1	1	0	$\bar{A} + \bar{R}$

Παράδειγμα 2.3

- Κανονική μορφή αθροίσματος γινομένου

A	R	E	Ελαχιστόρος
0	0	1	$\bar{A} \bar{R}$
0	1	0	$\bar{A} R$
1	0	0	$A \bar{R}$
1	1	0	$A R$

$$E = \bar{A} \bar{R} = \Sigma(0)$$



- Κανονική μορφή γινομένου αθροισμάτων

A	R	E	Μεγιστόρος
0	0	1	$A + R$
0	1	0	$A + \bar{R}$
1	0	0	$\bar{A} + R$
1	1	0	$\bar{A} + \bar{R}$

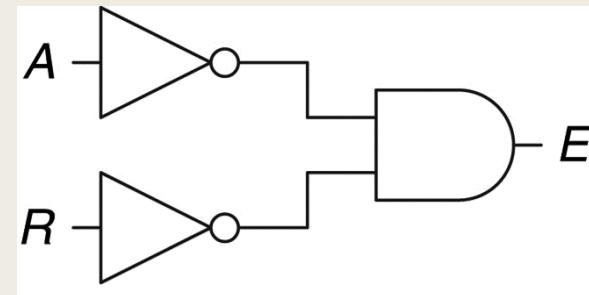
$$\begin{aligned} E &= (A + \bar{R})(\bar{A} + R)(\bar{A} + \bar{R}) \\ &= \Pi(1,2,3) = \overline{A + R} \end{aligned}$$

Παράδειγμα 2.3

- Κανονική μορφή αθροίσματος γινομένου

A	R	E	Ελαχιστόρος
0	0	1	$\bar{A} \bar{R}$
0	1	0	$\bar{A} R$
1	0	0	$A \bar{R}$
1	1	0	$A R$

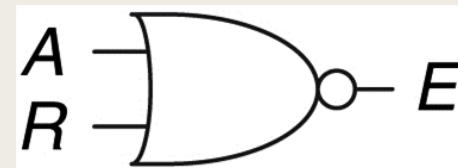
$$E = \bar{A} \bar{R} = \Sigma(0)$$



- Κανονική μορφή γινομένου αθροισμάτων

A	R	E	Μεγιστόρος
0	0	1	$A + R$
0	1	0	$A + \bar{R}$
1	0	0	$\bar{A} + R$
1	1	0	$\bar{A} + \bar{R}$

$$\begin{aligned} E &= (A + \bar{R})(\bar{A} + R)(\bar{A} + \bar{R}) \\ &= \Pi(1,2,3) = \overline{A + R} \end{aligned}$$



Επιλεγμένες ασκήσεις

■ Ασκηση 2.33

Ο Γιάννης Μπιτάκης απολαμβάνει το πικνίκ του όταν η μέρα είναι ηλιόλουστη και δεν υπάρχουν μυρμήγκια. Το απολαμβάνει επίσης όταν βλέπει ένα κολιμπρί την ημέρα του πικνίκ, καθώς επίσης όταν, την ημέρα του πικνίκ, υπάρχουν μυρμήγκια και πασχαλίτσες.

- Γράψτε μια εξίσωση Boolean για την απόλαυσή του (*enjoyment, E*) σε σχέση με:
 - τον ήλιο (sun, S),
 - τα μυρμήγκια (ants, A),
 - τα κολιμπρί (hummingbirds, H), και
 - τις πασχαλίτσες (ladybugs, L)

$$E = \bar{S} \bar{A} + AL + H$$

Επιλεγμένες ασκήσεις

■ Άσκηση 2.2

Γράψτε μια εξίσωση Boole σε κανονική μορφή αθροίσματος γινομένων για τον πίνακα αληθείας (β) της Εικόνας 2.81

- Χρησιμοποιήστε τους ελαχιστόρους

■ Άσκηση 2.4

Γράψτε μια εξίσωση Boole σε κανονική μορφή γινομένου αθροισμάτων για τον πίνακα αληθείας (β) της Εικόνας 2.81

- Χρησιμοποιήστε τους μεγιστόρους

(b)

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

$$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C}$$

$$Y = (A + B + C)(\bar{A} + B + \bar{C})(\bar{A} + \bar{B} + \bar{C})$$

Άλγεβρα Boole

- Η κανονική μορφή αθροίσματος γινομένων ή γινομένου αθροισμάτων συνήθως **δεν δίνει την απλούστερη εξίσωση**
- Χρησιμοποιώντας την άλγεβρα Boole μπορούμε να **ελαχιστοποιήσουμε** τις εξισώσεις Boole
- Η άλγεβρα Boole βασίζεται σε ένα σύνολο αξιωμάτων και θεωρημάτων
 - *Τα αξιώματα είναι αναπόδεικτα με την έννοια ότι ένας ορισμός δεν μπορεί να αποδειχθεί*
 - *Τα θεωρήματα αποδεικνύονται με βάση τα αξιώματα*
- Τα αξιώματα και τα θεωρήματα της άλγεβρας Boole ακολουθούν την **αρχή του δυϊσμού** (δεν ισχύει στη συνήθη άλγεβρα)
 - *Αν, σε μία λογική έκφραση, τα σύμβολα 0 και 1 και οι τελεστές • (AND) και + (OR) εναλλαχθούν, η λογική έκφραση θα εξακολουθεί να είναι ορθή*

Αξιώματα άλγεβρας Boole

	Αξίωμα	Ονομασία
A1	$B = 0 \text{ αν } B \neq 1$	Δυαδικό πεδίο
A2	$0 = \bar{1}$	NOT
A3	$0 \bullet 0 = 0$	AND
A4	$1 \bullet 1 = 1$	AND
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	AND

Αξιώματα áλγεβρας Boole

	Αξίωμα	Ονομασία
A1	$B = 0 \text{ αν } B \neq 1$	Δυαδικό πεδίο
A2	$0 = \bar{1}$	NOT
A3	$0 \bullet 0 = 0$	AND
A4	$1 \bullet 1 = 1$	AND
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	AND

Δυϊσμός: Αντιστρέψτε: • με +
0 με 1

Αξιώματα άλγεβρας Boole

	Αξίωμα	Δυϊκό αξίωμα	Ονομασία
A1	$B = 0 \text{ αν } B \neq 1$	$B = 1 \text{ αν } B \neq 0$	Δυαδικό πεδίο
A2	$0 = \bar{1}$	$1 = \bar{0}$	NOT
A3	$0 \bullet 0 = 0$	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	$1 + 0 = 0 + 1 = 1$	AND/OR

Δυϊσμός: Αντιστρέψτε: • με +
0 με 1

Αξιώματα áλγεβρας Boole

	Αξίωμα	Δυϊκό αξίωμα	Ονομασία
A1	$B = 0 \text{ αν } B \neq 1$	$B = 1 \text{ αν } B \neq 0$	Δυαδικό πεδίο

- Το αξίωμα A1 ορίζει ότι μια μεταβλητή Boole B έχει τιμή 0 αν δεν έχει τιμή 1
- Το δυϊκό αξίωμα, ορίζει ότι η μεταβλητή έχει τιμή 1 αν δεν έχει τιμή 0
- Και τα δύο μαζί, μας λένε ότι εργαζόμαστε σε ένα πεδίο Boole ή **δυαδικό πεδίο με μηδενικά (0) και άσους (1)**

Αξιώματα áλγεβρας Boole

	Αξίωμα	Δυϊκό αξίωμα	Ονομασία
A2	$0 = \bar{1}$	$1 = \bar{0}$	NOT

- Τα αξιώματα A2 και A2' ορίζουν την **πράξη NOT**

	Αξίωμα	Δυϊκό αξίωμα	Ονομασία
A3	$0 \bullet 0 = 0$	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	$1 + 0 = 0 + 1 = 1$	AND/OR

- Τα αξιώματα A3 έως A5 ορίζουν την **πράξη AND**
- Τα δυϊκά τους, ορίζουν την **πράξη OR**

Θεωρήματα μίας μεταβλητής

	Θεώρημα	Ονομασία
T1	$B \bullet 1 = B$	Ουδέτερο στοιχείο
T2	$B \bullet 0 = 0$	Κυρίαρχο στοιχείο
T3	$B \bullet B = B$	Ταυτοδυναμία
T4	$\bar{\bar{B}} = B$	Διπλό συμπλήρωμα
T5	$B \bullet \bar{B} = 0$	Συμπληρώματα

Θεωρήματα μίας μεταβλητής

	Θεώρημα	Ονομασία
T1	$B \bullet 1 = B$	Ουδέτερο στοιχείο
T2	$B \bullet 0 = 0$	Κυρίαρχο στοιχείο
T3	$B \bullet B = B$	Ταυτοδυναμία
T4	$\bar{\bar{B}} = B$	Διπλό συμπλήρωμα
T5	$B \bullet \bar{B} = 0$	Συμπληρώματα

Δυϊσμός: Αντιστρέψτε: • με +
0 με 1

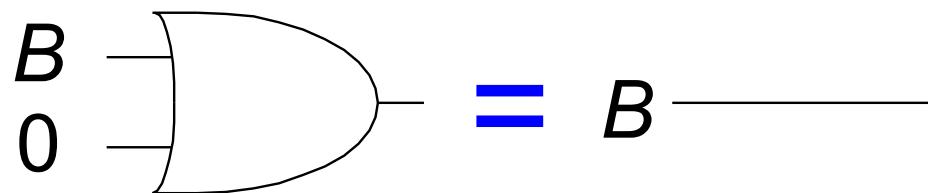
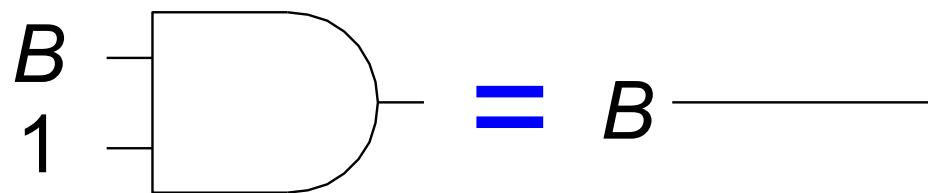
Θεωρήματα μίας μεταβλητής

	Θεώρημα	Διϊκό Θεώρημα	Ονομασία
T1	$B \bullet 1 = B$	$B + 0 = B$	Ουδέτερο στοιχείο
T2	$B \bullet 0 = 0$	$B + 1 = 1$	Κυρίαρχο στοιχείο
T3	$B \bullet B = B$	$B + B = B$	Ταυτοδυναμία
T4		$\bar{\bar{B}} = B$	Διπλό συμπλήρωμα
T5	$B \bullet \bar{B} = 0$	$B + \bar{B} = 1$	Συμπληρώματα

Δυϊσμός: Αντιστρέψτε: • με +
0 με 1

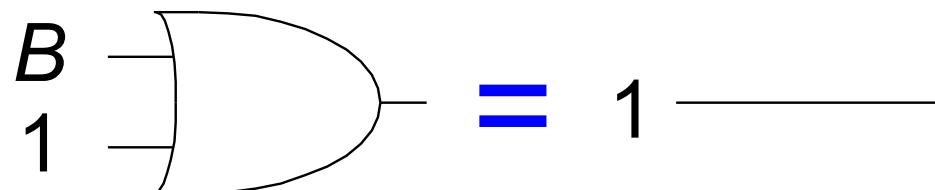
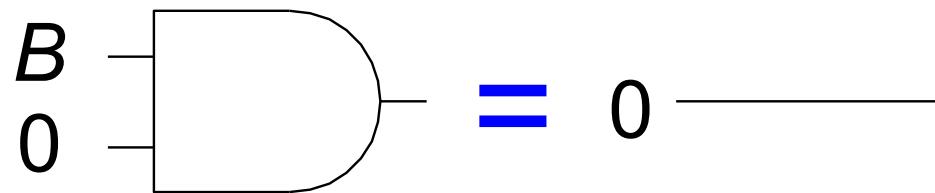
T1: Ουδέτερο στοιχείο

- $B \bullet 1 = B$ (Το 1 είναι ουδέτερο στοιχείο στην πράξη AND)
- $B + 0 = B$ (Το 0 είναι ουδέτερο στοιχείο στην πράξη OR)



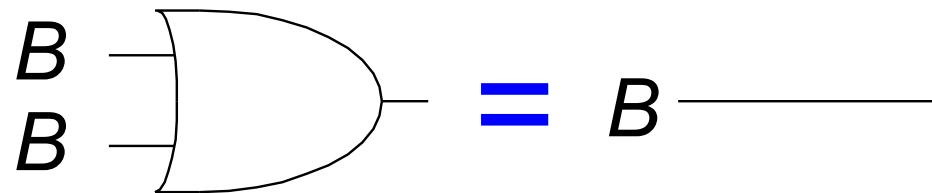
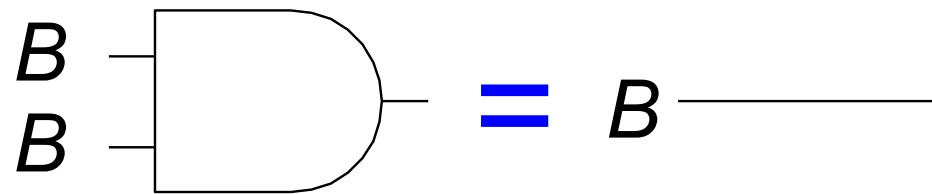
T2: Κυρίαρχο στοιχείο

- $B \bullet 0 = 0$ (Το 0 είναι κυρίαρχο στοιχείο στην πράξη AND)
- $B + 1 = 1$ (Το 1 είναι κυρίαρχο στοιχείο στην πράξη OR)
 - (δεν ισχύει στη συνήθη άλγεβρα)



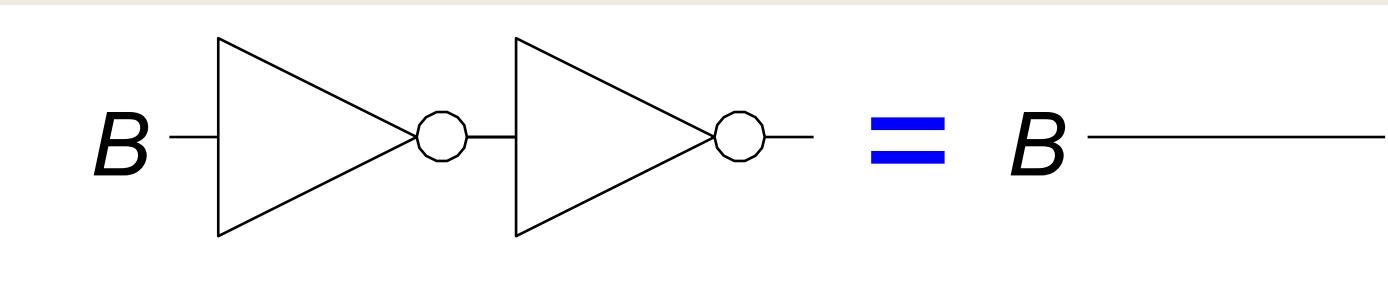
Τ3: Ταυτοδυναμία (Idempotency)

- $B \bullet B = B$ (δεν ισχύει στη συνήθη άλγεβρα)
- $B + B = B$ (δεν ισχύει στη συνήθη άλγεβρα)



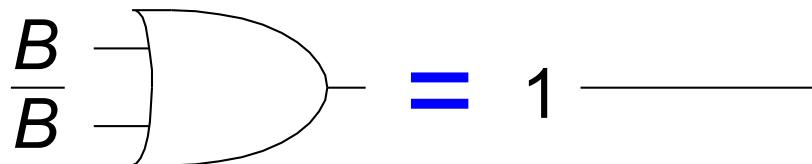
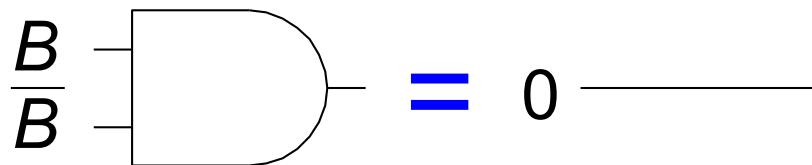
T4: Διπλό συμπλήρωμα

- $\bar{\bar{B}} = B$ (δεν ισχύει στη συνήθη άλγεβρα)



T5: Θεώρημα των συμπληρωμάτων (Complement Theorem)

- Το **Θεώρημα των συμπληρωμάτων** (Complement Theorem), ορίζει ότι
 - η πράξη AND μιας μεταβλητής με το συμπλήρωμά της δίνει αποτέλεσμα 0 ($B \cdot \bar{B} = 0$)
 - η πράξη OR μιας μεταβλητής με το συμπλήρωμά της δίνει αποτέλεσμα 1 ($B + \bar{B} = 1$)



Θεωρήματα μίας μεταβλητής

	Θεώρημα	Διϊκό Θεώρημα	Ονομασία
T1	$B \bullet 1 = B$	$B + 0 = B$	Ουδέτερο στοιχείο
T2	$B \bullet 0 = 0$	$B + 1 = 1$	Κυρίαρχο στοιχείο
T3	$B \bullet B = B$	$B + B = B$	Ταυτοδυναμία
T4		$\bar{\bar{B}} = B$	Διπλό συμπλήρωμα
T5	$B \bullet \bar{B} = 0$	$B + \bar{B} = 1$	Συμπληρώματα

Δυϊσμός: Αντιστρέψτε: • με +
0 με 1

Θεωρήματα πολλών μεταβλητών

	Θεώρημα	Δυϊκό Θεώρημα	Ονομασία
T6	$B \bullet C = C \bullet B$	$B + C = C + B$	Αντιμεταθετικότητα
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Προσεταιριστικότητα
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Επιμεριστικότητα
T9	$B \bullet (B + C) = B$	$B + (B \bullet C) = B$	Κάλυψη
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	$(B + C) \bullet (B + \bar{C}) = B$	Συνδυασμός
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = B \bullet C + \bar{B} \bullet D$	$(B + C) \bullet (\bar{B} + D) \bullet (C + D) = (B + C) \bullet (B + D)$	Ομοφωνία

Δυϊσμός:

Αντιστρέψτε:

$$\begin{aligned} \bullet &\text{ με } + \\ 0 &\text{ με } 1 \end{aligned}$$

Θεωρήματα πολλών μεταβλητών

	Θεώρημα	Ονομασία
T6	$B \bullet C = C \bullet B$	Αντιμεταθετικότητα
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	Προσεταιριστικότητα
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	Επιμεριστικότητα
T9	$B \bullet (B + C) = B$	Κάλυψη
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	Συνδυασμός
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = B \bullet C + \bar{B} \bullet D$	Ομοφωνία

Πώς αποδεικνύονται?

Μέθοδοι απόδειξης Θεωρημάτων Boole

- **Μέθοδος 1:** Τέλεια επαγωγή (perfect induction)
η οποία στηρίζεται στον **πίνακα αλήθειας**
 - Να δειχτεί δηλαδή ότι το θεώρημα ισχύει για όλες τις **πιθανές τιμές** των μεταβλητών
- **Μέθοδος 2:** Χρησιμοποιώντας **άλλα Θεωρήματα** ή **αξιώματα** με σκοπό να απλοποιήσουμε την εξίσωση
 - Δηλαδή μορφοποιούμε το ένα μέλος της εξίσωσης έτσι ώστε να είναι **πανομοιότυπο με το άλλο**

Απόδειξη με τέλεια επαγωγή

	Θεώρημα	Ονομασία
T6	$B \bullet C = C \bullet B$	Αντιμεταθετικότητα

B	C	BC	CB
0	0		
0	1		
1	0		
1	1		

Απόδειξη με τέλεια επαγωγή

	Θεώρημα	Ονομασία
T6	$B \bullet C = C \bullet B$	Αντιμεταθετικότητα

B	C	BC	CB
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Θεωρήματα πολλών μεταβλητών

	Θεώρημα	Ονομασία
T6	$B \bullet C = C \bullet B$	Αντιμεταθετικότητα
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	Προσεταιριστικότητα
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	Επιμεριστικότητα
T9	$B \bullet (B + C) = B$	Κάλυψη
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	Συνδυασμός
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = B \bullet C + \bar{B} \bullet D$	Ομοφωνία

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Να αποδειχθεί με:

- **Μέθοδος 1:** Τέλεια επαγωγή
- **Μέθοδος 2:** Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 1: Τέλεια επαγωγή

B	C	$(B + C)$	$B(B + C)$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 1: Τέλεια επαγωγή

B	C	(B + C)	B(B + C)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 1: Τέλεια επαγωγή

B	C	$(B + C)$	$B(B + C)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 1: Τέλεια επαγωγή

B	C	(B + C)	B(B + C)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$B \bullet (B+C) =$$

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$B \bullet (B+C) = B \bullet B + B \bullet C \quad T8: \text{Επιμεριστικότητα}$$

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$\begin{aligned} B \bullet (B+C) &= B \bullet B + B \bullet C && T8: \text{Επιμεριστικότητα} \\ &= B + B \bullet C && T3: \text{Ταυτοδυναμία} \end{aligned}$$

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$\begin{aligned} B \bullet (B+C) &= B \bullet B + B \bullet C && T8: \text{Επιμεριστικότητα} \\ &= B + B \bullet C && T3: \text{Ταυτοδυναμία} \\ &= B \bullet 1 + B \bullet C && T1: \text{Ουδέτερο στοιχείο} \end{aligned}$$

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$\begin{aligned} B \bullet (B+C) &= B \bullet B + B \bullet C && T8: \text{Επιμεριστικότητα} \\ &= B + B \bullet C && T3: \text{Ταυτοδυναμία} \\ &= B \bullet 1 + B \bullet C && T1: \text{Ουδέτερο στοιχείο} \\ &= B \bullet (1 + C) && T8: \text{Επιμεριστικότητα} \end{aligned}$$

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$\begin{aligned} B \bullet (B+C) &= B \bullet B + B \bullet C && T8: \text{Επιμεριστικότητα} \\ &= B + B \bullet C && T3: \text{Ταυτοδυναμία} \\ &= B \bullet 1 + B \bullet C && T1: \text{Ουδέτερο στοιχείο} \\ &= B \bullet (1 + C) && T8: \text{Επιμεριστικότητα} \\ &= B \bullet (1) && T2: \text{Κυρίαρχο στοιχείο} \end{aligned}$$

T9: Κάλυψη

	Θεώρημα	Ονομασία
T9	$B \bullet (B + C) = B$	Κάλυψη

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$\begin{aligned} B \bullet (B+C) &= B \bullet B + B \bullet C && T8: \text{Επιμεριστικότητα} \\ &= B + B \bullet C && T3: \text{Ταυτοδυναμία} \\ &= B \bullet 1 + B \bullet C && T1: \text{Ουδέτερο στοιχείο} \\ &= B \bullet (1 + C) && T8: \text{Επιμεριστικότητα} \\ &= B \bullet (1) && T2: \text{Κυρίαρχο στοιχείο} \\ &= B && T1: \text{Ουδέτερο στοιχείο} \end{aligned}$$

T10: Συνδυασμός

	Θεώρημα	Ονομασία
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	Συνδυασμός

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$(B \bullet C) + (B \bullet \bar{C}) =$$

T10: Συνδυασμός

	Θεώρημα	Ονομασία
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	Συνδυασμός

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$(B \bullet C) + (B \bullet \bar{C}) = B \bullet (C + \bar{C}) \quad T8: \text{Επιμεριστικότητα}$$

T10: Συνδυασμός

	Θεώρημα	Ονομασία
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	Συνδυασμός

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$\begin{aligned}(B \bullet C) + (B \bullet \bar{C}) &= B \bullet (C + \bar{C}) \\ &= B \bullet (1)\end{aligned}$$

T8: Επιμεριστικότητα

T5': Συμπλήρωμα

T10: Συνδυασμός

	Θεώρημα	Ονομασία
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	Συνδυασμός

Μέθοδος 2: Χρησιμοποιώντας άλλα θεωρήματα ή αξιώματα

$$\begin{aligned}(B \bullet C) + (B \bullet \bar{C}) &= B \bullet (C + \bar{C}) \\ &= B \bullet (1) \\ &= B\end{aligned}$$

T8: Επιμεριστικότητα

T5': Συμπλήρωμα

T1: Ουδέτερο στοιχείο

T11: Ομοφωνία

	Θεώρημα	Ονομασία
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = \\ B \bullet C + \bar{B} \bullet D$	Ομοφωνία

Αποδείξτε το χρησιμοποιώντας (1) την τέλεια επαγωγή και (2) άλλα θεωρήματα ή αξιώματα

T11: Ομοφωνία

	Θεώρημα	Ονομασία
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = \\ B \bullet C + \bar{B} \bullet D$	Ομοφωνία

Μέθοδος 1: τέλεια επαγωγή (Παράδειγμα 2.5)

B	C	D	B • C	\bar{B} • D	C • D	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D)$	$B \bullet C + \bar{B} \bullet D$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

T11: Ομοφωνία

	Θεώρημα	Ονομασία
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = \\ B \bullet C + \bar{B} \bullet D$	Ομοφωνία

Μέθοδος 1: τέλεια επαγωγή (Παράδειγμα 2.5)

B	C	D	$B \bullet C$	$\bar{B} \bullet D$	$C \bullet D$	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D)$	$B \bullet C + \bar{B} \bullet D$
0	0	0	0	0	0	0	
0	0	1	0	1	0	1	
0	1	0	0	0	0	0	
0	1	1	0	1	1	1	
1	0	0	0	0	0	0	
1	0	1	0	0	0	0	
1	1	0	1	0	0	1	
1	1	1	1	0	1	1	

T11: Ομοφωνία

	Θεώρημα	Ονομασία
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = \\ B \bullet C + \bar{B} \bullet D$	Ομοφωνία

Μέθοδος 1: τέλεια επαγωγή (Παράδειγμα 2.5)

B	C	D	$B \bullet C$	$\bar{B} \bullet D$	$C \bullet D$	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D)$	$B \bullet C + \bar{B} \bullet D$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

T11: Ομοφωνία

	Θεώρημα	Ονομασία
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = \\ B \bullet C + \bar{B} \bullet D$	Ομοφωνία

Μέθοδος 1: τέλεια επαγωγή (Παράδειγμα 2.5)

B	C	D	$B \bullet C$	$\bar{B} \bullet D$	$C \bullet D$	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D)$	$B \bullet C + \bar{B} \bullet D$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

T11: Ομοφωνία

	Θεώρημα	Ονομασία
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = \\ B \bullet C + \bar{B} \bullet D$	Ομοφωνία

Μέθοδος 2: άλλα θεωρήματα ή αξιώματα

$$B \bullet C + \bar{B} \bullet D + C \bullet D$$

$$= B \bullet C + \bar{B} \bullet D + C \bullet D \bullet 1$$

T1: Ουδέτερο στοιχείο

$$= B \bullet C + \bar{B} \bullet D + C \bullet D \bullet (B + \bar{B})$$

T5': Συμπληρώματα

$$= B \bullet C + \bar{B} \bullet D + C \bullet D \bullet B + C \bullet D \bullet \bar{B}$$

T8: Επιμεριστικότητα

$$= B \bullet C + \bar{B} \bullet D + B \bullet C \bullet D + \bar{B} \bullet C \bullet D$$

T6: Αντιμεταθετικότητα

$$= B \bullet C + \bar{B} \bullet D$$

T9': Κάλυψη

Εφαρμογή των Θεώρημάτων

	Θεώρημα	Δυϊκό Θεώρημα	Ονομασία
T6	$B \bullet C = C \bullet B$	$B + C = C + B$	Αντιμεταθετικότητα
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Προσεταιριστικότητα
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Επιμεριστικότητα
T9	$B \bullet (B + C) = B$	$B + (B \bullet C) = B$	Κάλυψη
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	$(B + C) \bullet (B + \bar{C}) = B$	Συνδυασμός
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = B \bullet C + \bar{B} \bullet D$	$(B + C) \bullet (\bar{B} + D) \bullet (C + D) = (B + C) \bullet (B + D)$	Ομοφωνία

Δυϊσμός:

Αντιστρέψτε:

$$\begin{aligned} \bullet &\text{ με } + \\ 0 &\text{ με } 1 \end{aligned}$$

Ανεξαρτησία στη σειρά των εισόδων

	Θεώρημα	Δυϊκό Θεώρημα	Ονομασία
T6	$B \bullet C = C \bullet B$	$B + C = C + B$	Αντιμεταθετικότητα
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Προσεταιριστικότητα

- Λόγω της αντιμεταθετικότητας, η σειρά των εισόδων σε μια πύλη AND ή OR δεν επηρεάζει την τιμή της εξόδου της.
- Λόγω της προσεταιριστικότητας, οι εκάστοτε ομαδοποιήσεις των εισόδων δεν επηρεάζουν την τιμή της εξόδου
 - *Παράδειγμα:* Σε ένα δένδρο από πύλες AND ή πύλες OR δύο εισόδων για υλοποιήσεις πράξεων πολλών bit (π.χ. 8 bit) δεν έχει σημασία η σειρά των εισόδων στις πύλες δύο εισόδων
 - Στα 8 bit, το δένδρο θα έχει 7 πύλες κατανεμημένες σε 3 επίπεδα: στο 1^o επίπεδο είναι 4 πύλες, στο 2^o επίπεδο 2 πύλες και στο 3^o επίπεδο 1 πύλη

Ελαχιστοποίηση εξισώσεων Boole

	Θεώρημα	Δυϊκό Θεώρημα	Ονομασία
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Επιμεριστικότητα

- Ελαχιστοποίηση (ή απλοποίηση) εξισώσεων Boole σημαίνει η εύρεση εκείνης της εξίσωσης που έχει:
 - *Τους λιγότερους όρους*
 - *Κάθε όρος έχει τα λιγότερα λεκτικά (είναι πρώτος όρος)*
- Η ελαχιστοποίηση των εξισώσεων Boole βασίζεται κυρίως στα θεωρήματα της επιμεριστικότητας, των συμπληρωμάτων, του ουδέτερου στοιχείου και του κυρίαρχου στοιχείου
 - Οι εξισώσεις Boole σε μορφή αθροίσματος γινομένων ελαχιστοποιούνται με το T8 (που ισχύει στη συνήθη άλγεβρα)
 - Οι εξισώσεις Boole σε μορφή γινομένου αθροισμάτων ελαχιστοποιούνται με το T8' (που δεν ισχύει στη συνήθη άλγεβρα)
 - Έχουμε μία δυσκολία στην εφαρμογή του T8' και για αυτό το λόγο συνήθως χρησιμοποιούμε τα **αθροίσματα γινομένων**, αντί των γινομένων αθροισμάτων

Ελαχιστοποίηση εξισώσεων Boole

	Θεώρημα	Δυϊκό Θεώρημα	Ονομασία
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Επιμεριστικότητα
T9	$B \bullet (B + C) = B$	$B + (B \bullet C) = B$	Κάλυψη
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	$(B + C) \bullet (B + \bar{C}) = B$	Συνδυασμός
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = B \bullet C + \bar{B} \bullet D$	$(B + C) \bullet (\bar{B} + D) \bullet (C + D) = (B + C) \bullet (B + D)$	Ομοφωνία

- Τα θεωρήματα της **κάλυψης** (T9), του **συνδυασμού** (T10), και της **ομοφωνίας** (T11), μας επιτρέπουν να **εξαλείφουμε πλεονάζουσες μεταβλητές**
 - *Βασίζονται κυρίως στα θεωρήματα της επιμεριστικότητας, των συμπληρωμάτων, του ουδέτερου στοιχείου και του κυρίαρχου στοιχείου*

Ελαχιστοποίηση εξισώσεων Boole

- Αναφερόμαστε σε εξισώσεις Boole που είναι σε **μορφή αθροίσματος γινομένων**
 - Έστω P οποιοδήποτε γινόμενο λεκτικών
 - Βασίζομαστε κυρίως στα θεωρήματα της επιμεριστικότητας, των συμπληρωμάτων, του ουδέτερου στοιχείου και του κυρίαρχου στοιχείου
 - Χρησιμοποιούμε και τα ακόλουθα θεωρήματα:
 - Γενίκευση του θεωρήματος της ταυτοδυναμίας T3':
 - $P = P + P$
 - Γενίκευση του θεωρήματος της κάλυψης T9':
 - $P \cdot A + A = P \cdot A + A \cdot 1 = A \cdot (P + 1) = A \cdot 1 = A$
 - Γενίκευση του θεωρήματος του συνδυασμού T10:
 - $P \cdot A + P \cdot \bar{A} = P \cdot (A + \bar{A}) = P \cdot 1 = P$
 - Χρήση του θεωρήματος της επιμεριστικότητας T8':
 - $P \cdot A + \bar{A} = (P + \bar{A}) \cdot (A + \bar{A}) = (P + \bar{A}) \cdot 1 = P + \bar{A}$
 - $P \cdot \bar{A} + A = (P + A) \cdot (A + \bar{A}) = (P + A) \cdot 1 = P + A$

Παράδειγμα 2.6

- Να ελαχιστοποιήσετε την εξίσωση Boole: $\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$

■ Λύση 1 (ημιτελής)

$$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$$

$$\bar{B}\bar{C}(\bar{A} + A) + A\bar{B}\bar{C}$$

$$\bar{B}\bar{C}(1) + A\bar{B}\bar{C}$$

$$\bar{B}\bar{C} + A\bar{B}\bar{C}$$

Λύση 2 (πλήρης)

$$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \textcolor{red}{A\bar{B}\bar{C}} + A\bar{B}C \quad (\text{T3}')$$

$$\bar{B}\bar{C}(\bar{A} + A) + A\bar{B}(\bar{C} + C) \quad (\text{T8})$$

$$\bar{B}\bar{C}(1) + A\bar{B}(1) \quad (\text{T5}')$$

$$\bar{B}\bar{C} + A\bar{B} \quad (\text{T1})$$

- Η χρήση ενός **κοινού ελαχιστόρου** σύμφωνα με το θεώρημα της ταυτοδυναμίας (όπως ο $A\bar{B}\bar{C}$) δύο ή περισσότερες φορές κατά τη διαδικασία της ελαχιστοποίησης οδηγεί σε πιο ελαχιστοποιημένη εξίσωση Boole σε **μορφή αθροίσματος γινομένων** (λύση 2)

- Η λύση 2 με χρήση του T8 οδηγεί σε πιο ελαχιστοποιημένη εξίσωση Boole σε **μορφή γινομένου αθροισμάτων** (λύση 3)

$$\bar{B} (A + \bar{C})$$

Επιλεγμένες ασκήσεις

■ Άσκηση 2.6

Ελαχιστοποιήστε την εξίσωση Boole σε κανονική μορφή αθροίσματος γινομένων της Άσκησης 2.2 – (β) (έχει 2 λύσεις σε μορφή αθροίσματος γινομένων)

$$Y = A\bar{C} + \bar{A}C + B\bar{C}$$

$$Y = A\bar{C} + \bar{A}C + \bar{A}B$$

- Ξεκινήστε από την εξίσωση Boole σε κανονική μορφή αθροίσματος γινομένων που προκύπτει από τον Πίνακα Αλήθειας

$$Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C}$$

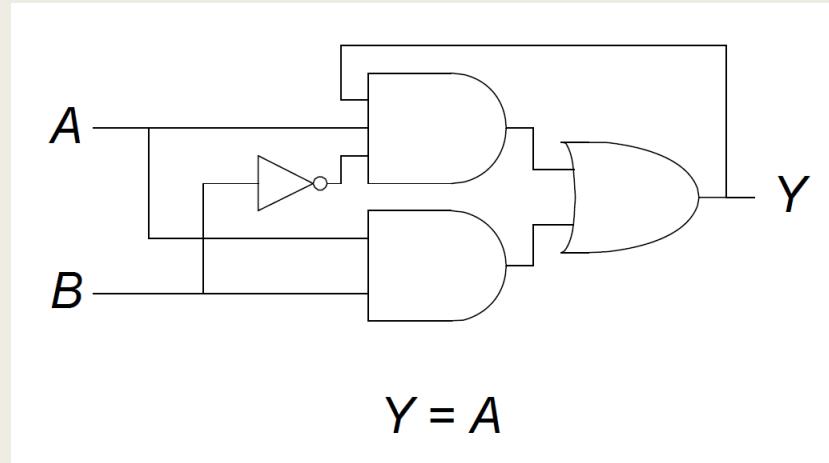
(b)

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Επιλεγμένες ασκήσεις

■ Άσκηση 2.20

Παραθέστε ένα παράδειγμα κυκλώματος το οποίο θα περιέχει μια κυκλική διαδρομή, αλλά θα εξακολουθεί μολαταύτα να είναι συνδυαστικό



Το Θεώρημα De Morgan

	Θεώρημα	Ονομασία
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \dots} = \overline{\overline{B}_0 + \overline{B}_1 + \overline{B}_2 \dots}$	DeMorgan
	Δυϊκό Θεώρημα	Ονομασία
T12'	$\overline{B_0 + B_1 + B_2 \dots} = \overline{\overline{B}_0 \bullet \overline{B}_1 \bullet \overline{B}_2 \dots}$	DeMorgan

Το συμπλήρωμα του γινομένου

είναι το

άθροισμα των συμπληρωμάτων

Δυϊκό: Το συμπλήρωμα των αθροισμάτων

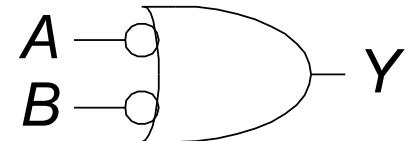
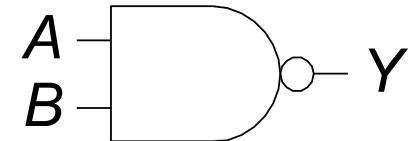
είναι το

γινόμενο των συμπληρωμάτων

Το Θεώρημα De Morgan

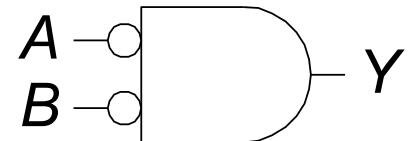
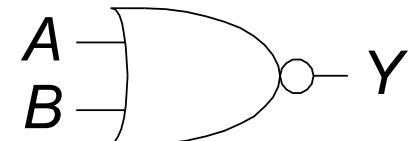
$$\blacksquare Y = \overline{AB} = \overline{A} + \overline{B}$$

- μια **πύλη NAND** είναι **ισοδύναμη** με μια πύλη **OR** με **αντεστραμμένες εισόδους**



$$\blacksquare Y = \overline{A + B} = \overline{A} \bullet \overline{B}$$

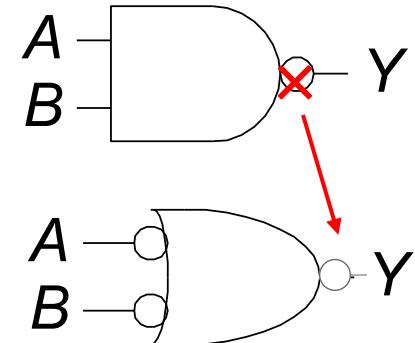
- μια **πύλη NOR** είναι **ισοδύναμη** με μια πύλη **AND** με **αντεστραμμένες εισόδους**



Το Θεώρημα De Morgan

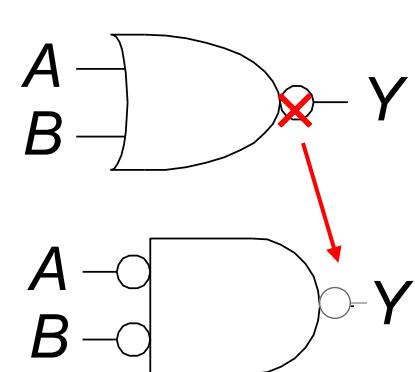
$$\blacksquare Y = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}} = AB$$

- μια **πύλη AND** είναι **ισοδύναμη** με μια **πύλη NOR** με **αντεστραμμένες εισόδους**



$$\blacksquare Y = \overline{\overline{A} + \overline{B}} = \overline{\overline{A} \bullet \overline{B}} = A + B$$

- μια **πύλη OR** είναι **ισοδύναμη** με μια πύλη **NAND** με **αντεστραμμένες εισόδους**



Καθολικότητα πυλών

■ Ερώτηση 2.4

Μια πύλη ή ένα σύνολο πυλών είναι **καθολική(-ό)** αν μπορεί να χρησιμοποιηθεί για την υλοποίηση οποιασδήποτε συνάρτησης Boole

- Για παράδειγμα, **το σύνολο πυλών AND, OR, NOT είναι καθολικό**

(α) Είναι μια πύλη AND καθολική από μόνη της; OXI

(β) Είναι το σύνολο πυλών OR και NOT καθολικό; NAI

(γ) Είναι μια πύλη NAND καθολική από μόνη της; NAI

Γιατί NAI ή γιατί OXI;

Εξετάζουμε εάν από το διαθέσιμο σύνολο πυλών μπορούμε να δημιουργήσουμε τις πύλες AND, OR, NOT

Σημείωση: Μία πύλη NAND ή μία πύλη NOR με βραχυκυκλωμένες εισόδους υλοποιεί μία πύλη NOT

Το θεώρημα De Morgan

- Ποια είναι η εξίσωση Boole σε μορφή αθροίσματος γινομένων;
 - Εφαρμόζουμε το θεώρημα De Morgan όσες φορές χρειαστεί

Παράδειγμα 1

$$\begin{aligned} Y &= \overline{(A + \overline{BD})\overline{C}} \\ &= \overline{(A + \overline{BD})} + \overline{\overline{C}} \\ &= (\overline{A} \bullet (\overline{BD})) + C \\ &= (\overline{A} \bullet (BD)) + C \\ &= \overline{A}BD + C \end{aligned}$$

Παράδειγμα 2

$$\begin{aligned} Y &= \overline{(ACE + D)} + B \\ &= \overline{\overline{(ACE + D)}} \bullet \overline{B} \\ &= (\overline{ACE} \bullet \overline{D}) \bullet \overline{B} \\ &= ((AC + E) \bullet D) \bullet \overline{B} \\ &= ((AC + \overline{E}) \bullet D) \bullet \overline{B} \\ &= (ACD + D\overline{E}) \bullet \overline{B} \\ &= \overline{A}\overline{B}CD + \overline{B}\overline{D}\overline{E} \end{aligned}$$

Επιλεγμένες ασκήσεις

■ Άσκηση 2.13

Ελαχιστοποιήστε την παρακάτω εξίσωση Boole χρησιμοποιώντας θεωρήματα της άλγεβρας Boole

- Να ονοματίσετε τα θεωρήματα που χρησιμοποιείτε

$$Y = \overline{A}\overline{B} + \overline{A}B\overline{C} + (\overline{A} + \overline{\overline{C}})$$

- Λύση

$$Y = \overline{A}$$

Οφέλη ελαχιστοποίησης

- Αφού η τελική εξίσωση είναι λογικά ισοδύναμη με την αρχική γιατί είναι τόσο σημαντική η διαδικασία της ελαχιστοποίησης;
 - Η ελαχιστοποίηση μειώνει το πλήθος των πυλών που χρησιμοποιούνται για τη φυσική υλοποίηση της εξίσωσης (συνάρτησης) *Boole*, με αποτέλεσμα η τελική υλοποίηση να έχει:
 - πιο μικρό μέγεθος και κόστος,
 - μικρότερη κατανάλωση ισχύος, και
 - πιθανώς αυξημένη ταχύτητα.

Συμπληρωματική συνάρτηση \bar{Y}

- Για κάθε συνάρτηση Y ορίζεται η συμπληρωματική της συνάρτηση \bar{Y}
 - Σε κάθε σειρά του πίνακα αλήθειας που η συνάρτηση Y έχει την τιμή 1, η συμπληρωματική συνάρτηση \bar{Y} έχει την τιμή 0
 - Σε κάθε σειρά του πίνακα αλήθειας που η συνάρτηση Y έχει την τιμή 0, η συμπληρωματική συνάρτηση \bar{Y} έχει την τιμή 1
- Παράδειγμα

A	B	Y	\bar{Y}
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Κανονική μορφή αθροίσματος γινομένων

- Μπορούμε να βρούμε την εξίσωση Boole από οποιονδήποτε πίνακα αληθείας **αθροίζοντας (πράξη OR) καθέναν από τους ελαχιστόρους (πράξη AND)** για τους οποίους η έξοδος έχει τιμή '**1**' (TRUE)

A	B	Y	\bar{Y}	Ελαχιστόρος
0	0	0	1	$\bar{A} \bar{B}$
0	1	0	1	$\bar{A} B$
1	0	1	0	$A \bar{B}$
1	1	1	0	$A B$

$$Y = F(A, B) = A\bar{B} + AB = \Sigma(m_2, m_3) = \Sigma(2, 3)$$

$$\bar{Y} = F(A, B) = \bar{A}\bar{B} + \bar{A}B = \Sigma(m_0, m_1) = \Sigma(0, 1)$$

Εύρεση της μορφής γινομένου αθροισμάτων*

- Χρησιμοποιώντας το θεώρημα De Morgan, βρείτε την κανονική μορφή γινομένου αθροισμάτων της **συνάρτησης Y** από τη μορφή αθροίσματος γινομένων της **συμπληρωματικής της συνάρτησης \bar{Y}**
 - *Εφαρμόζουμε δύο φορές το θεώρημα De Morgan*

$$\bar{Y} = \overline{AB} + \overline{A}B$$

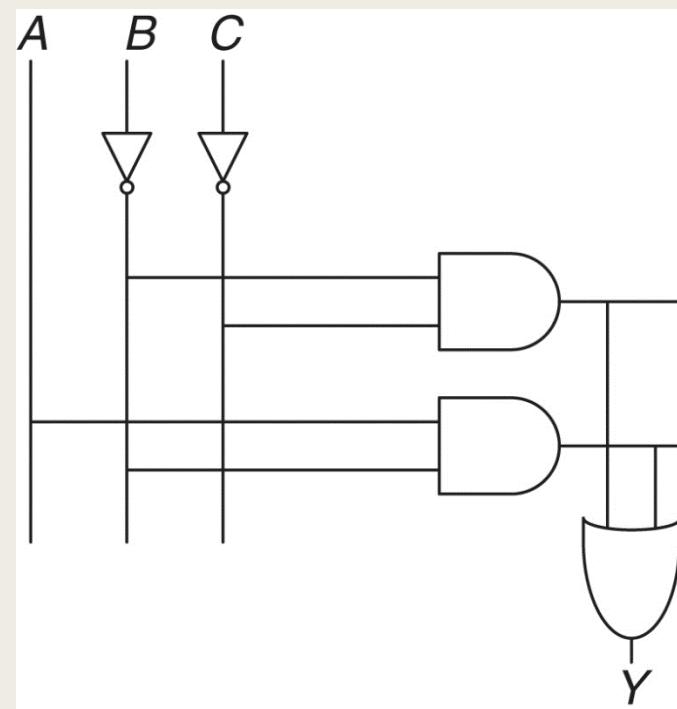
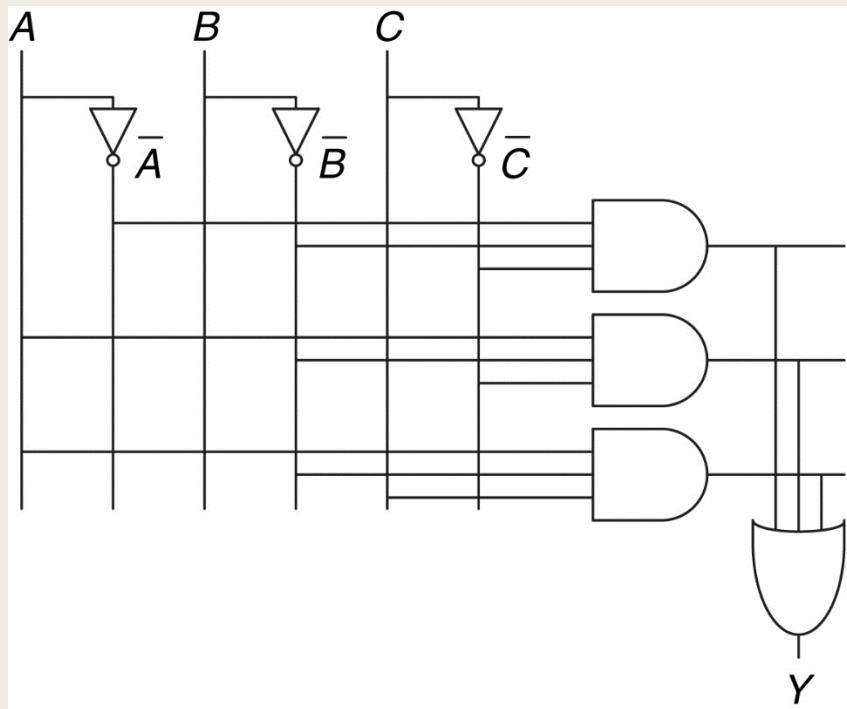
$$\bar{\bar{Y}} = Y = \overline{\overline{AB} + \overline{A}B} = (\overline{\overline{AB}})(\overline{\overline{A}B}) = (A+B)(A+\overline{B})$$

- *Iσχύει ο εξής κανόνας:*
 - Η συμπληρωματική μίας συνάρτησης προκύπτει, εάν στη συνάρτηση:
 - *οι μεταβλητές αλλάζουν από X σε \bar{X} και από \bar{X} σε X*
 - *οι τελεστές • (AND) και + (OR) εναλλαχθούν χωρίς να αλλάξει η ιεραρχία των πράξεων*

*Παράδειγμα 2.4

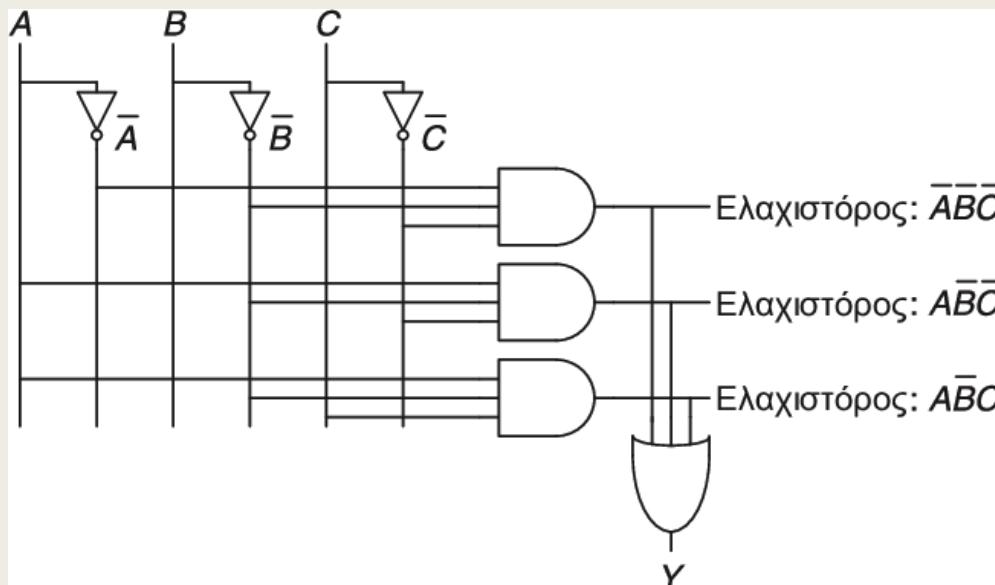
Σχηματικά διάγραμματα

- Το **σχηματικό διάγραμμα** είναι ένα διάγραμμα του ψηφιακού κυκλώματος στο οποίο φαίνονται τα στοιχεία (π.χ. πύλες) και τα σύρματα που τα συνδέουν
 - Παράδειγμα: $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$ (κανονική μορφή)
 - $Y = \bar{B}\bar{C} + A\bar{B}$ (ελαχιστοποιημένη – πρότυπη μορφή)
 - Υλοποίηση με λιγότερες πύλες και πιο γρήγορη



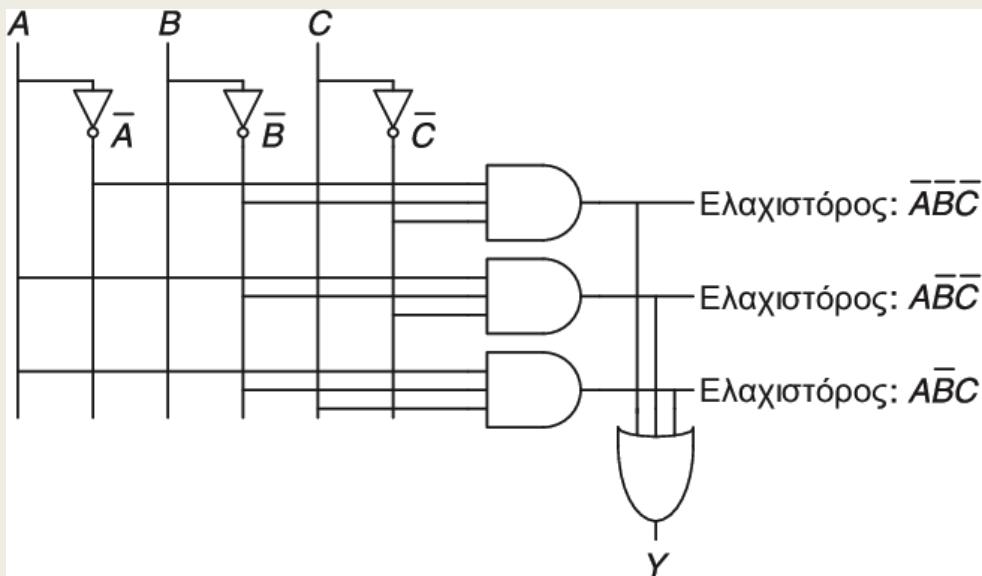
Σχηματικά διαγράμματα: κανόνες

- Οι είσοδοι βρίσκονται στην αριστερή (ή την πάνω) πλευρά
- Οι έξοδοι βρίσκονται στη δεξιά (ή την κάτω) πλευρά
- Οποτεδήποτε είναι εφικτό, οι πύλες θα πρέπει να έχουν κατεύθυνση **από τα αριστερά προς τα δεξιά**
- Είναι καλύτερο να χρησιμοποιούνται **σύρματα σε ευθεία** αντί για σύρματα με πολλές γωνίες.



Σχηματικά διαγράμματα: κανόνες

- Τα σύρματα συνδέονται πάντα σε **επαφές Τ**
- Μια **τελεία** στο σημείο τομής συρμάτων υποδεικνύει ότι **υπάρχει σύνδεση μεταξύ των συρμάτων**
- Σύρματα τα οποία τέμνονται **χωρίς τελεία** δεν συνδέονται μεταξύ τους



τα σύρματα συνδέονται
σε επαφή Τ

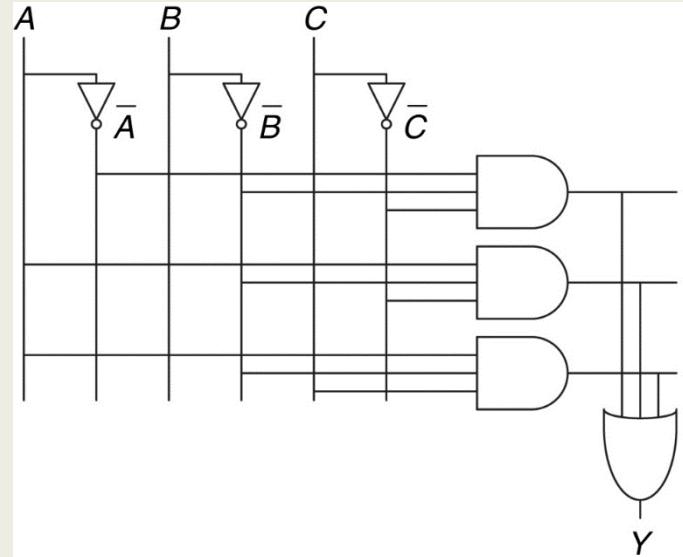
τα σύρματα συνδέονται
σε τελεία

τα σύρματα που
τέμνονται χωρίς τελεία
δεν συνδέονται

Σχηματικά διαγράμματα: κανόνες

- Τα βήματα για την σχεδίαση σχηματικών διαγραμμάτων για εξισώσεις Boole σε **μορφή αθροίσματος γινομένων**:

- Πρώτα σχεδιάζουμε στήλες για τις εισόδους
- Τοποθετούμε αντιστροφείς σε γειτονικές στήλες για να παρέχουμε τις συμπληρωματικές εισόδους, αν είναι απαραίτητο
- Σχεδιάζουμε γραμμές με πύλες AND για καθέναν από τους ελαχιστόρους
- Έπειτα, για κάθε έξοδο, σχεδιάζουμε μια πύλη OR η οποία συνδέεται με τις πύλες AND (τους ελαχιστόρους) που σχετίζονται με τη συγκεκριμένη έξοδο



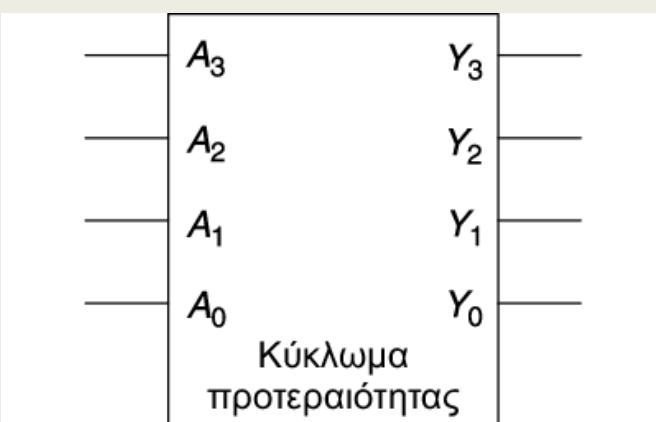
- Αυτό το στυλ ονομάζεται **προγραμματιζόμενη λογική διάταξη** (programmable logic array, PLA) επειδή οι αντιστροφείς, οι πύλες AND και οι πύλες OR διατάσσονται με συστηματικό τρόπο

Παράδειγμα 2.7 Κυκλώματα με πολλές εξόδους

- Ο Πρύτανης, ο Πρόεδρος ενός Τμήματος, ένας βοηθός διδασκαλίας και ο υπεύθυνος κοινωνικών εκδηλώσεων ενός Πανεπιστημίου χρησιμοποιούν ενίοτε το **κεντρικό αμφιθέατρο** του ίδρυματος με σκοπό την οργάνωση εκδηλώσεων
- Αυτό έχει σαν αποτέλεσμα μερικές φορές, παραπάνω από ένα άτομα να έχουν κάνει κράτηση του κεντρικού αμφιθέατρου την ίδια στιγμή
- Το σύστημα έχει τέσσερις εισόδους $A_3 - A_0$ και τέσσερις εξόδους $Y_3 - Y_0$
- Κάθε χρήστης ενεργοποιεί την είσοδό του όταν αιτείται τη χρήση του αμφιθέατρου
 - $A_3 = 1$ (Πρύτανης), $A_2 = 1$ (Πρόεδρος ενός Τμήματος),
 $A_1 = 1$ (βοηθός διδασκαλίας) και $A_0 = 1$ (υπεύθυνος εκδηλώσεων)
- Το σύστημα ενεργοποιεί το πολύ μία έξοδο, παραχωρώντας το αμφιθέατρο στον χρήστη με την υψηλότερη προτεραιότητα
 - $Y_3 = 1$ (Πρύτανης), $Y_2 = 1$ (Πρόεδρος ενός Τμήματος),
 $Y_1 = 1$ (βοηθός διδασκαλίας) και $Y_0 = 1$ (υπεύθυνος εκδηλώσεων)
- Συμπληρώστε τον πίνακα αληθείας και γράψτε τις εξισώσεις Boole για το σύστημα. Σχεδιάστε ένα κύκλωμα γι' αυτή τη συνάρτηση

Παράδειγμα 2.7

Η συγκεκριμένη συνάρτηση ονομάζεται **κύκλωμα προτεραιότητας** (priority circuit) με τέσσερις εισόδους



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Παράδειγμα 2.7: κύκλωμα προτεραιότητας

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

- Το Y_3 έχει την **μεγαλύτερη προτεραιότητα**, άρα όταν αυτό είναι TRUE τότε οι τιμές των υπολοίπων μεταβλητών εισόδου είναι αδιάφορες
- Το Y_2 έχει τιμή TRUE **αν είναι ενεργοποιημένο το A_2** και δεν είναι ενεργοποιημένο το A_3 . Οι τιμές των A_1 και A_0 είναι αδιάφορες.
- Το Y_1 έχει τιμή TRUE **αν είναι ενεργοποιημένο το A_1** και δεν είναι ενεργοποιημένη καμία από τις εισόδους A_3 και A_2 . Η τιμή του A_0 είναι αδιάφορη.
- Το Y_0 έχει τιμή TRUE **αν είναι ενεργοποιημένο το A_0** και δεν είναι ενεργοποιημένη κάποια άλλη είσοδος

Παράδειγμα 2.7: κύκλωμα προτεραιότητας

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

$$Y_3 = A_3$$

$$Y_2 = \overline{A}_3 A_2$$

$$Y_1 = \overline{A}_3 \overline{A}_2 A_1$$

$$Y_0 = \overline{A}_3 \overline{A}_2 \overline{A}_1 A_0$$

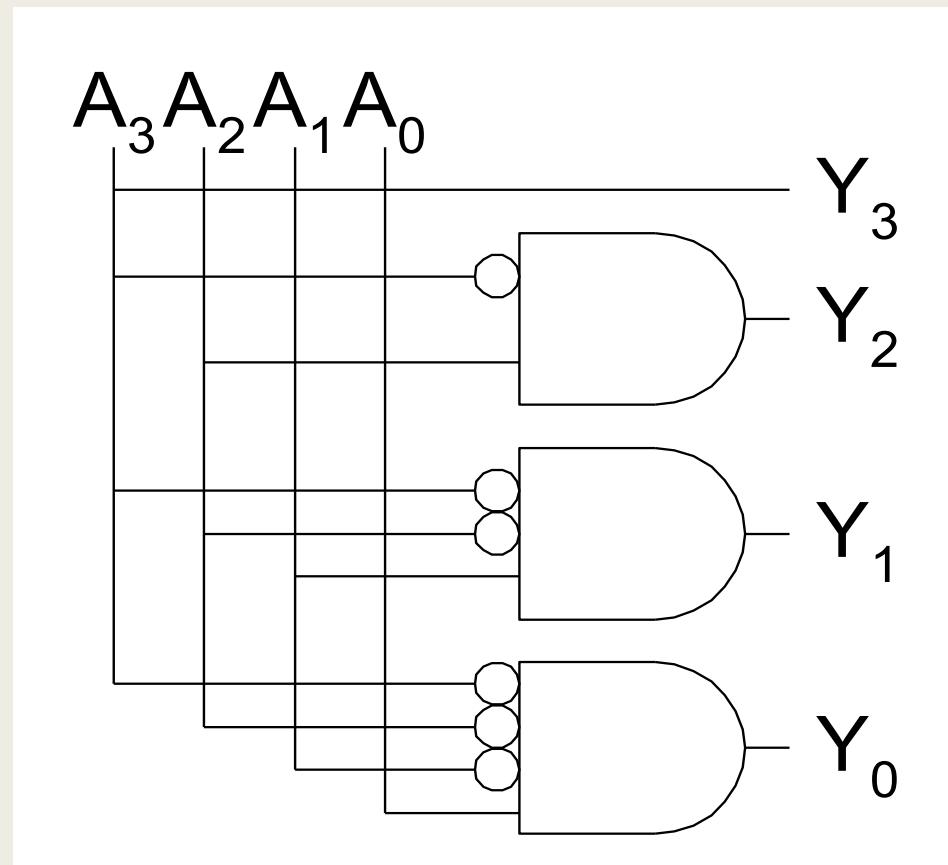
Παράδειγμα 2.7: κύκλωμα προτεραιότητας

$$Y_3 = A_3$$

$$Y_2 = \overline{A_3} A_2$$

$$Y_1 = \overline{A_3} \overline{A_2} A_1$$

$$Y_0 = \overline{A_3} \overline{A_2} \overline{A_1} A_0$$



Χρήση του «X» στους πίνακες αλήθειας

- Σε έναν πίνακα αλήθειας η χρήση του **συμβόλου X** σημαίνει ότι υπάρχουν **αδιάφορες τιμές**.
 - *To X μπορεί να είναι είτε 0, είτε 1 (όχι και τα δύο ταυτόχρονα)*
- Το X στις εισόδους απλοποιεί τον πίνακα αλήθειας (μειώνει τις γραμμές)
- Το Y_3 έχει την **μεγαλύτερη προτεραιότητα**, άρα όταν αυτό είναι TRUE τότε οι τιμές των υπολοίπων μεταβλητών εισόδου είναι αδιάφορες
- Το Y_2 έχει τιμή TRUE **αν είναι ενεργοποιημένο το A_2** και δεν είναι ενεργοποιημένο το A_3 . Οι τιμές των A_1 και A_0 είναι αδιάφορες.
- Το Y_1 έχει τιμή TRUE **αν είναι ενεργοποιημένο το A_1** και δεν είναι ενεργοποιημένη καμία από τις εισόδους A_3 και A_2 . Η τιμή του A_0 είναι αδιάφορη.
- Το Y_0 έχει τιμή TRUE **αν είναι ενεργοποιημένο το A_0** και δεν είναι ενεργοποιημένη κάποια άλλη είσοδος

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

Κωδικοποιητής προτεραιότητας

■ Άσκηση 2.36

Ένας κωδικοποιητής προτεραιότητας (priority encoder) διαθέτει 2^N εισόδους, διατεταγμένες από το 2^N-1 έως το 0

- Παράγει μια **δυαδική έξοδο των N bit** που υποδεικνύει το περισσότερο σημαντικό bit της εισόδου το οποίο έχει την τιμή ‘1’ (TRUE), ή την έξοδο μηδέν αν καμία από τις εισόδους δεν έχει την τιμή ‘1’ (TRUE)
- Παράγει επίσης το **σήμα εγκυρότητας VALID** (validation) το οποίο έχει την τιμή ‘1’ (TRUE) αν τουλάχιστον μία από τις εισόδους έχει την τιμή ‘1’ (TRUE)
- Σχεδιάστε τον πίνακα αληθείας ενός κωδικοποιητή προτεραιότητας οκτώ εισόδων, με εισόδους $A_{7:0}$ και έξόδους $Y_{2:0}$ και το σήμα VALID
- Παρέχετε μια ελαχιστοποιημένη εξίσωση Boole για κάθε έξοδο

Κωδικοποιητής προτεραιότητας

- Πίνακας αληθείας ενός κωδικοποιητή προτεραιότητας οκτώ εισόδων, με εισόδους $A_{7:0}$ και εξόδους $Y_{2:0}$ και VALID

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Y_2	Y_1	Y_0	VALID
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	X	0	0	1	1
0	0	0	0	0	1	X	X	0	1	0	1
0	0	0	0	1	X	X	X	0	1	1	1
0	0	0	1	X	X	X	X	1	0	0	1
0	0	1	X	X	X	X	X	1	0	1	1
0	1	X	X	X	X	X	X	1	1	0	1
1	X	X	X	X	X	X	X	1	1	1	1

- Το **σήμα VALID** στην έξοδο ενός κυκλώματος δηλώνει ότι η έξοδος δεδομένων αυτού του κυκλώματος έχει έγκυρα δεδομένα
 - Η έξοδος δεδομένων $Y_{2:0}$ είναι έγκυρη όταν υποδεικνύει το MSB της εισόδου $A_{7:0}$ το οποίο έχει την τιμή 1

Κωδικοποιητής προτεραιότητας

- Παρέχετε μια ελαχιστοποιημένη εξίσωση Boole για κάθε έξοδο

$$Y_2 = A_7 + A_6 + A_5 + A_4$$

$$Y_1 = A_7 + A_6 + \overline{A}_5 \overline{A}_4 A_3 + \overline{A}_5 \overline{A}_4 A_2$$

$$Y_0 = A_7 + \overline{A}_6 A_5 + \overline{A}_6 \overline{A}_4 A_3 + \overline{A}_6 \overline{A}_4 \overline{A}_2 A_1$$

- Για παράδειγμα στην έξοδο Y_0 ο όρος $\overline{A}_6 \overline{A}_4 A_3$ εξασφαλίζει ότι το Y_0 έχει την τιμή 1 όταν $A_6 = 0$, $A_4 = 0$ και $A_3 = 1$
 - Σημειώστε ότι το Y_0 έχει την τιμή 0 όταν $A_6 = 1$ ή $A_4 = 1$ ενώ το Y_0 έχει την τιμή 1 όταν $A_7 = 1$ ή $A_5 = 1$ και για αυτό τον λόγο δεν εμφανίζονται τα A_7 και A_5 στον συγκεκριμένο όρο

$$VALID = A_7 + A_6 + A_5 + A_4 + A_3 + A_2 + A_1 + A_0$$

Πολυεπίπεδη Συνδυαστική Λογική

- Η λογική είτε σε μορφή αθροίσματος γινομένων είτε σε μορφή γινομένου αθροισμάτων ονομάζεται **λογική δύο επιπέδων** (two-level logic) επειδή αποτελείται από λεκτικά συνδεδεμένα σε **ένα επίπεδο πυλών AND** το οποίο συνδέεται με **ένα επίπεδο πυλών OR**
 - οι αντιστροφείς στην είσοδο αγνοούνται στον υπολογισμό των επιπέδων (ενσωματώνονται στις πύλες)
- Συχνά οι σχεδιαστές κατασκευάζουν κυκλώματα με περισσότερα από δύο επίπεδα λογικών πυλών
- Αυτά τα πολυεπίπεδα συνδυαστικά κυκλώματα στοχεύουν στο να χρησιμοποιούν **λιγότερο υλικό** από τα αντίστοιχα κυκλώματα δύο επιπέδων (να έχουν μικρότερο κόστος σε υλικό)
 - Σε πολλές περιπτώσεις **μειώνεται η ταχύτητα** του κυκλώματος

Πολυεπίπεδη Συνδυαστική Λογική

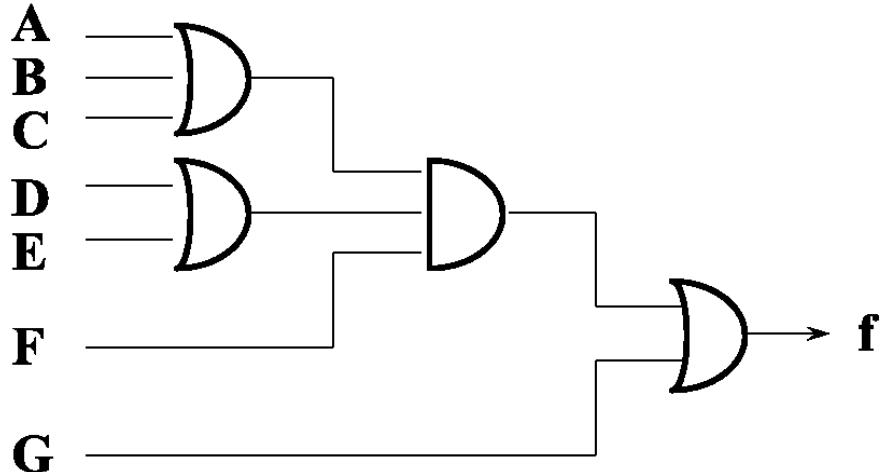
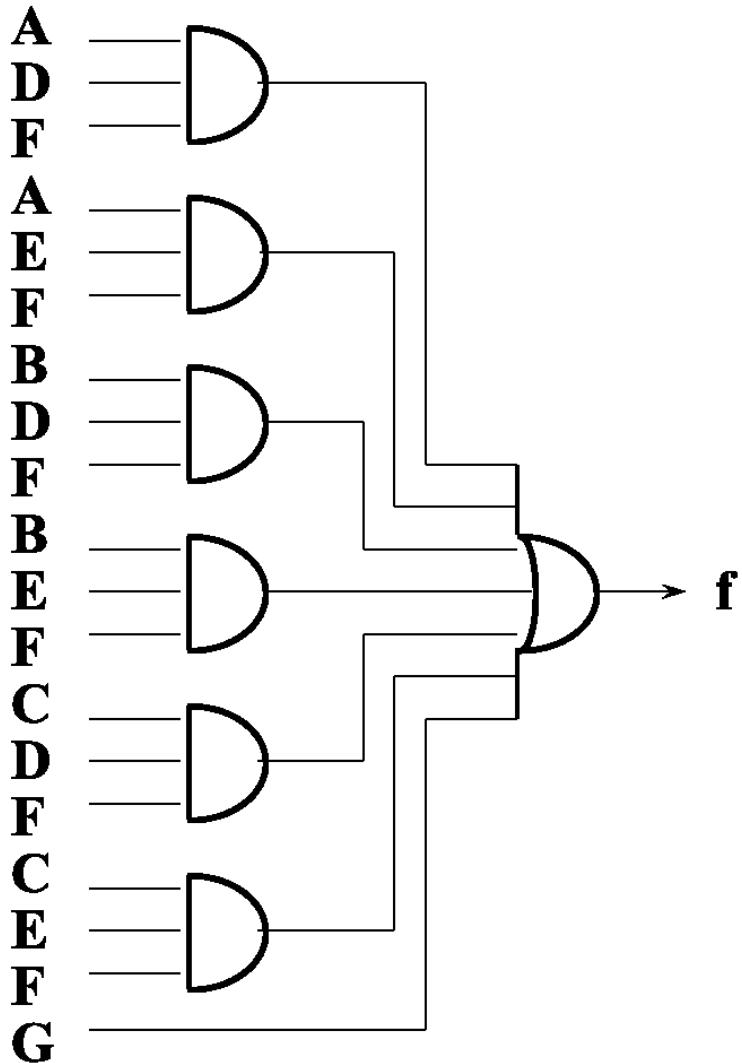
■ Εύρεση κοινών μεταβλητών

- Παράδειγμα: Η διεπίπεδη απλοποιημένη συνάρτηση f
 - $f = ADF+AEF+BDF+BEF+CDF+CEF+G$
- Απλοποιείται με την εφαρμογή της μεθόδου της εύρεσης κοινών μεταβλητών
 - *Εφαρμογή του θεωρήματος της επιμεριστικότητας (T8) όσες φορές απαιτείται*

$$\begin{aligned}f &= (AD+AE+BD+BE+CD+CE)F+G \\&= [(A+B+C)D+(A+B+C)E]F+G \\&= (A+B+C)(D+E)F+G\end{aligned}$$

Πολυεπίπεδη Συνδυαστική Λογική

■ Εύρεση κοινών μεταβλητών



- Στη συγκεκριμένη περίπτωση η πολυεπίπεδη υλοποίηση της συνάρτησης f **μικρότερου κόστους σε υλικό** είναι και **πιο γρήγορη**

- Η πύλη OR των 7 εισόδων υλοποιείται σε δύο επίπεδα με τρεις πύλες OR των 3 εισόδων

Μείωση υλικού με τη χρήση πυλών XOR

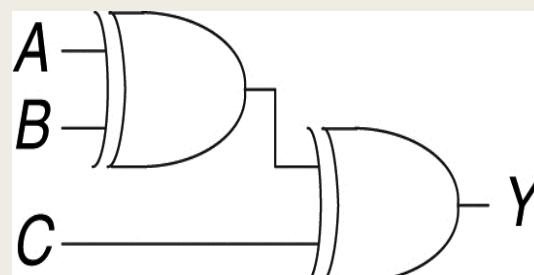
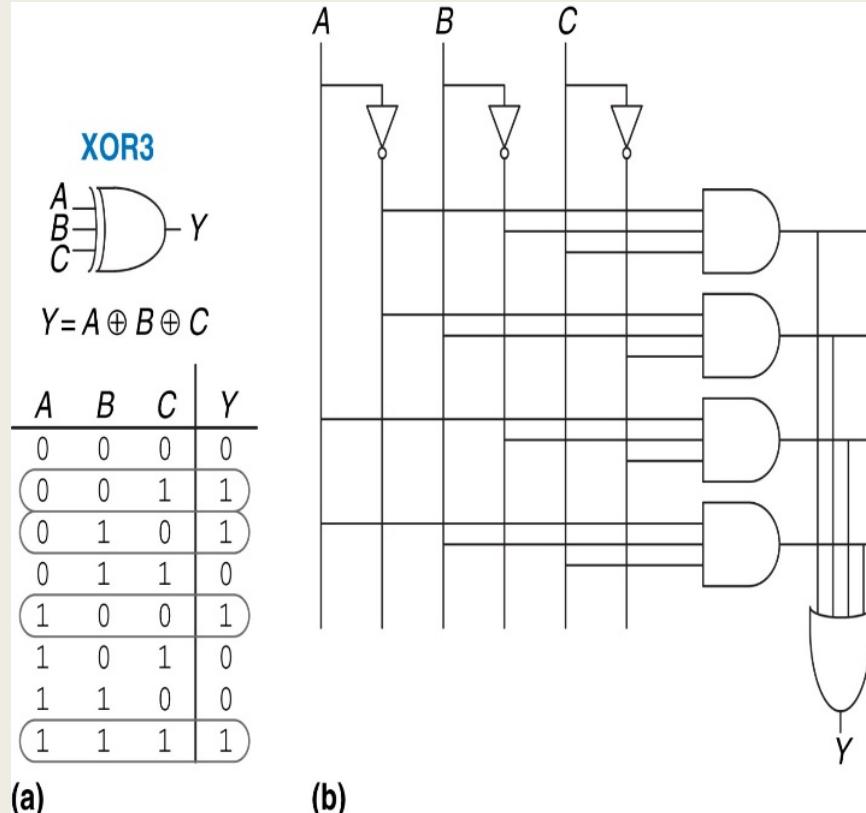
- Παράδειγμα: **Πύλη XOR με 3 εισόδους**
- Η εξίσωση Boole στην απλοποιημένη μορφή αθροίσματος γινομένων δύο επιπέδων είναι:

$$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

- Εάν προχωρήσουμε την ελαχιστοποίηση, ώστε να χρησιμοποιήσουμε την πύλη **XOR** έχουμε:

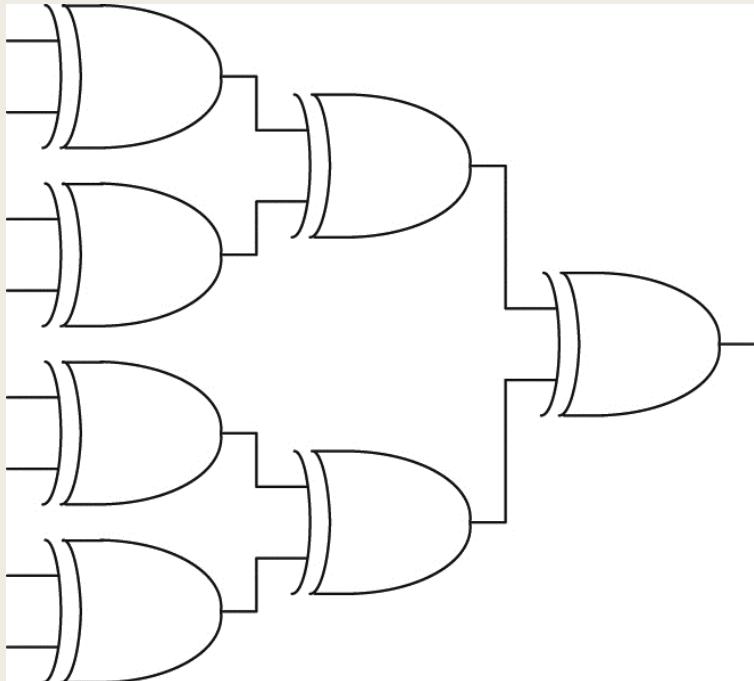
$$\begin{aligned} Y &= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC) \\ &= \bar{A}(B \oplus C) + A(\overline{B \oplus C}) \\ &= A \oplus B \oplus C = (A \oplus B) \oplus C \end{aligned}$$

- Άρα μπορεί να υλοποιηθεί με δύο πύλες XOR των 2 εισόδων



Μείωση υλικού με τη χρήση πυλών XOR

- Παράδειγμα: **Πύλη XOR με 8 εισόδους**
- Η εξίσωση Boole στην απλοποιημένη μορφή αθροίσματος γινομένων δύο επιπέδων θα απαιτούσε:
 - **128 πύλες AND των 8 εισόδων**
 - **1 πύλη OR των 128 εισόδων!**
- Μπορεί όμως να υλοποιηθεί με μόνο:
 - **7 πύλες XOR των 2 εισόδων σε 3 επίπεδα**
 - στο 1o επίπεδο είναι 4 πύλες,
στο 2o επίπεδο 2 πύλες
και στο 3o επίπεδο 1 πύλη



Κώδικες ανίχνευσης λαθών

- Σε ένα ψηφιακό σύστημα ορίζουμε ως **λάθος**, την αλλαγή τιμής σε ένα ή περισσότερα ψηφία πληροφορίας ($0 \rightarrow 1$ ή $1 \rightarrow 0$)
 - *Παράδειγμα απλού λάθους (αλλαγή τιμής σε ένα ψηφίο)*
 - $1010 \rightarrow 101\textcolor{red}{1}$, $1010 \rightarrow 10\textcolor{red}{0}$
- Για να ανιχνεύσουμε την ύπαρξη λαθών κωδικοποιούμε την πληροφορία που παράγουμε, μεταδίδουμε ή αποθηκεύουμε σε έναν **κώδικα ανίχνευσης λαθών**, ο οποίος αποτελείται από **κωδικές λέξεις** και **μη-κωδικές λέξεις**
- Η ύπαρξη λάθους ανιχνεύεται με την **εμφάνιση μίας μη-κωδικής λέξης** κατά τον έλεγχο της κωδικοποιημένης πληροφορίας
- Ως **απόσταση** μεταξύ δύο κωδικών λέξεων ορίζεται το πλήθος των ψηφίων στα οποία αυτές διαφέρουν
- Ένας κώδικας ανίχνευσης λαθών ανιχνεύει όλα τα **απλά λάθη**, εάν η ελάχιστη απόσταση μεταξύ όλων των πιθανών ζευγών κωδικών λέξεων είναι 2

Κώδικες άρτιας ή περιττής ισοτιμίας

- Προσθέτουμε στην πληροφορία των N bit ένα επιπλέον ψηφίο, **το ψηφίο ισοτιμίας P (parity bit)** ώστε το πλήθος των 1 να είναι άρτιο (ή περιττό) σε όλες τις κωδικές λέξεις του κώδικα άρτιας (ή περιττής) ισοτιμίας
- Κωδικές λέξεις του κώδικα άρτιας (ή περιττής) ισοτιμίας

Άρτιας ισοτιμίας

XYZ	P_{even}
000	0
001	1
010	1
100	1
011	0
101	0
110	0
111	1

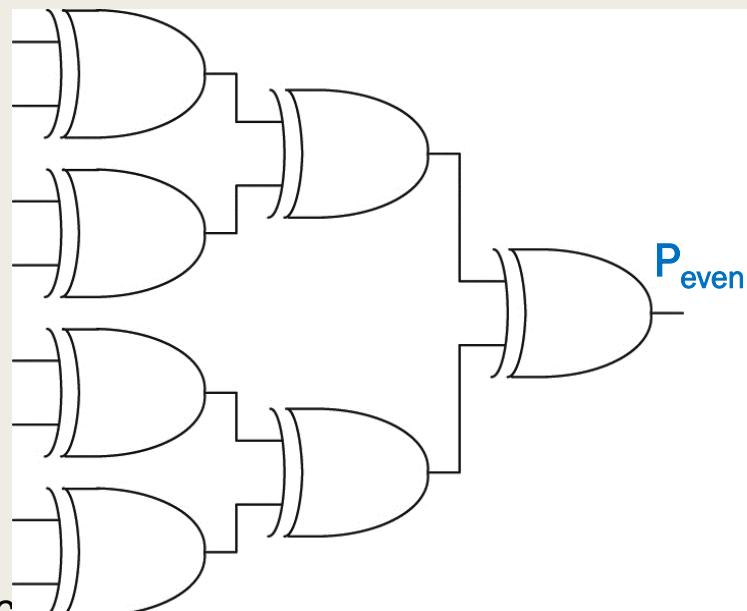
Περιττής ισοτιμίας

XYZ	P_{odd}
000	1
001	0
010	0
100	0
011	1
101	1
110	1
111	0

Οι μη κωδικές λέξεις του κώδικα άρτιας (ή περιττής) ισοτιμίας είναι οι κωδικές λέξεις του κώδικα περιττής (ή άρτιας) ισοτιμίας

Κώδικες άρτιας ή περιπτής ισοτιμίας

- Ο κώδικας ισοτιμίας είναι ο πιο απλός κώδικας ανίχνευσης λαθών και ανιχνεύει **απλά λάθη ή ένα περιπτό πλήθος λαθών**
 - *Είναι κώδικας απόστασης 2*
- Το κύκλωμα που παράγει το ψηφίο ισοτιμίας για μία **πληροφορία των N bit** είναι:
 - **μία πύλη XOR με N εισόδους για τον κώδικα άρτιας ισοτιμίας** ή
 - **μία πύλη XNOR με N εισόδους για τον κώδικα περιπτής ισοτιμίας**
- Στην πράξη η κωδικοποίηση στον κώδικα ισοτιμίας γίνεται συνήθως **ανά byte (8 bit)**
 - *Στο σχήμα φαίνεται η πύλη XOR με 8 εισόδους που παράγει το ψηφίο άρτιας ισοτιμίας P_{even} για ένα byte πληροφορίας*



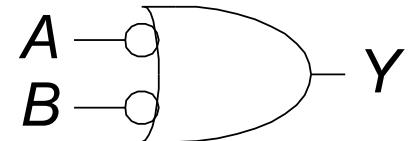
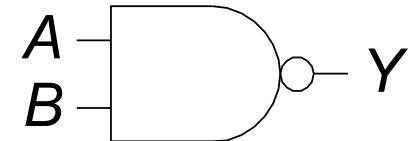
΄Ωθηση φυσαλίδας

- Οπως προαναφέρθηκε για τα κυκλώματα CMOS προτιμώνται οι πύλες NAND και NOR αντί των πυλών AND και OR.
- Όμως, η ανάγνωση της εξίσωσης με απλή εξέταση για ένα πολυεπίπεδο κύκλωμα με πύλες NAND και NOR ενδέχεται να αποδειχθεί αρκετά δύσκολη.
- Η **ώθηση φυσαλίδων** αποτελεί έναν χρήσιμο τρόπο για να ξανασχεδιάζουμε αυτά τα κυκλώματα έτσι ώστε οι φυσαλίδες να αλληλοακυρώνονται και να καθίσταται πιο εύκολος ο προσδιορισμός της συνάρτησης
- Βασίζεται στο **Θεώρημα De Morgan**

Το Θεώρημα De Morgan

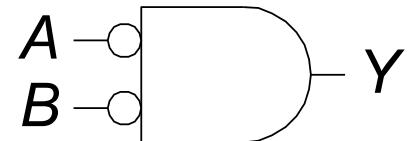
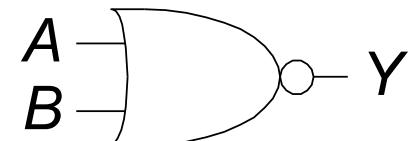
$$\blacksquare Y = \overline{AB} = \overline{A} + \overline{B}$$

- μια **πύλη NAND** είναι **ισοδύναμη** με μια πύλη **OR** με **αντεστραμμένες εισόδους**



$$\blacksquare Y = \overline{A + B} = \overline{A} \bullet \overline{B}$$

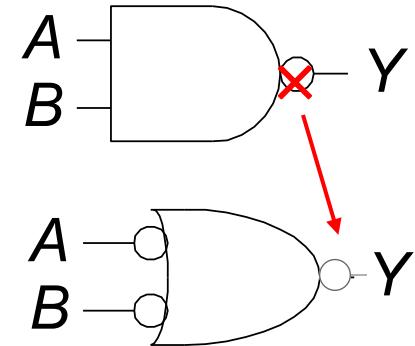
- μια **πύλη NOR** είναι **ισοδύναμη** με μια πύλη **AND** με **αντεστραμμένες εισόδους**



Το Θεώρημα De Morgan

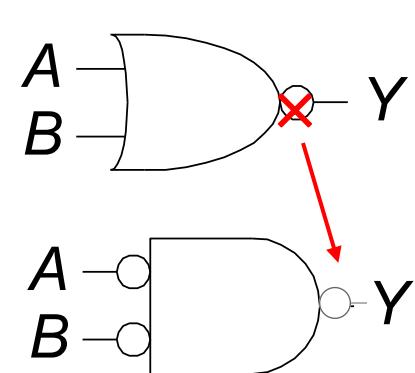
$$\blacksquare Y = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}} = AB$$

- μια **πύλη AND** είναι **ισοδύναμη** με μια **πύλη NOR** με **αντεστραμμένες εισόδους**



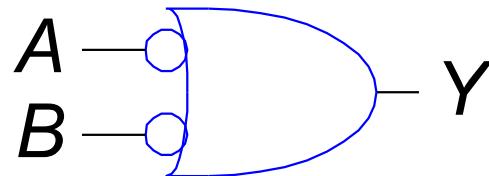
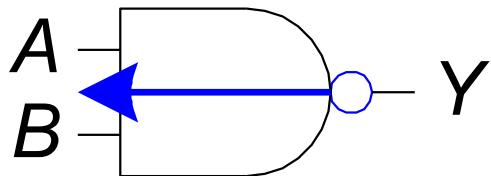
$$\blacksquare Y = \overline{\overline{A} + \overline{B}} = \overline{\overline{A} \bullet \overline{B}} = A + B$$

- μια **πύλη OR** είναι **ισοδύναμη** με μια πύλη **NAND** με **αντεστραμμένες εισόδους**

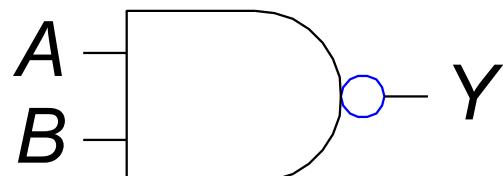
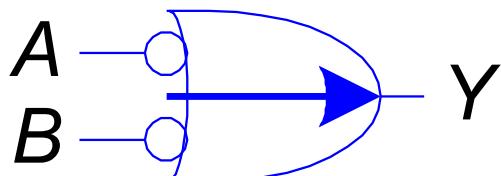


Ωθηση φυσαλίδας

- Ωθηση φυσαλίδων **προς τα πίσω**
 - αλλάζει το σώμα της πύλης (από AND/OR σε OR/AND)
 - φυσαλίδες σε όλες τις εισόδους της πύλης

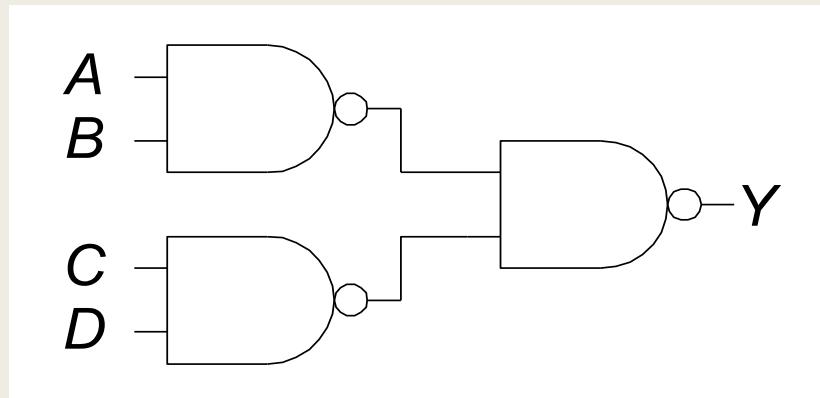


- Ωθηση φυσαλίδων **προς τα μπροστά**
 - αλλάζει το σώμα της πύλης (από AND/OR σε OR/AND)
 - μια φυσαλίδα στην έξοδο της πύλης



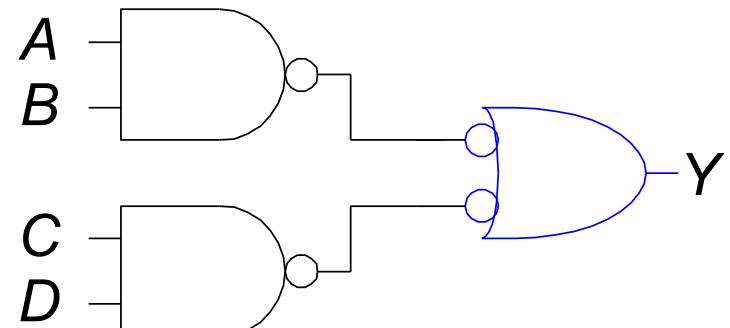
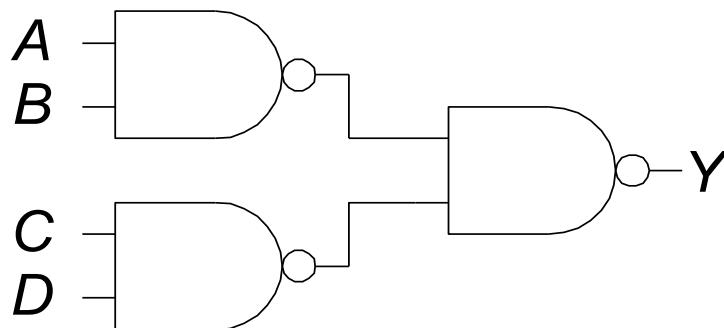
Ώθηση φυσαλίδας

- Ποια είναι η εξίσωση Boole του παρακάτω κυκλώματος?
 - Οι πύλες *NAND* και *NOR* χωρίς τη χρήση των ισοδυνάμων τους δεν βοηθούν στο να βρούμε την εξίσωση
 - Απαιτείται η εφαρμογή της ώθησης φυσαλίδας



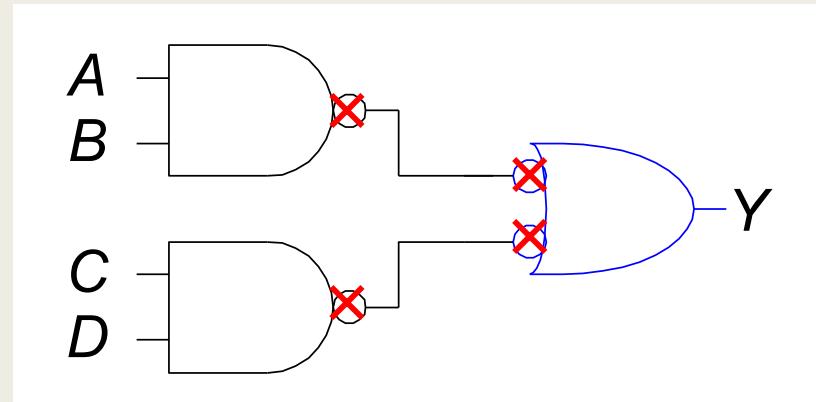
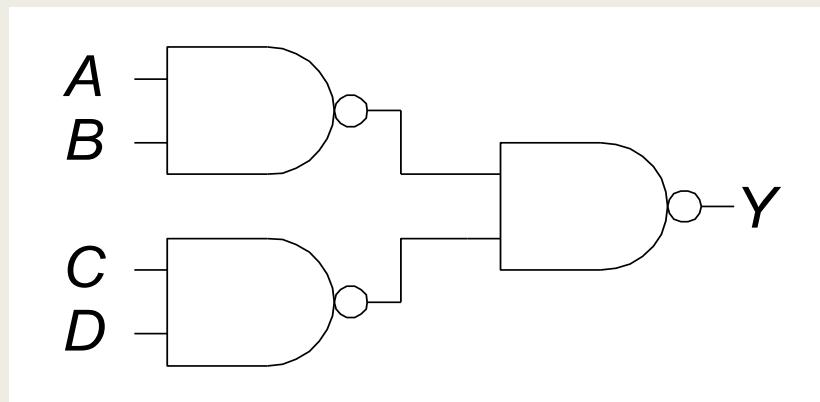
΄Ωθηση φυσαλίδας

- Ποια είναι η εξίσωση Boole του παρακάτω κυκλώματος?
 - Οι πύλες *NAND* και *NOR* χωρίς τη χρήση των ισοδυνάμων τους δεν βοηθούν στο να βρούμε την εξίσωση
 - Απαιτείται η εφαρμογή της ώθησης φυσαλίδας
 - Ξεκινάμε από την έξοδο με πορεία προς τα πίσω για να φύγει η αντιστροφή



Ώθηση φυσαλίδας

- Ποια είναι η εξίσωση Boole του παρακάτω κυκλώματος?
 - Οι πύλες *NAND* και *NOR* χωρίς τη χρήση των ισοδυνάμων τους δεν βοηθούν στο να βρούμε την εξίσωση
 - Απαιτείται η εφαρμογή της ώθησης φυσαλίδας
 - Ξεκινάμε από την έξοδο με πορεία προς τα πίσω για να φύγει η αντιστροφή
 - Δύο φυσαλίδες στην ίδια διασύνδεση αλληλοακυρώνονται



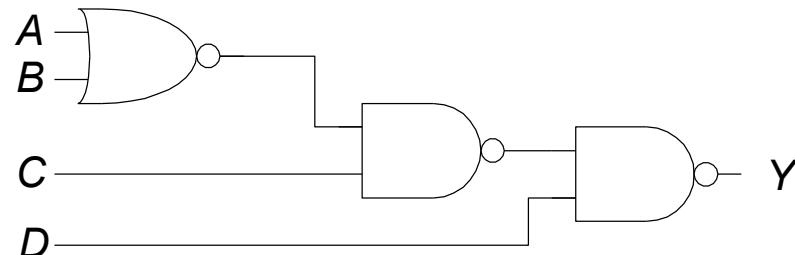
- Άρα η εξίσωση Boole είναι: $Y = AB + CD$

΄Ωθηση φυσαλίδας: κανόνες

- Ξεκινάμε από την **έξοδο** του κυκλώματος, και δουλεύουμε **με κατεύθυνση προς τις εισόδους**
- «Σπρώχνουμε» τυχόν φυσαλίδες που υπάρχουν στην τελική έξοδο **προς τα πίσω**, δηλαδή προς τις εισόδους
 - έτσι μπορούμε να «διαβάσουμε» την εξίσωση σε σχέση με την αληθινή έξοδο (**Υ**), και όχι σε σχέση με το συμπλήρωμά της (**Ȳ**)
 - Δουλεύοντας προς τα πίσω, σχεδιάζουμε κάθε πύλη σε τέτοια μορφή ώστε σε μία διασύνδεση οι φυσαλίδες να **αλληλοακυρώνονται**
ή να μην υπάρχουν
 - Αν η τρέχουσα πύλη διαθέτει **φυσαλίδα σε κάποια είσοδο**, σχεδιάστε την **προηγούμενη πύλη με φυσαλίδα στην έξοδο**
 - Αν η τρέχουσα πύλη δεν έχει φυσαλίδες στις εισόδους, σχεδιάστε την **προηγούμενη πύλη χωρίς φυσαλίδα στην έξοδο**
- Στο τέλος, κάποιες τερματικές είσοδοι διαθέτουν φυσαλίδες
 - Αν η τερματική είσοδος **διαθέτει φυσαλίδα** τότε **αντιστρέφεται**

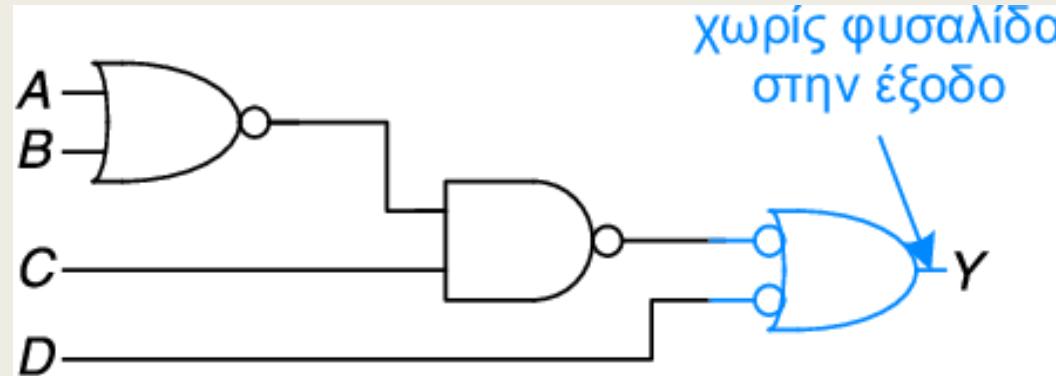
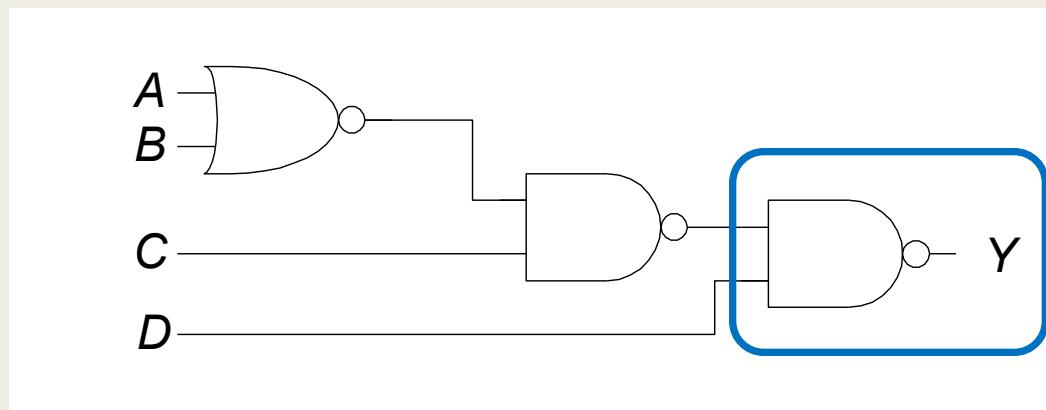
Ώρηση φυσαλίδας: παράδειγμα

- Ξεκινάμε από την έξοδο του κυκλώματος, και δουλεύουμε με κατεύθυνση προς τις εισόδους



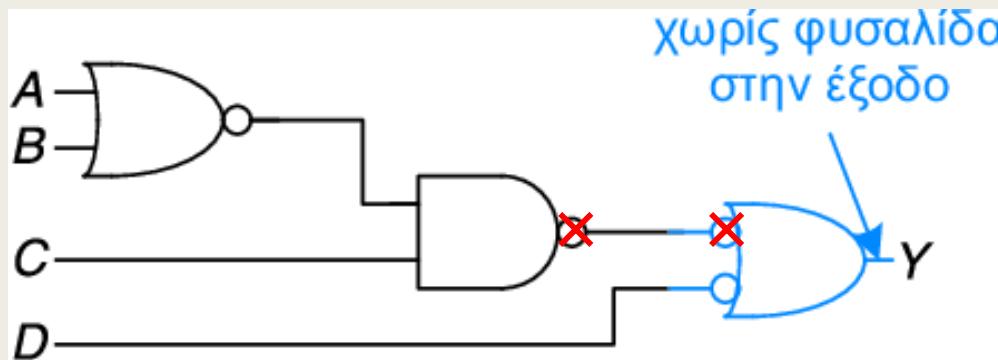
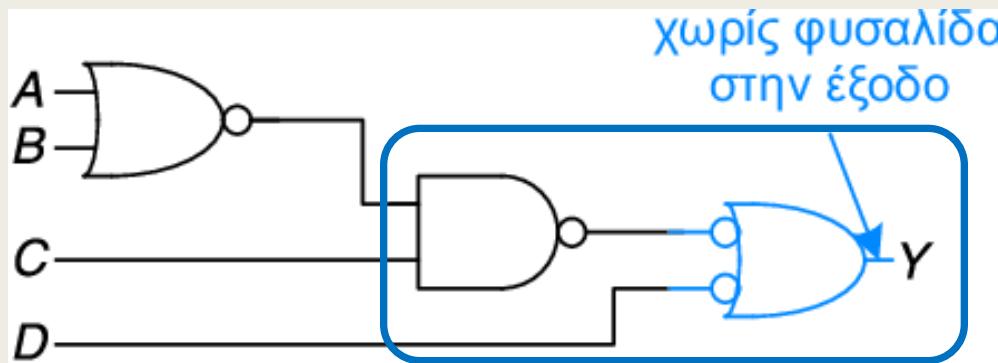
Ώθηση φυσαλίδας: παράδειγμα

- «Σπρώχνουμε» τυχόν φυσαλίδες που υπάρχουν στην τελική έξοδο **προς τα πίσω**, δηλαδή προς τις εισόδους, ώστε να βρούμε τη συνάρτηση Y και όχι τη συμπληρωματική της



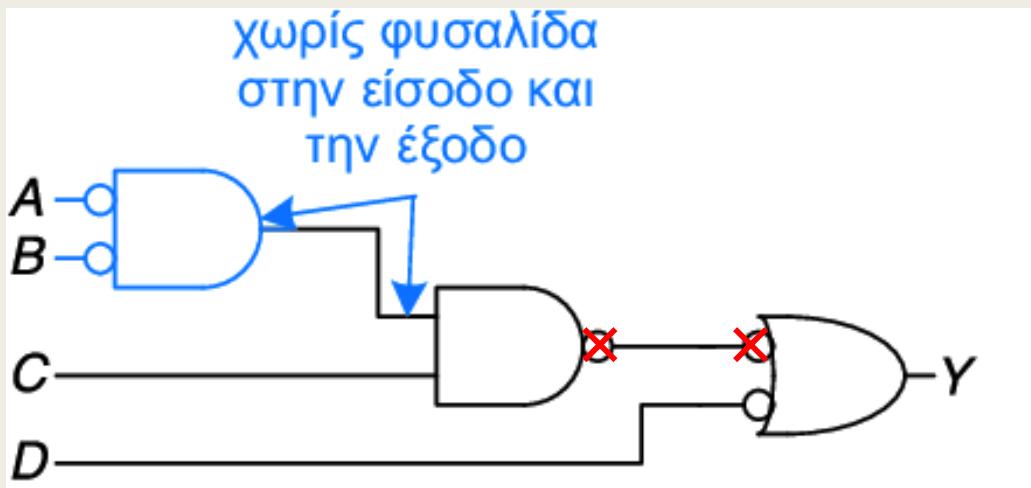
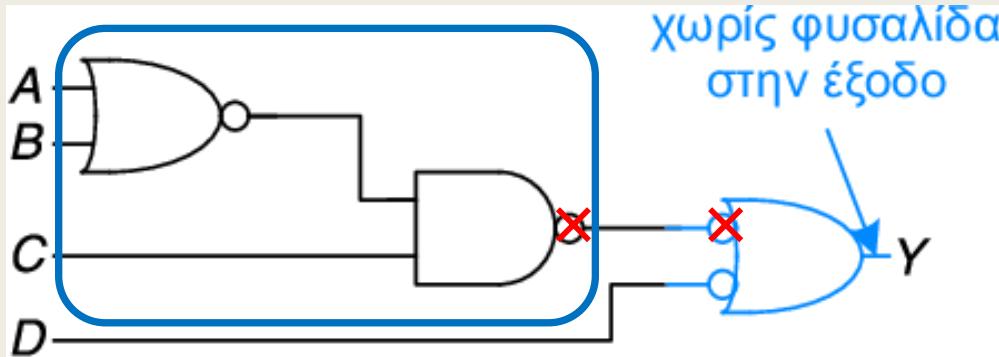
Ώρηση φυσαλίδας: παράδειγμα

- Αν η τρέχουσα πύλη διαθέτει φυσαλίδα σε κάποια είσοδο, σχεδιάστε την προηγούμενη πύλη με φυσαλίδα στην έξοδο
 - Υπάρχει τέτοια πύλη – δεν κάνουμε τίποτα
 - οι φυσαλίδες αλληλοακυρώνονται



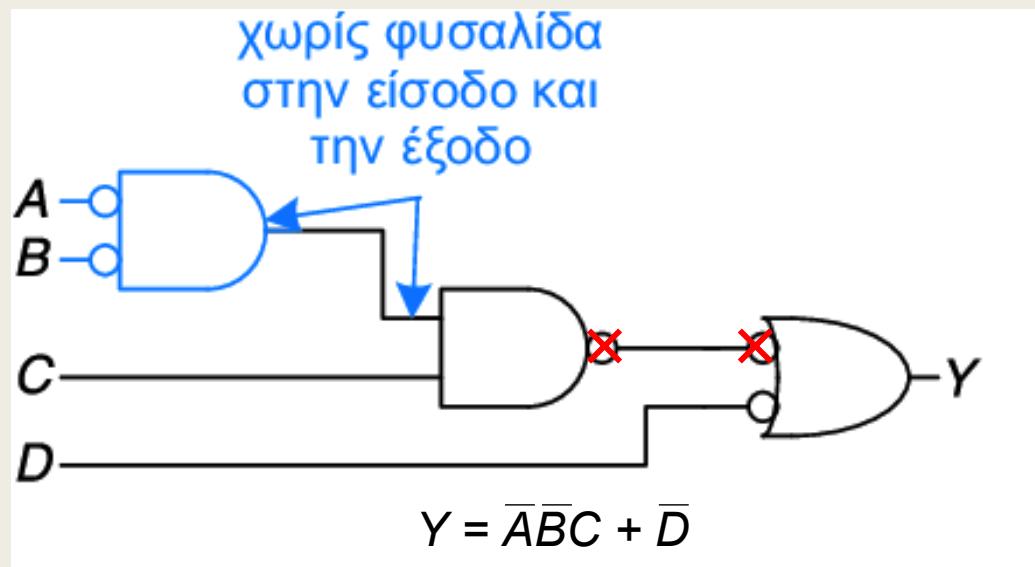
Ωθηση φυσαλίδας: παράδειγμα

- Αν η τρέχουσα πύλη δεν έχει φυσαλίδες στις εισόδους, σχεδιάστε την προηγούμενη πύλη **χωρίς φυσαλίδα στην έξοδο**

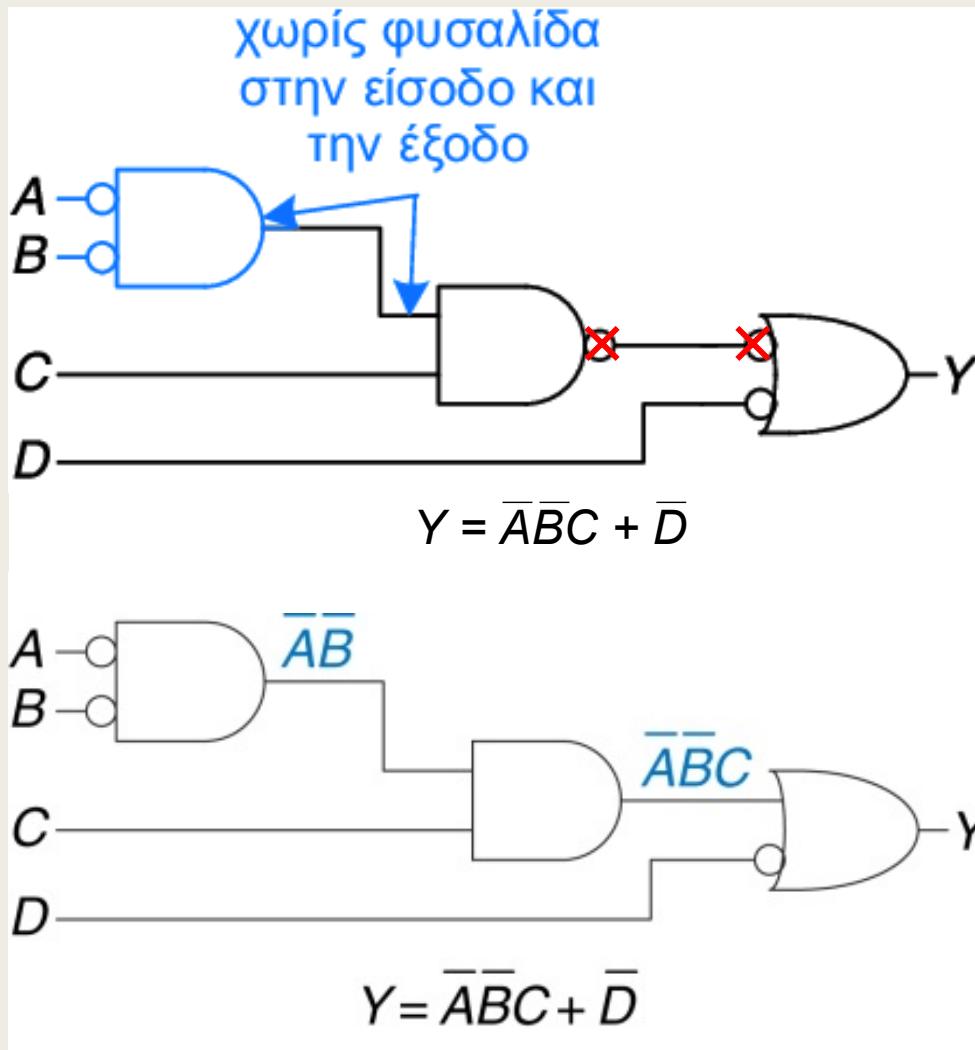


ΌΩθηση φυσαλίδας: παράδειγμα

- Στο τέλος, κάποιες τερματικές είσοδοι διαθέτουν φυσαλίδες
 - Αν η τερματική είσοδος **διαθέτει φυσαλίδα** τότε **αντιστρέφεται**
 - Στο παράδειγμα ισχύει για τις εισόδους **A**, **B** και **D**



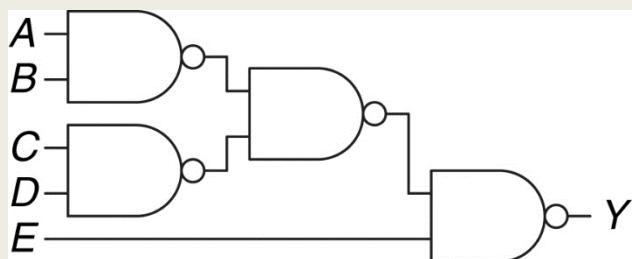
Ωθηση φυσαλίδας: ισοδύναμα κυκλώματα



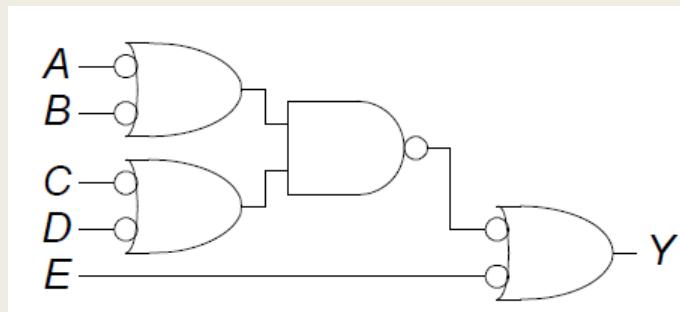
Επιλεγμένες ασκήσεις

■ Άσκηση 2.26

Χρησιμοποιώντας πύλες ισοδύναμες κατά De Morgan και μεθόδους «ώθησης» φυσαλίδων, ξανασχεδιάστε το κύκλωμα της εικόνας 2.83 ώστε να μπορείτε να βρείτε τις εξισώσεις Boole με απλή οπτική εξέταση. Γράψτε τις εξισώσεις Boole.



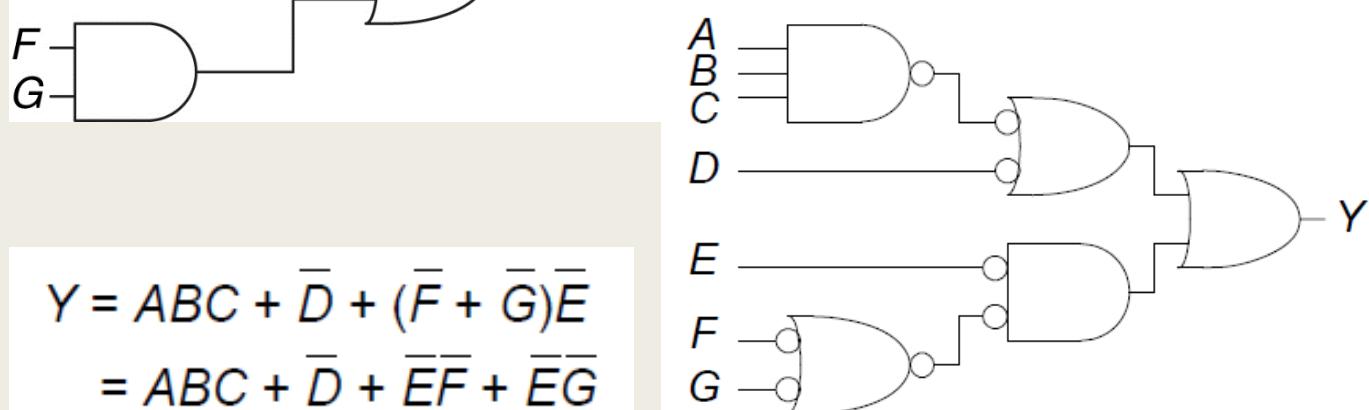
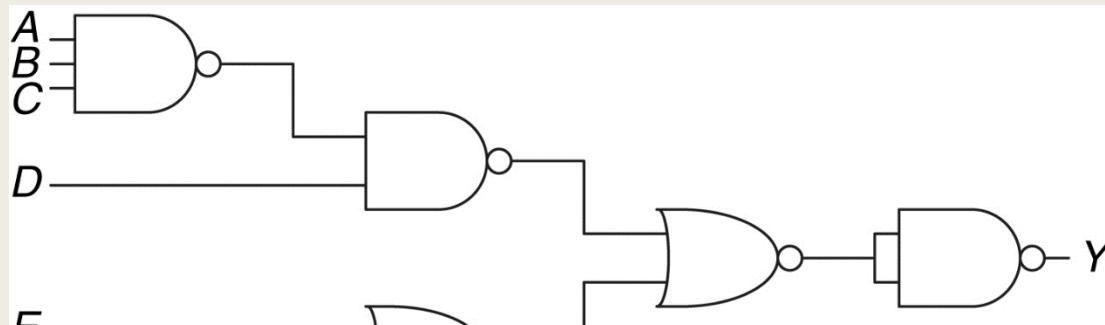
$$Y = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) + \overline{E}$$



Επιλεγμένες ασκήσεις

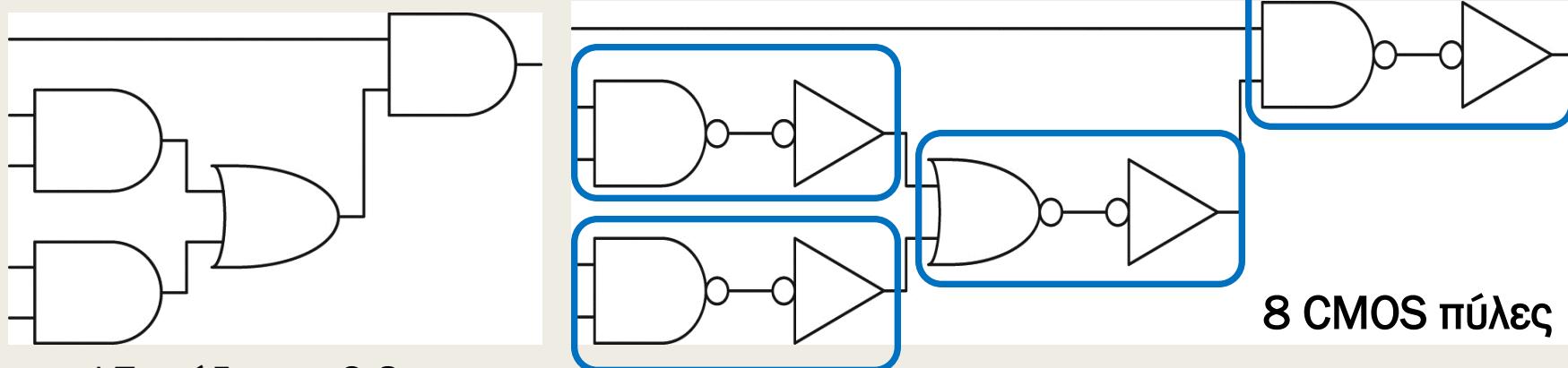
■ Άσκηση 2.27

Χρησιμοποιώντας πύλες ισοδύναμες κατά De Morgan και μεθόδους «ώθησης» φυσαλίδων, ξανασχεδιάστε το κύκλωμα της εικόνας 2.84 ώστε να μπορείτε να βρείτε τις εξισώσεις Boole με απλή οπτική εξέταση. Γράψτε τις εξισώσεις Boole.



Ώθηση φυσαλίδας: λογική CMOS*

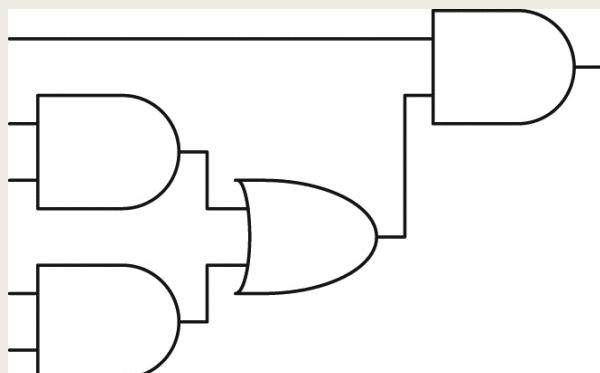
- Οι περισσότεροι σχεδιαστές σκέφτονται με γνώμονα τις πύλες AND και OR, αλλά στη CMOS τεχνολογία προτιμώνται **οι πύλες NAND, οι πύλες NOR και οι αντιστροφείς**.
- Χρησιμοποιήστε την «ώθηση» φυσαλίδων για να μετατρέψετε το κύκλωμα σε πύλες NAND, πύλες NOR και αντιστροφείς
- Μέθοδος 1 (μη αποδοτική):
 - Απλή αντικατάσταση κάθε πύλης AND με ένα ζεύγος πύλης NAND και αντιστροφέα, και κάθε πύλης OR με ένα ζεύγος πύλης NOR και αντιστροφέα



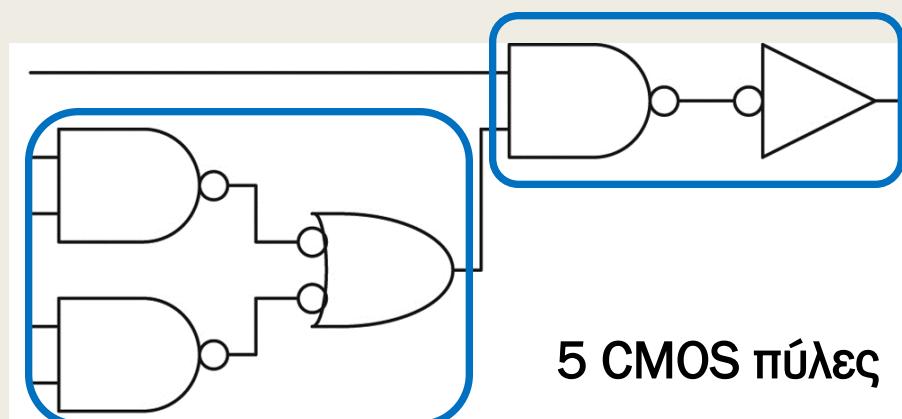
*Παράδειγμα 2.8

΄Ωθηση φυσαλίδας: λογική CMOS*

- Χρησιμοποιήστε την «ώθηση» φυσαλίδων για να μετατρέψετε το κύκλωμα σε πύλες NAND, πύλες NOR και αντιστροφείς
- Μέθοδος 2 (αποδοτική):
 - Προσθέτουμε ζεύγη φυσαλίδων στην έξοδο μιας πύλης και την είσοδο την επόμενης πύλης χωρίς να μεταβληθεί η συνάρτηση
 - Για NAND υλοποίηση μεταξύ πυλών AND (έξοδος) και OR (είσοδος)
 - Για NOR υλοποίηση μεταξύ πυλών OR (έξοδος) και AND (είσοδος)
 - Προσθέτουμε και αντιστροφείς, όπου απαιτείται
 - Όταν οι πύλες στην έξοδο και την είσοδο είναι ίδιες (AND ή OR)
 - Στην έξοδο για μετατροπή από AND σε NAND ή από OR σε NOR



*Παράδειγμα 2.8

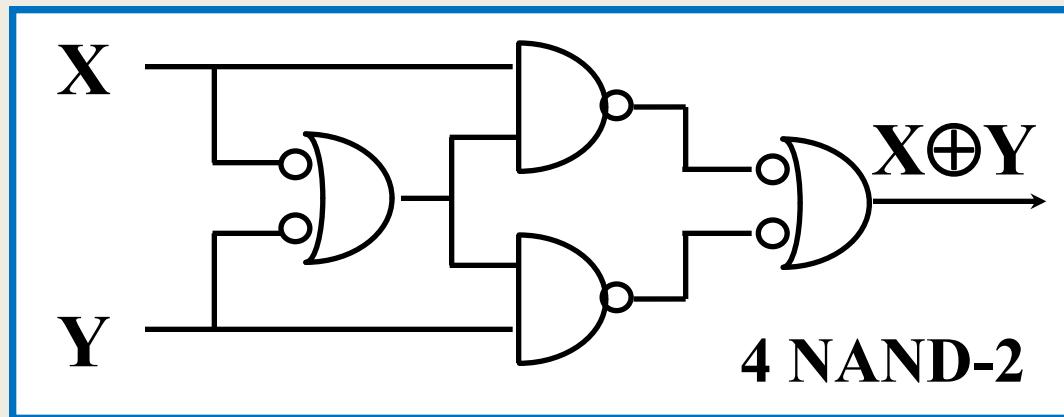


Υλοποίηση XOR με NAND

- Εξισώσεις Boole των XOR και XNOR

- $X \oplus Y = \bar{X}Y + X\bar{Y}$
- $\overline{X \oplus Y} = \bar{X}\bar{Y} + XY$

- Υλοποίηση πύλης XOR με 4 πύλες NAND δύο εισόδων σε 3 επίπεδα
- Σχηματικό διάγραμμα



- Αλγεβρική απόδειξη

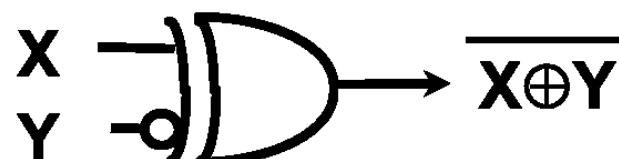
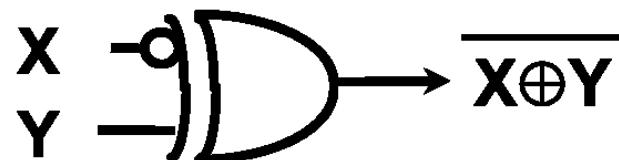
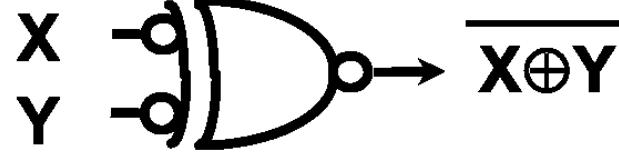
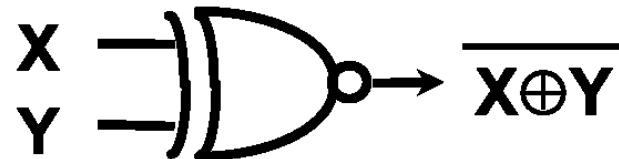
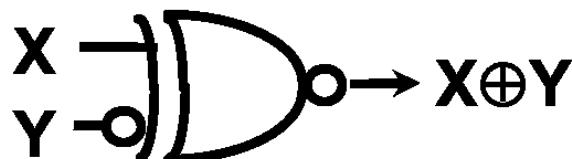
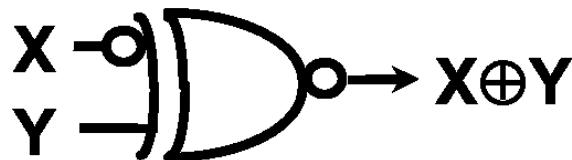
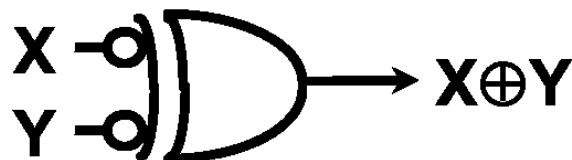
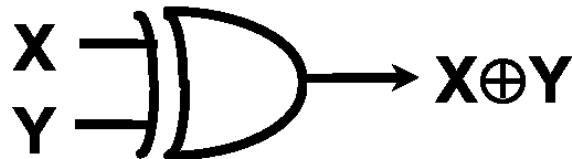
$$X(\bar{X} + \bar{Y}) + Y(\bar{X} + \bar{Y}) = X\bar{X} + X\bar{Y} + Y\bar{X} + Y\bar{Y} = \bar{X}Y + X\bar{Y} = X \oplus Y$$

Αλγεβρικές Ισότητες για XOR/XNOR

- $X \oplus Y = Y \oplus X$
- $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$
- $X \oplus 0 = X$
- $X \oplus 1 = \bar{X}$
- $X \oplus X = 0$
- $X \oplus \bar{X} = 1$
- $\bar{X} \oplus \bar{Y} = X \oplus Y$
- $\bar{X} \oplus Y = \overline{X \oplus Y}$
- $X \oplus \bar{Y} = \overline{X \oplus Y}$

Συμβολισμοί πυλών XOR και XNOR

- Ισοδύναμοι συμβολισμοί XOR και XNOR (για ώθηση φυσαλίδων)



Επιλεγμένες ασκήσεις

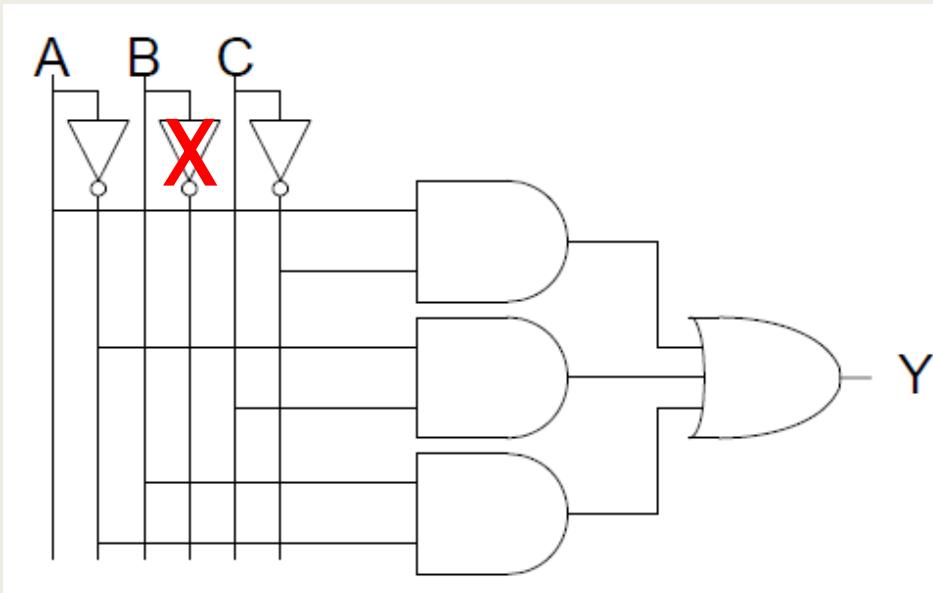
■ Ασκήσεις 2.10

Με βάση την απλοποιημένη εξίσωση Boole της Άσκησης 2.6 – (β)

$$Y = A\bar{C} + \bar{A}C + \bar{A}B$$

να σχεδιάσετε με πύλες NOT στις εισόδους, όπου απαιτείται:

- Την υλοποίηση AND-OR δύο επιπέδων



Επιλεγμένες ασκήσεις

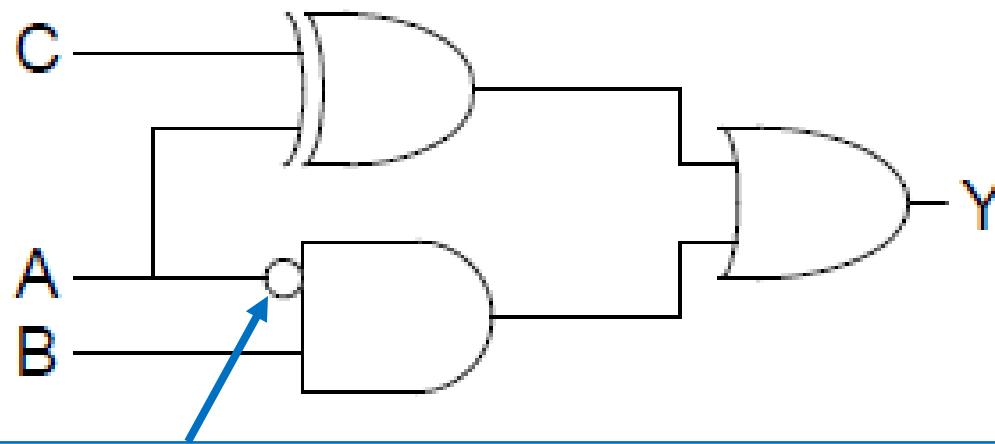
■ Ασκήσεις 2.8

Με βάση την απλοποιημένη εξίσωση Boole της Άσκησης 2.6 – (β)

$$Y = A\bar{C} + \bar{A}C + \bar{A}B$$

να σχεδιάσετε με πύλες NOT στις εισόδους, όπου απαιτείται:

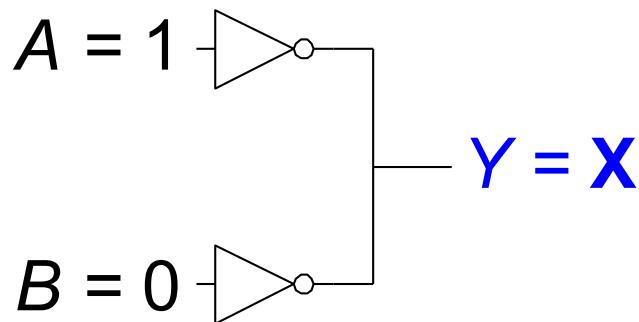
- Την υλοποίηση που χρησιμοποιεί και πύλη XOR για μείωση υλικού



Ενσωμάτωση της πύλης NOT στην είσοδο της πύλης AND

Μη αποδεκτή τιμή: X

- Η áλγεβρα Boole περιορίζεται στη χρήση των τιμών 0 και 1
- Ωστόσο, τα κυκλώματα στον πραγματικό κόσμο μπορούν επίσης να έχουν **μη αποδεκτές** τιμές
- Το σύμβολο **X** υποδεικνύει ότι ο κόμβος του κυκλώματος έχει **άγνωστη ή μη αποδεκτή τιμή**
- Αυτό συνήθως συμβαίνει αν οδηγείται ταυτόχρονα και στην τιμή 0 και στην τιμή 1
 - Αυτή η κατάσταση στον κόμβο, που ονομάζεται **ανταγωνισμός**, θεωρείται εσφαλμένη σχεδίαση και πρέπει να αποφεύγεται



Μη αποδεκτή τιμή: X

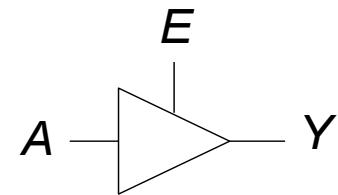
- Η πραγματική τάση σε έναν κόμβο με ανταγωνισμό μπορεί να έχει κάποια **τιμή μεταξύ 0 και V_{DD}**
 - Συχνά, αλλά όχι πάντα, έχει τιμή που εμπίπτει στην **απαγορευμένη ζώνη**
- Ο **ανταγωνισμός** μπορεί επίσης να προκαλέσει την κατανάλωση **μεγάλων ποσοτήτων ισχύος** μεταξύ των αντιμαχόμενων πυλών, με αποτέλεσμα το κύκλωμα να **υπερθερμανθεί** και ίσως να **καταστραφεί**
- Οι τιμές X χρησιμοποιούνται από τους προσομοιωτές κυκλωμάτων για να υποδεικνύουν **αρχικές τιμές που δεν έχουν καθοριστεί**
- **Προσοχή:**
 - *Tα X χρησιμοποιούνται για να δείξουν και τους αδιάφορους όρους στους πίνακες αλήθειας αλλά και τον ανταγωνισμό στους κόμβους*

Μετέωρη τιμή: Z

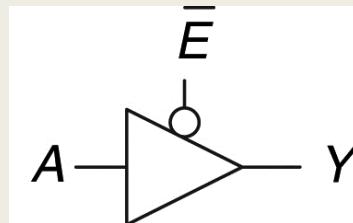
- Το σύμβολο Z υποδεικνύει ότι ένας κόμβος δεν οδηγείται ούτε σε τάση HIGH ούτε σε τάση LOW. Τότε λέμε ότι κόμβος είναι «**μετέωρος**», έχει **υψηλή αντίσταση**, ή **υψηλό Z**
- Ένας μετέωρος κόμβος μπορεί να έχει τιμή 0, τιμή 1, ή κάποια τάση στο ενδιάμεσο, ανάλογα με το ιστορικό της λειτουργίας του κυκλώματος
- Η ύπαρξη ενός μετέωρου κόμβου δεν σημαίνει πάντα ότι υπάρχει σφάλμα στο κύκλωμα
 - για παράδειγμα, όταν υπάρχει κάποιο άλλο στοιχείο που οδηγεί τον ίδιο κόμβο στην τιμή 0 ή την τιμή 1
- Τρόποι δημιουργίας ενός μετέωρου κόμβου:
 - όταν **ξεχνάμε να συνδέσουμε μια τάση σε μια είσοδο ενός κυκλώματος**
 - όταν **υποθέτουμε ότι μια μη συνδεδεμένη είσοδος είναι ίδια με μία είσοδο που έχει την τιμή 0 (λάθος)**

Απομονωτής τριών καταστάσεων

- Ο απομονωτής τριών καταστάσεων (tristate buffer), διαθέτει τρεις πιθανές καταστάσεις εξόδου: HIGH (1), LOW (0), και μετέωρη (Z).
- Ο απομονωτής αριστερά έχει μια είσοδο A , την έξοδο Y και ένα σήμα **enable** (έγκρισης) E **ενεργό στο HIGH (active high)**
- Ο απομονωτής δεξιά έχει μια είσοδο A , την έξοδο Y και ένα σήμα **enable** (έγκρισης) E **ενεργό στο LOW (active low)**



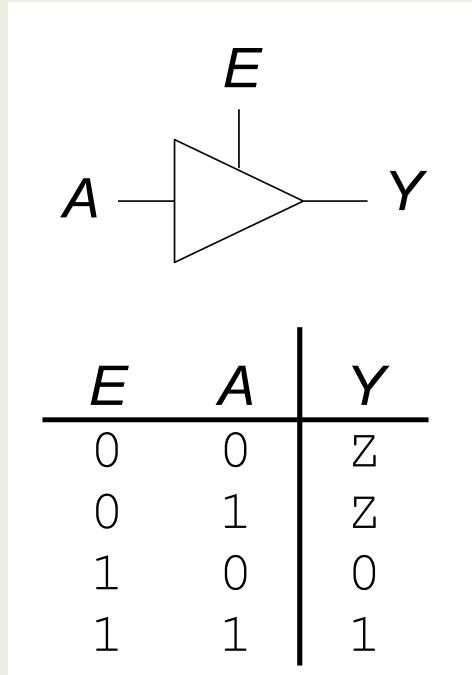
E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1



\bar{E}	A	Y
0	0	0
0	1	1
1	0	Z
1	1	Z

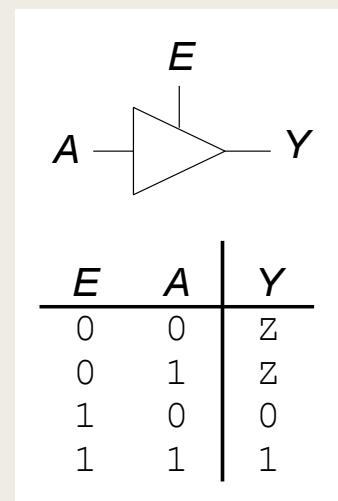
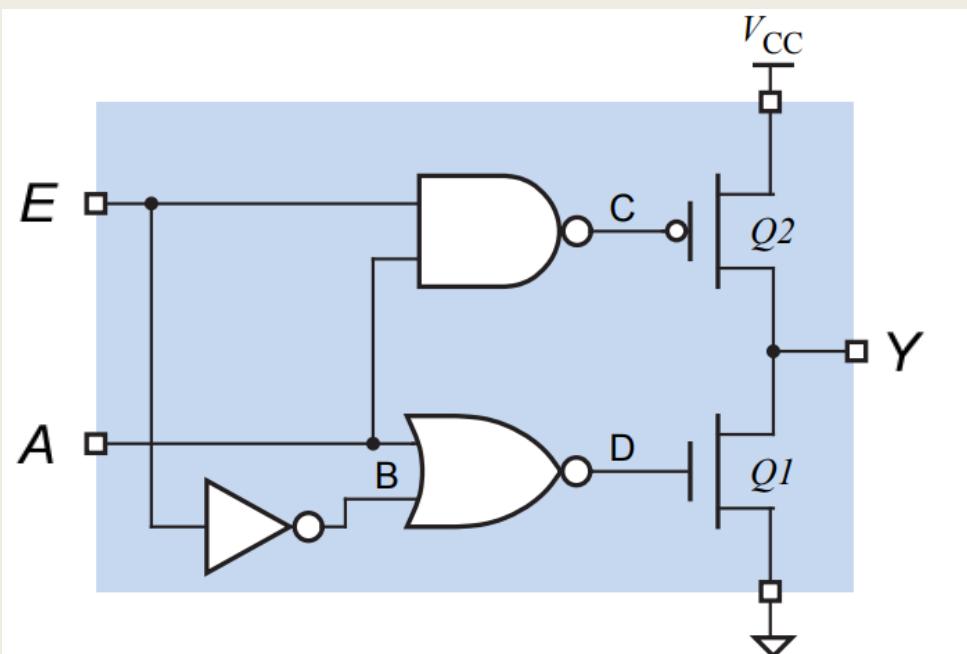
Απομονωτής τριών καταστάσεων

- Στον απομονωτή τριών καταστάσεων (tri-state buffer) όταν το **σήμα enable** **έχει τιμή '1' (TRUE)**, επιτρέπεται η μεταφορά τιμής από την είσοδο στην έξοδο, οπότε η έξοδος έχει την τιμή της εισόδου
 - ο απομονωτής τριών καταστάσεων **συμπεριφέρεται ως απλώς απομονωτής**
- Όταν το **σήμα enable** **έχει τιμή '0' (FALSE)**, δεν επιτρέπεται η μεταφορά τιμής από την είσοδο στην έξοδο, οπότε η έξοδος έχει υψηλή αντίσταση ή είναι μετέωρη (Z)
 - ο απομονωτής τριών καταστάσεων **συμπεριφέρεται ως πυκνωτής**



Απομονωτής τριών καταστάσεων

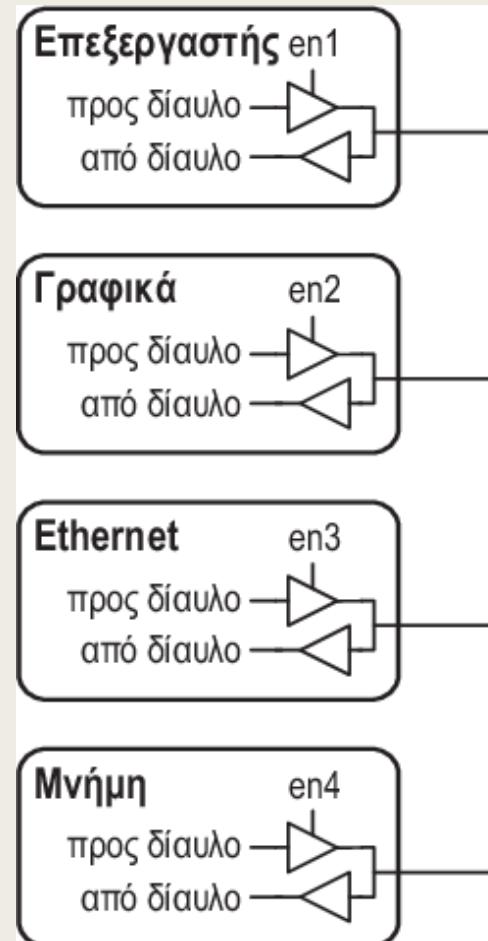
- Απλουστευμένο διάγραμμα κυκλώματος για έναν **απομονωτή τριών καταστάσεων** (tri-state buffer) της λογικής οικογένειας CMOS
 - Οι εσωτερικές πύλες *NAND*, *NOR* και ο αντιστροφέας απεικονίζονται με τα σύμβολά τους και όχι με τρανζίστορ
- Όταν το **σήμα enable έχει τιμή '0'**, τα δύο τρανζίστορα της εξόδου δεν άγουν και η έξοδος βρίσκεται στη μετέωρη κατάσταση *Z*
- Όταν το **σήμα enable έχει τιμή '1'**, η έξοδος καθορίζεται από την είσοδο *A*



Διάγραμμα από το βιβλίο «Ψηφιακή Σχεδίαση: Αρχές και Πρακτικές» του John F. Wakerly

Παράδειγμα χρήσης του απομονωτή

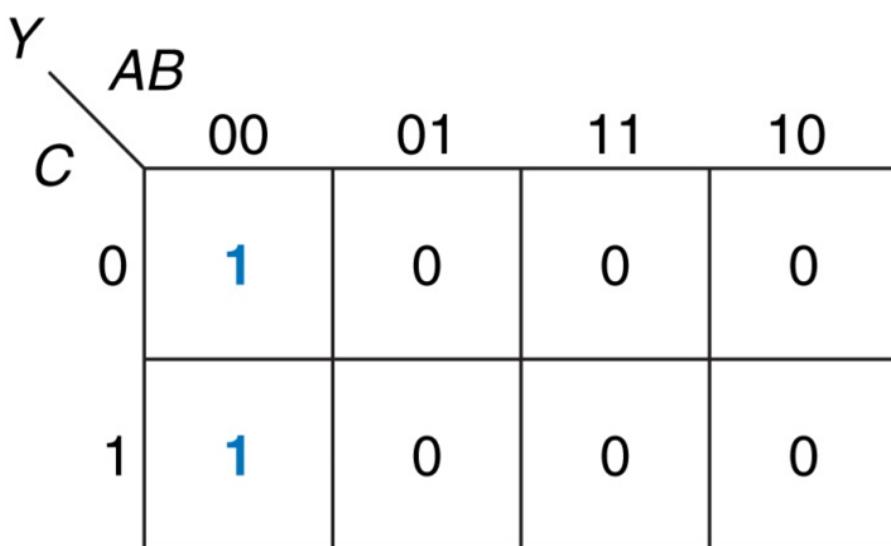
- Οι απομονωτές τριών καταστάσεων συνήθως χρησιμοποιούνται σε **διαύλους (buses)** που συνδέουν πολλά τσιπ
- Κάθε τσιπ μπορεί να συνδέεται σε έναν κοινόχρηστο δίαυλο μνήμης χρησιμοποιώντας απομονωτές τριών καταστάσεων
- Μόνο ένα τσιπ τη φορά επιτρέπεται να ενεργοποιεί το σήμα enable για να οδηγεί μια τιμή στον δίαυλο.
- Τα υπόλοιπα τσιπ πρέπει να παράγουν μετέωρους εξόδους, έτσι ώστε να μην προκαλούν ανταγωνισμό με το τσιπ που «συνομιλεί» με τη μνήμη



Χάρτες Karnaugh (K-maps)

- Οι χάρτες Karnaugh (Karnaugh maps, K-maps) αποτελούν μια οπτικοποιημένη (γραφική) μέθοδο για την ελαχιστοποίηση εξισώσεων Boole σε κανονική μορφή αθροίσματος γινομένων μέχρι 4 μεταβλητές (εισόδους)
- Η λειτουργική προδιαγραφή ενός συνδυαστικού κυκλώματος εκφράζεται και με τη μορφή του χάρτη Karnaugh
- Τους επινόησε το 1953 ο Maurice Karnaugh

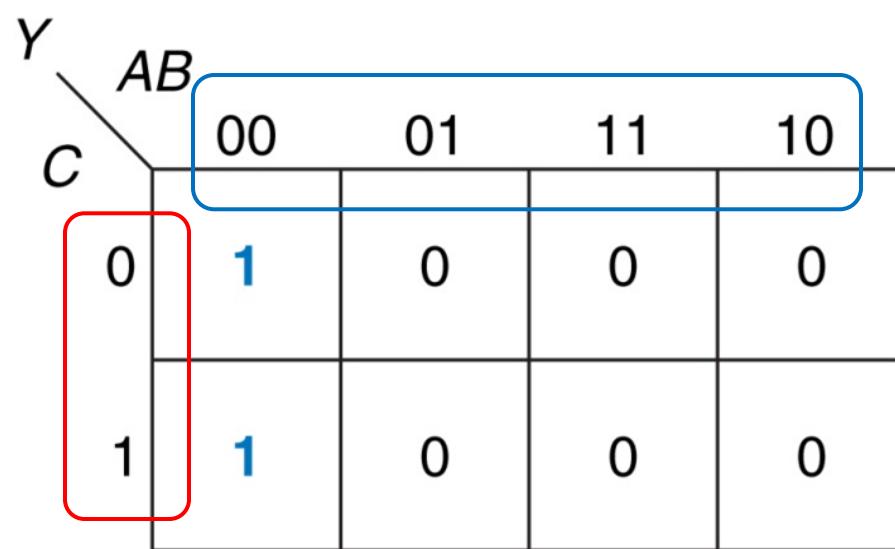
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



Χάρτες Karnaugh των 3 εισόδων

- Η πάνω γραμμή του χάρτη περιέχει τις τέσσερις πιθανές τιμές για τις εισόδους A και B . Οι τιμές των εισόδων A και B είναι:
 - **00, 01, 11, 10 (κώδικας Gray)** και **όχι 00, 01, 10, 11** ώστε δύο γειτονικά τετράγωνα να διαφέρουν στην τιμή μίας μόνο εισόδου
- Η αριστερή στήλη περιέχει τις δύο πιθανές τιμές για την είσοδο C

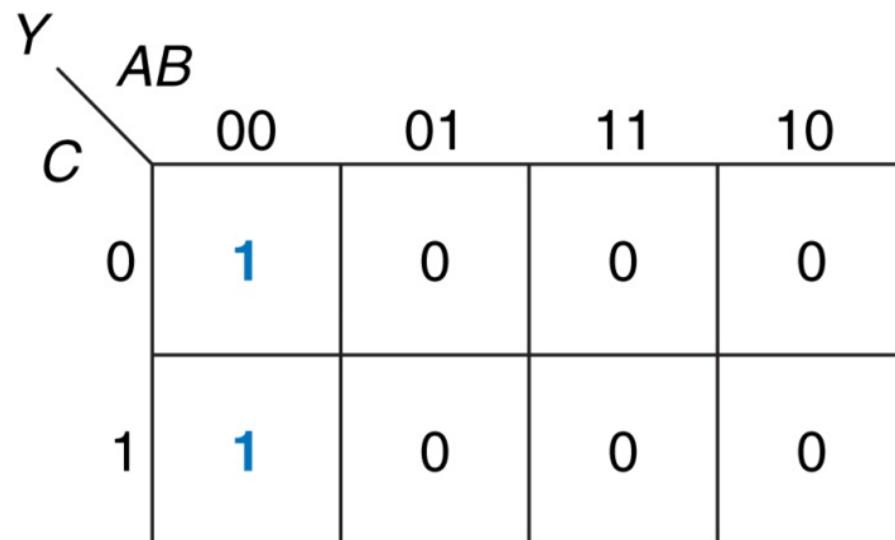
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



Χάρτες Karnaugh των 3 εισόδων

- Κάθε τετράγωνο του χάρτη αντιστοιχεί σε μια γραμμή του πίνακα αληθείας και περιέχει την τιμή της εξόδου Y για τη συγκεκριμένη γραμμή, ανάλογα με την τιμή των εισόδων
- Κάθε δύο γειτονικά τετράγωνα διαφέρουν στην τιμή μίας μόνο εισόδου. Οι άλλες δύο είσοδοι έχουν την ίδια τιμή

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



Χάρτες Karnaugh των 3 εισόδων

- Ακριβώς όπως ισχύει για τις γραμμές ενός πίνακα αληθείας, έτσι και κάθε τετράγωνο ενός χάρτη Karnaugh αναπαριστά έναν **ελαχιστόρο**
- Κάθε δύο γειτονικά τετράγωνα έχουν τα ίδια λεκτικά εκτός από ένα, το οποίο εμφανίζεται με την αληθινή μορφή του στο ένα τετράγωνο και με τη συμπληρωματική μορφή του στο άλλο
 - *Για παράδειγμα στην πρώτη στήλη τα τετράγωνα που αναπαριστούν τους ελαχιστούς $\bar{A}\bar{B}\bar{C}$ και $\bar{A}\bar{B}C$ είναι γειτονικά και διαφέρουν μόνο ως προς τη μεταβλητή C*

		Y	AB	C	
		00	01	11	10
0	0	1	0	0	0
	1	1	0	0	0

		Y	AB	C	
		00	01	11	10
0	0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$A\bar{B}\bar{C}$	$A\bar{B}C$
	1	$\bar{A}B\bar{C}$	$\bar{A}BC$	ABC	$A\bar{B}C$

Αναδίπλωση του χάρτη Karnaugh

- Ένα άλλο χαρακτηριστικό του χάρτη Karnaugh είναι ότι «**αναδιπλώνεται**»
- Τα τετράγωνα που βρίσκονται **εντελώς δεξιά** είναι ουσιαστικά **γειτονικά** με τα τετράγωνα που βρίσκονται **εντελώς αριστερά**, με την έννοια ότι διαφέρουν μόνο ως προς μία μεταβλητή
 - Για παράδειγμα στην πρώτη σειρά τα τετράγωνα που αναπαριστούν τους ελαχιστόρους $\bar{A}\bar{B}\bar{C}$ και $A\bar{B}\bar{C}$ είναι γειτονικά και διαφέρουν μόνο ως προς τη μεταβλητή **A**

		Y	AB			
		C	00	01	11	10
0	0	1	0	0	0	0
	1	1	0	0	0	0

		Y	AB			
		C	00	01	11	10
0	0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}\bar{C}$	$A\bar{B}\bar{C}$	$A\bar{B}\bar{C}$	
	1	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	ABC	$A\bar{B}C$	

Ελαχιστοποίηση εξισώσεων Boole

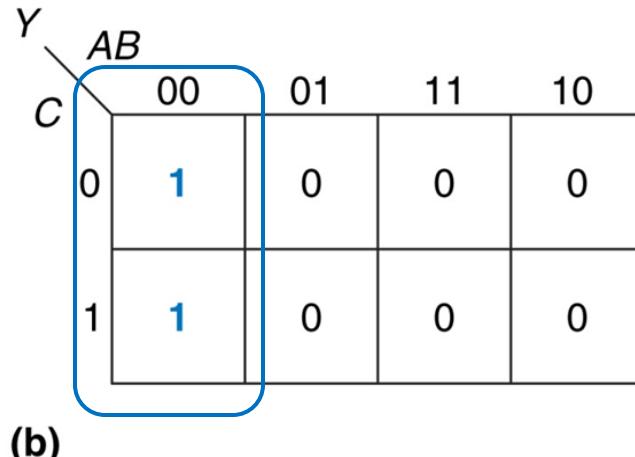
- Η ελαχιστοποίηση λογικών εξισώσεων Boole σε κανονική μορφή αθροίσματος γινομένων περιλαμβάνει τον **συνδυασμό δύο ελαχιστόρων** για τους οποίους η έξοδος έχει την τιμή 1, οι οποίοι περιέχουν ένα **γινόμενο λεκτικών P**, καθώς και την **αληθινή και τη συμπληρωματική μορφή κάποιας μεταβλητής, έστω A**
- Όταν συνδυαστούν οι 2 ελαχιστόροι εξαλείφεται η μεταβλητή A, ενώ παραμένει το **γινόμενο λεκτικών P**
 - $P \cdot A + P \cdot \bar{A} = P \cdot (A + \bar{A}) = P \cdot 1 = P$

Χάρτες Karnaugh: Ελαχιστοποίηση

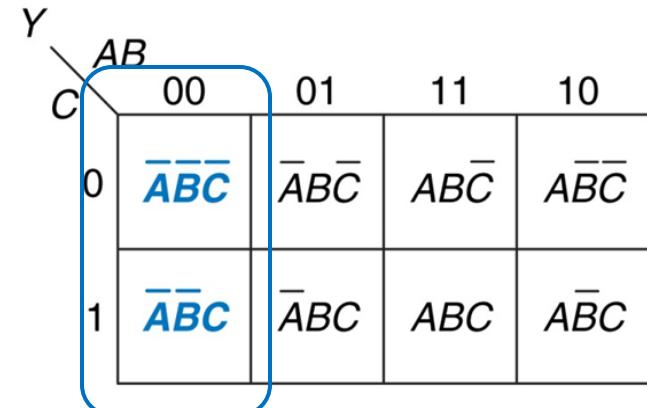
- Η εύρεση των εξισώσεων από τον χάρτη Karnaugh είναι ακριβώς ισοδύναμη με την εύρεση των εξισώσεων σε μορφή αθροίσματος γινομένων απευθείας από τον πίνακα αληθείας.
- Στον χάρτη Karnaugh του σχήματος υπάρχουν μόνο δύο ελαχιστόροι στην εξίσωση: $\bar{A}\bar{B}\bar{C}$ και $\bar{A}\bar{B}C$ για τους οποίους η έξοδος έχει την τιμή 1
- $Y = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C$

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(a)



(b)



(c)

Χάρτες Karnaugh: Ελαχιστοποίηση

- Αν ελαχιστοποιήσουμε αυτήν την εξίσωση με τη βοήθεια της άλγεβρας Boole έχουμε:

$$Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C = \overline{A}\overline{B}(\overline{C} + C) = \overline{A}\overline{B}(1) = \overline{A}\overline{B}$$

- Οι χάρτες Karnaugh μας βοηθούν να κάνουμε αυτή ακριβώς την ελαχιστοποίηση γραφικά, **κυκλώνοντας τους άσους σε δύο γειτονικά τετράγωνα που έχουν την τιμή 1**

- Για κάθε κύκλο, γράφουμε τον αντίστοιχο όρο στην εξίσωση
- Μεταβλητές των οποίων η αληθινή και η συμπληρωματική μορφή περιέχονται και οι δύο στον κύκλο εξαιρούνται από τον όρο
- Δηλαδή στον όρο κρατούμε τις μεταβλητές που είναι κοινές και στα δύο γειτονικά τετράγωνα.

Επομένως:

$$Y = \overline{A}\overline{B}$$

		AB	00	01	11	10
		C	0	1	0	0
Y	AB	0	1	0	0	0
		1	1	0	0	0

Ελαχιστοποίηση εξισώσεων Boole

- Η ελαχιστοποίηση λογικών εξισώσεων Boole σε κανονική μορφή αθροίσματος γινομένων **γενικεύεται**, ώστε να περιλαμβάνει τον **συνδυασμό τεσσάρων ελαχιστόρων** για τους οποίους η έξοδος έχει την τιμή 1, οι οποίοι περιέχουν ένα **γινόμενο λεκτικών P**, καθώς και έναν από τους τέσσερεις πιθανούς συνδυασμούς της **αληθινής και της συμπληρωματικής μορφής κάποιων δύο μεταβλητών**, έστω A και B. ήτοι $\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$, AB
- Όταν συνδυαστούν οι 4 ελαχιστόροι εξαλείφονται οι μεταβλητές A και B, ενώ παραμένει το **γινόμενο λεκτικών P**
 - $P \cdot \bar{A}\bar{B} + P \cdot \bar{A}B + P \cdot A\bar{B} + P \cdot AB =$
 $P \cdot (\bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB) =$
 $P \cdot (\bar{A}(\bar{B} + B) + A(\bar{B} + B)) =$
 $P \cdot (\bar{A}(1) + A(1)) = P \cdot (\bar{A} + A) = P \cdot 1 = P$
- Αντίστοιχη και η περίπτωση των 8 συνδυάσιμων ελαχιστόρων, όπου εξαλείφονται 3 μεταβλητές

Χάρτες Karnaugh: Ελαχιστοποίηση

- Στον χάρτη Karnaugh του σχήματος αριστερά υπάρχουν 4 ελαχιστόροι στην εξίσωση: $\bar{A}\bar{B}\bar{C}$, $\bar{A}\bar{B}C$, $\bar{A}B\bar{C}$, και $\bar{A}BC$ για τους οποίους η έξοδος έχει την τιμή 1
- $Y = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC$
- Στον χάρτη Karnaugh του σχήματος δεξιά υπάρχουν 4 ελαχιστόροι στην εξίσωση: $A\bar{B}\bar{C}$, $\bar{A}B\bar{C}$, $AB\bar{C}$ και $A\bar{B}C$ για τους οποίους η έξοδος έχει την τιμή 1
- $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}C$

		AB	00	01	11	10	
		C	0	1	1	0	0
Y	AB	0	1	1	0	0	
		1	1	1	0	0	

		AB	00	01	11	10
		C	0	1	1	1
Y	AB	0	1	1	1	1
		1	0	0	0	0

Χάρτες Karnaugh: Ελαχιστοποίηση

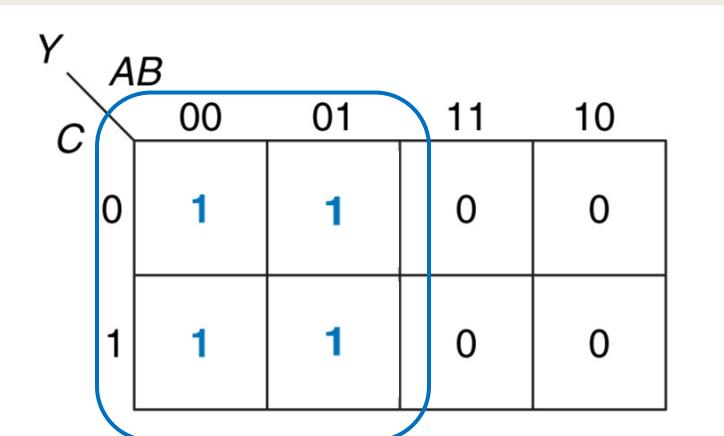
- Αν ελαχιστοποιήσουμε αυτές τις εξισώσεις με την άλγεβρα Boole έχουμε:

$$Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC = \overline{A} \cdot (\overline{B}\overline{C} + \overline{B}C + B\overline{C} + BC) = \overline{A} \quad (\text{αριστερά})$$

$$Y = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C = \overline{C} \cdot (\overline{A}\overline{B} + \overline{A}B + A\overline{B} + AB) = \overline{C} \quad (\text{δεξιά})$$

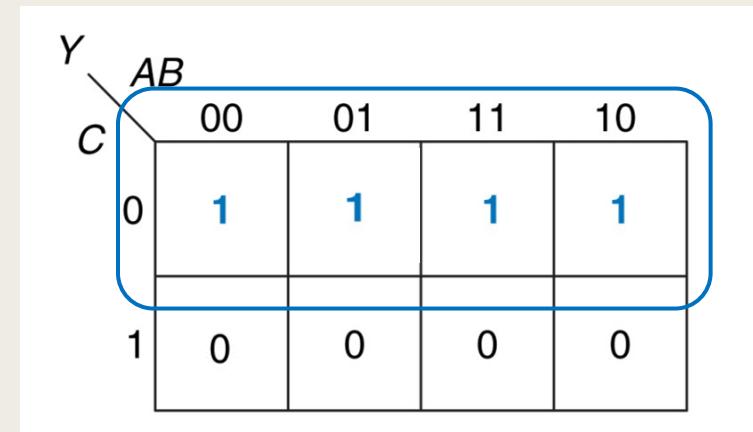
- Οι χάρτες Karnaugh μας βοηθούν να κάνουμε την ελαχιστοποίηση γραφικά, **κυκλώνοντας τους άσους σε 4 γειτονικά τετράγωνα που έχουν την τιμή 1**

- Τα 4 γειτονικά τετράγωνα σχηματίζουν ένα τετράγωνο (αριστερά) ή απαρτίζουν μία σειρά (δεξιά) (ή μία στήλη) του χάρτη Karnaugh
- Για κάθε κύκλο, γράφουμε τον αντίστοιχο όρο στην εξίσωση
- Στον όρο κρατούμε τις μεταβλητές που είναι κοινές και στα 4 γειτονικά τετράγωνα. Επομένως: $Y = \overline{A}$ (αριστερά) και $Y = \overline{C}$ (δεξιά)



A Karnaugh map for three variables (A, B, C) showing the function $Y = \overline{A} + C$. The columns are labeled AB (00, 01, 11, 10) and the rows are labeled C (0, 1). The output Y is 1 for cells (00, 01, 11) and 0 for cells (10, 11, 10). A blue rounded rectangle highlights the top-left 3x2 block (00, 01, 11).

		AB	00	01	11	10
		C	1	1	0	0
		0	1	1	1	0
		1	0	0	0	0



A Karnaugh map for three variables (A, B, C) showing the function $Y = \overline{A} + C$. The columns are labeled AB (00, 01, 11, 10) and the rows are labeled C (0, 1). The output Y is 1 for cells (00, 01, 11) and 0 for cells (10, 11, 10). A blue rounded rectangle highlights the top row (00, 01, 11, 10).

		AB	00	01	11	10
		C	1	1	1	1
		0	1	1	1	1
		1	0	0	0	0

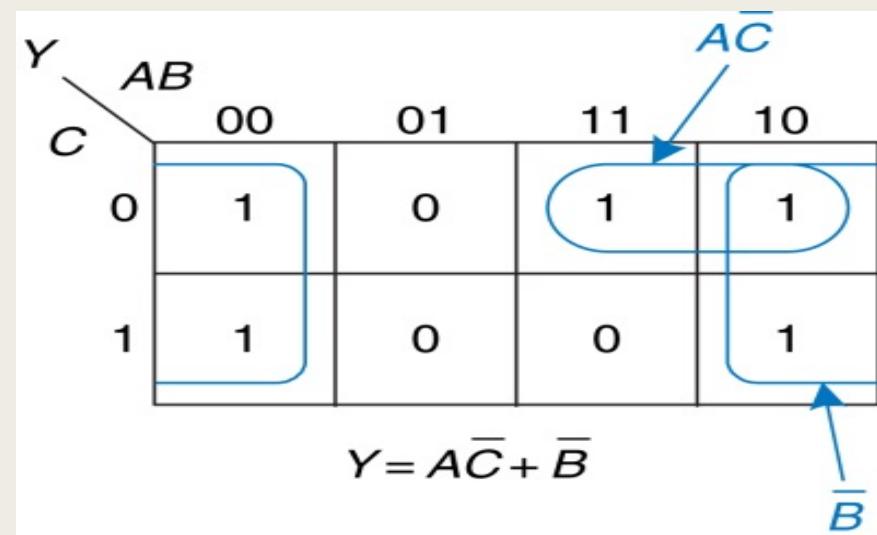
Διαδικασία ελαχιστοποίησης με K-map

- Χρησιμοποιήστε το **ελάχιστο πλήθος κύκλων** που είναι απαραίτητο για την κάλυψη όλων των τετραγώνων με τιμή 1
- Όλα τα τετράγωνα μέσα σε κάθε κύκλο πρέπει να περιέχουν την τιμή 1
- Κάθε κύκλος πρέπει να καλύπτει ένα ορθογώνιο τμήμα το οποίο να περιέχει ένα **πλήθος γειτονικών τετραγώνων που αποτελεί δύναμη του δύο** (δηλαδή 1, 2, 4 ή 8) προς κάθε κατεύθυνση
- Κάθε κύκλος πρέπει να έχει **το μεγαλύτερο δυνατό μέγεθος** έτσι ώστε να αναπαριστά **πρώτους όρους**
- Ένας κύκλος μπορεί να «**αναδιπλώνεται**» γύρω από τα άκρα ενός χάρτη Karnaugh.
- Ένα τετράγωνο με τιμή 1 στον χάρτη Karnaugh μπορεί να **κυκλωθεί πολλές φορές**, αν αυτό έχει ως αποτέλεσμα ότι θα χρησιμοποιηθούν **λιγότεροι κύκλοι**

Παράδειγμα 2.9

- Ελαχιστοποίηση μίας συνάρτησης τριών μεταβλητών με χρήση K-map
- Έστω ότι έχουμε τη συνάρτηση $Y = F(A, B, C)$ του K-map που φαίνεται κάτω αριστερά. Ελαχιστοποιήστε την εξίσωση χρησιμοποιώντας τον χάρτη.
 - Κυκλώνουμε τα τετράγωνα με τιμή 1 στον χάρτη Karnaugh χρησιμοποιώντας όσο το δυνατόν λιγότερους κύκλους
 - Κάθε κύκλος στον χάρτη αναπαριστά έναν πρώτο όρο, και η διάσταση κάθε κύκλου αποτελεί δύναμη του δύο

	AB		
	C	00	01
0		1	0
1		1	0



Χάρτες Karnaugh των 4 εισόδων

- Η πάνω γραμμή του χάρτη περιέχει τις τέσσερις πιθανές τιμές για τις εισόδους A και B . Οι τιμές των εισόδων A και B είναι:
 - **00, 01, 11, 10 (κώδικας Gray)**
- Η αριστερή στήλη του χάρτη περιέχει τις τέσσερις πιθανές τιμές για τις εισόδους C και D . Οι τιμές των εισόδων C και D επίσης είναι:
 - **00, 01, 11, 10 (κώδικας Gray)**

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Στήλη
 $AB=00$

Στήλη
 $AB=01$

Στήλη
 $AB=10$

Στήλη
 $AB=11$

Y	AB	00	01	11	10
CD	00	1	0	0	1
	01	0	1	0	1
	11	1	1	0	0
	10	1	1	0	1

Χάρτες Karnaugh των 4 εισόδων

- Κάθε τετράγωνο με τιμή 1 αντιστοιχεί σε έναν ελαχιστόρο με 4 λεκτικά. Είναι πρώτος όρος, εάν όλα τα γειτονικά του τετράγωνα έχουν τιμή 0, (όπως για παράδειγμα η πύλη XOR των 4 εισόδων)
- Κάθε 2 γειτονικά τετράγωνα με τιμή 1 αντιστοιχούν σε έναν όρο με 3 λεκτικά
- Κάθε 4 γειτονικά τετράγωνα με τιμή 1 αντιστοιχούν σε έναν όρο με 2 λεκτικά
- Κάθε 8 γειτονικά τετράγωνα με τιμή 1 αντιστοιχούν σε έναν όρο με 1 λεκτικό
- Κάθε 16 γειτονικά τετράγωνα με τιμή 1 αντιστοιχούν στη συνάρτηση $F = 1$

		Y	AB	CD		
		00	01	11	10	
		00	1	0	0	1
		01	0	1	0	1
		11	1	1	0	0
		10	1	1	0	1

$$Y = \bar{A}\bar{C} + \bar{A}BD + A\bar{B}\bar{C} + \bar{B}\bar{D}$$

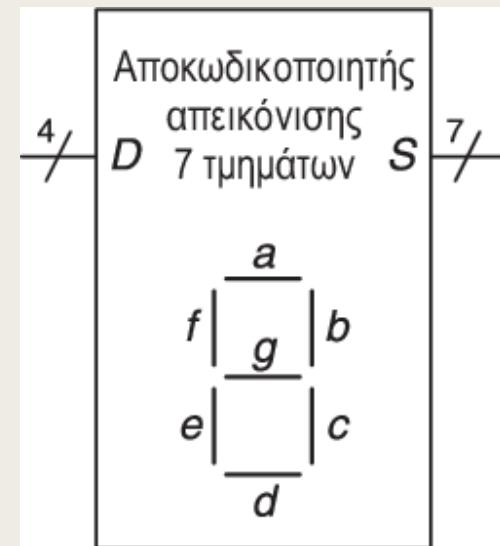
Δυαδικοί κώδικες για δεκαδικούς αριθμούς

- Ο **κώδικας BCD (Binary-Coded Decimal)** έχει 4 δυαδικά ψηφία και χρησιμοποιείται για την κωδικοποίηση των μονοψήφιων δεκαδικών αριθμών από το 0 μέχρι το 9.
 - Τα βάρη στον κώδικα BCD είναι 8-4-2-1, όπως των δυαδικών αριθμών
 - Στην πράξη χρησιμοποιείται όπου χειριζόμαστε δεκαδικούς αριθμούς,
 - Στους αποκωδικοποιητές απεικόνισης επτά τμημάτων
 - Στην αναπαράσταση του διψήφιου δεκαδικού κλασματικού μέρους με ένα byte στις χρηματοοικονομικές εφαρμογές
- Παράδειγμα: Το 37_{10} γράφεται ως 00110111_{BCD}

Παράδειγμα 2.10

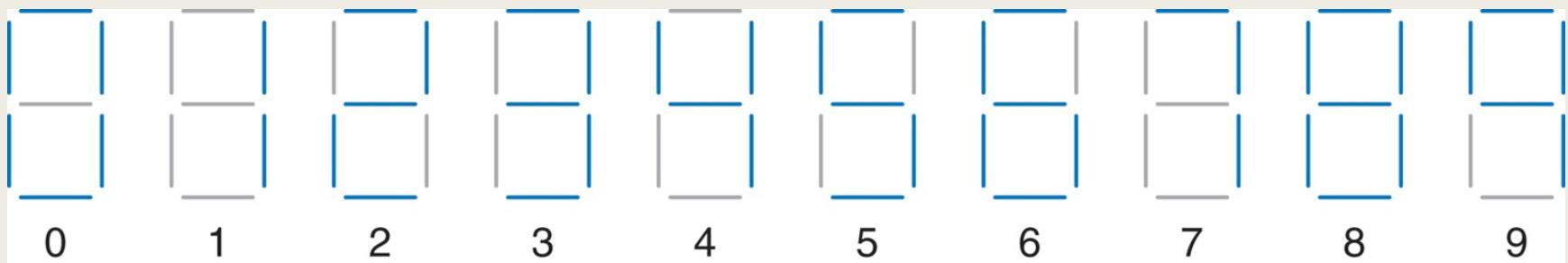
■ Αποκωδικοποιητής απεικόνισης επτά τμημάτων

- Δέχεται ένα δίαυλο δεδομένων $D_{3:0}$ των 4 bit στον κώδικα BCD
- Παράγει ένα δίαυλο σημάτων ελέγχου $S_a - S_g$ των 7 bit που ελέγχουν τα **επτά τμήματα a έως g** που σχηματίζουν ένα **δεκαδικό ψηφίο (0 – 9)**
- Κάθε τμήμα υλοποιείται με ένα LED. Εάν το αντίστοιχο σήμα ελέγχου έχει την τιμή 1 το LED ανάβει, αλλιώς είναι σβηστό
- Συμπληρώστε τον πίνακα αληθείας
- Χρησιμοποιήστε χάρτες Karnaugh προκειμένου να βρείτε τις ελαχιστοποιημένες εξισώσεις Boole για τις εξόδους S_a και S_b .
- Υποθέστε ότι όλα τα τμήματα παραμένουν «σβηστά» για τις μη αποδεκτές τιμές εισόδου (10–15).



Παράδειγμα 2.10

- Αποκωδικοποιητής απεικόνισης επτά τμημάτων
- Σχηματισμός ενός δεκαδικού αριθμού με επτά τμήματα



Παράδειγμα 2.10

- Αποκωδικοποιητής απεικόνισης επτά τμημάτων
- Πίνακας αληθείας

$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
άλλες	0	0	0	0	0	0	0

Παράδειγμα 2.10

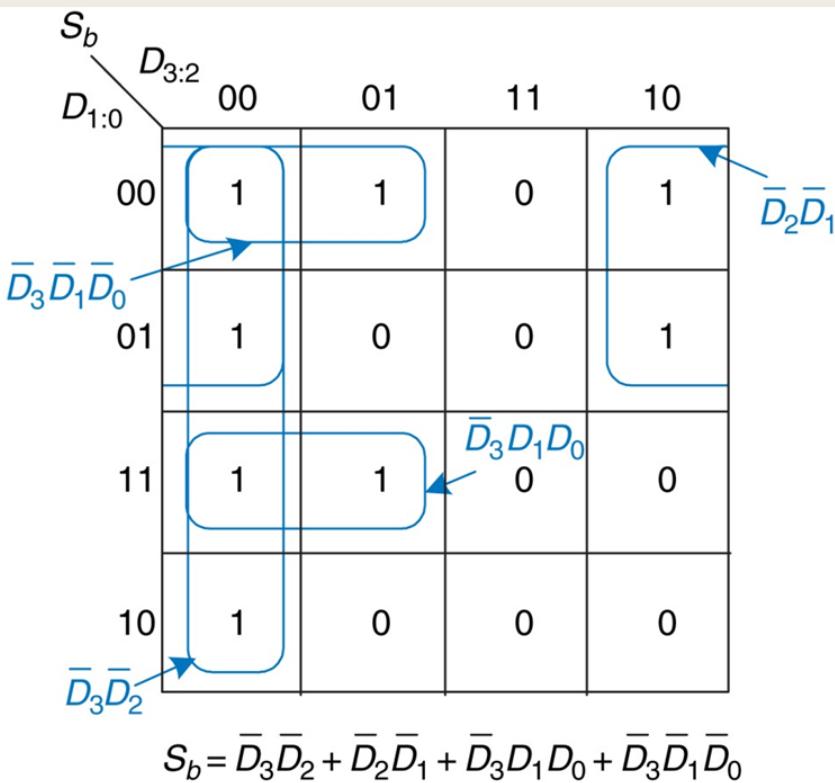
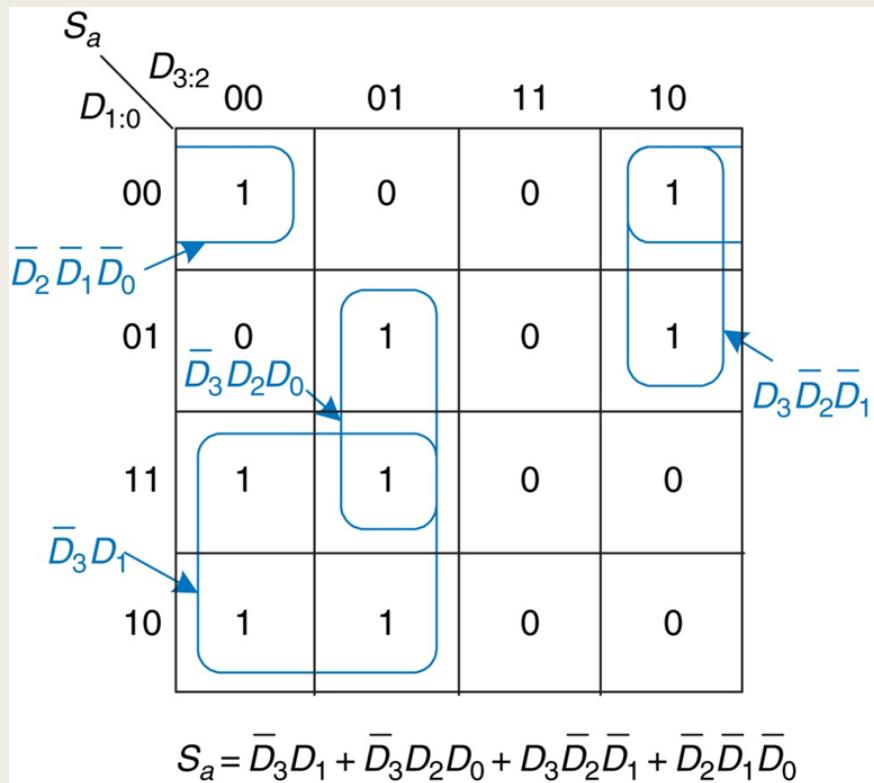
- Αποκωδικοποιητής απεικόνισης επτά τμημάτων
- Χάρτες Karnaugh για τις εξόδους S_a και S_b

S_a	$D_{3:2}$	00	01	11	10
$D_{1:0}$	00	1	0	0	1
01	0	1	0	1	
11	1	1	0	0	
10	1	1	0	0	

S_b	$D_{3:2}$	00	01	11	10
$D_{1:0}$	00	1	1	0	1
01	1	0	0	1	
11	1	1	0	0	
10	1	0	0	0	

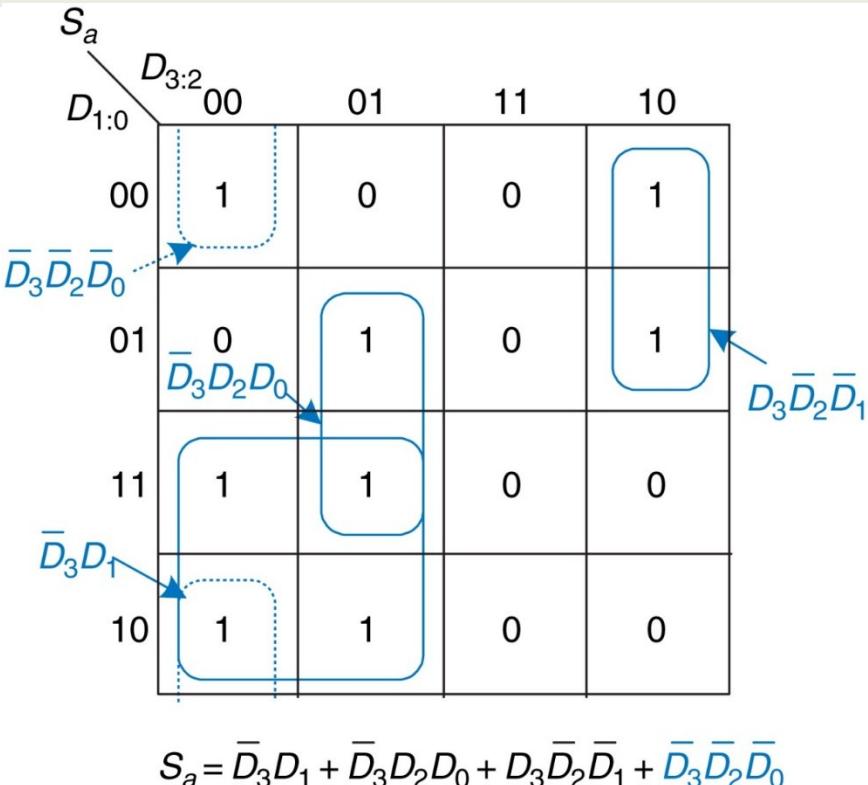
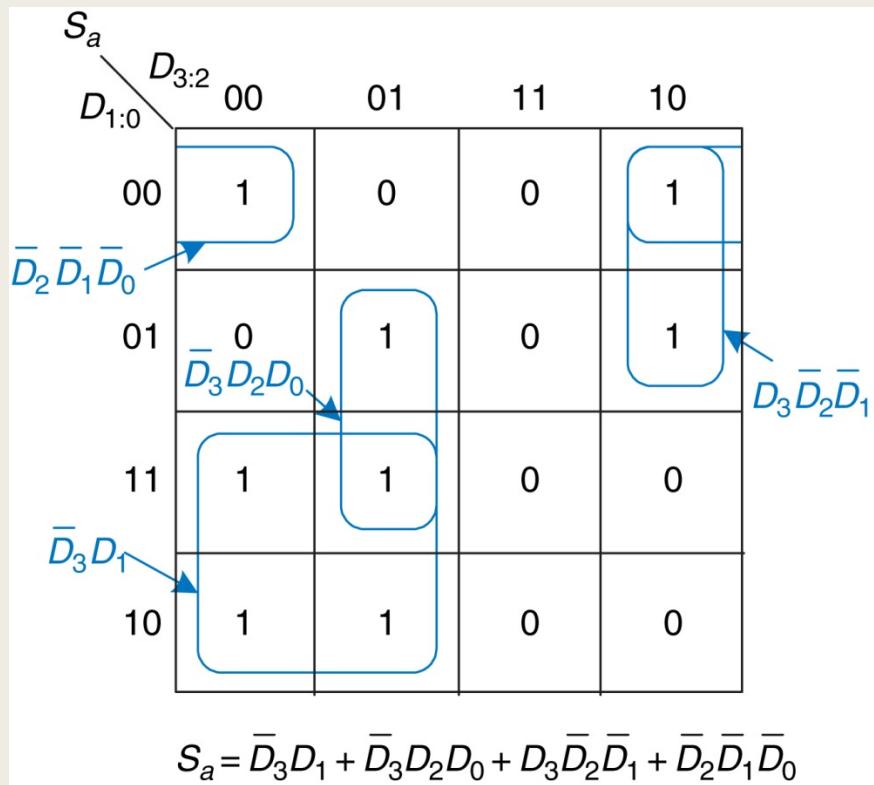
Παράδειγμα 2.10

- Αποκωδικοποιητής απεικόνισης επτά τμημάτων
- Απλοποιημένες εξισώσεις Boole για τις εξόδους S_a και S_b



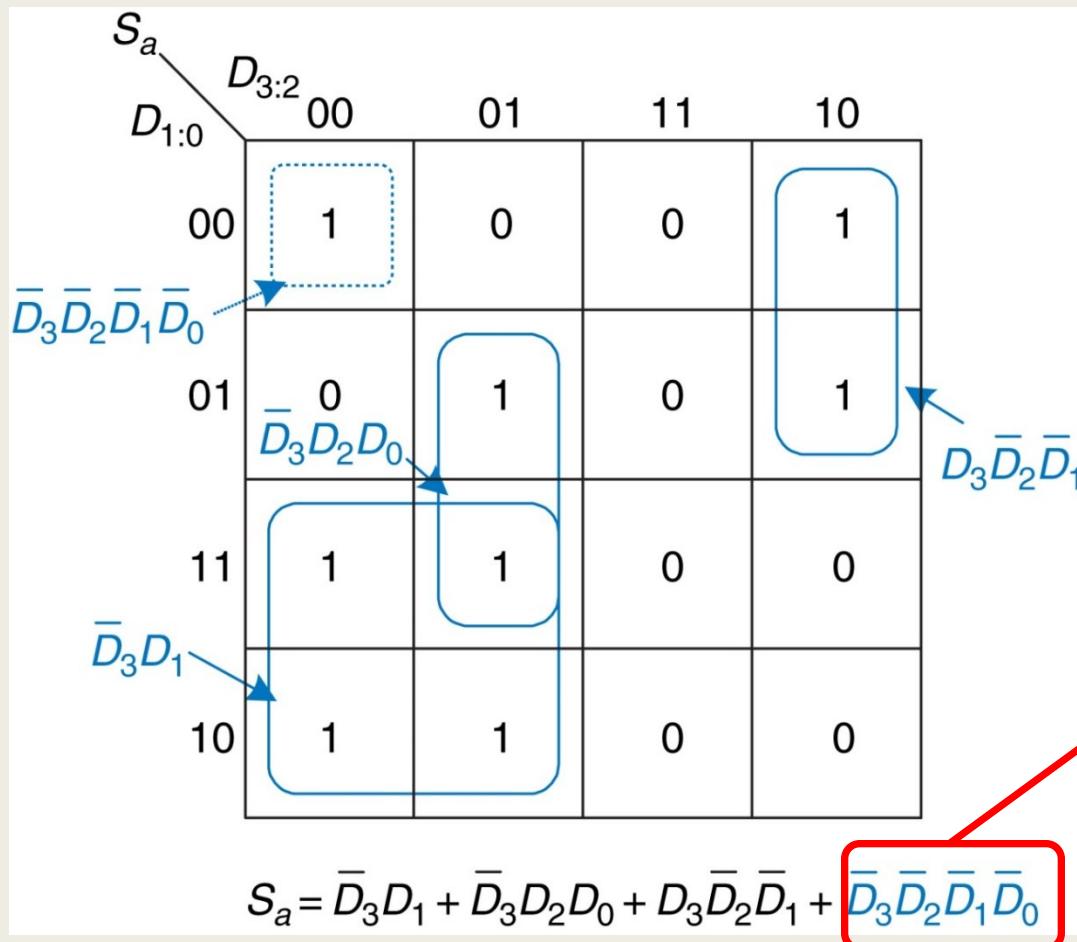
Παράδειγμα 2.10

- Αποκωδικοποιητής απεικόνισης επτά τμημάτων
- Απλοποιημένη εξίσωση Boole για την έξοδο S_a (εναλλακτική λύση)



Παράδειγμα 2.10

- Αποκωδικοποιητής απεικόνισης επτά τμημάτων
- Απλοποιημένη εξίσωση Boole για την έξοδο S_a (εσφαλμένη λύση)



Δεν ελήφθη υπόψη
η αναδίπλωση του
χάρτη Karnaugh:
τα ακραία είναι
γειτονικά

Δεν είναι πρώτος όρος

Αδιάφορες τιμές στην έξοδο

- Οι αδιάφορες τιμές (σύμβολο X) στις εισόδους σε έναν πίνακα αλήθειας χρησιμοποιούνται για να μειώσουμε το πλήθος των γραμμών του πίνακα, όταν κάποιες μεταβλητές δεν επηρεάζουν την έξοδο
- Τέτοιες αδιάφορες τιμές μπορούν επίσης να εμφανίζονται και σε εξόδους ενός πίνακα αληθείας όταν η τιμή της εξόδου δεν παίζει ρόλο ή όταν ο αντίστοιχος συνδυασμός εισόδων δεν εμφανίζεται σε κανονική λειτουργία του κυκλώματος
- Σε έναν χάρτη Karnaugh, τα σύμβολα X μας επιτρέπουν να προχωρήσουμε σε περαιτέρω ελαχιστοποίηση των λογικών εξισώσεων
- **Μπορούμε να κυκλώνουμε τα X αν αυτό μας βοηθάει να καλύπτουμε τα τετράγωνα με τιμή 1 με λιγότερους κύκλους ή κύκλους μεγαλύτερου μεγέθους, αλλά δεν χρειάζεται να τα κυκλώνουμε όταν αυτό δεν χρησιμεύει σε κάτι τέτοιο**
 - Οι κυκλωμένες αδιάφορες τιμές εκλαμβάνονται ως 1, ενώ οι μη κυκλωμένες εκλαμβάνονται ως 0.
 - Η χρήση αδιάφορων τιμών ελαχιστοποιεί σημαντικά τη λογική

Παράδειγμα 2.11

- **Αποκωδικοποιητής απεικόνισης επτά τμημάτων**
- Απλοποιημένες εξισώσεις Boole για τις εξόδους S_a και S_b λαμβάνοντας υπόψη ότι οι τιμές των εξόδων είναι αδιάφορες για τις εισόδους που δεν ανήκουν στον κώδικα BCD (**1010 – 1111**)

S_a	$D_{3:2}$	00	01	11	10
$D_{1:0}$	00	1	0	X	1
00	0	1	X	1	
01	1	1	X	X	
11	1	1	X	X	
10	1	1	X	X	

$$S_a = D_3 + D_2 D_0 + \bar{D}_2 \bar{D}_0 + D_1$$

$$S_a = \bar{D}_3 D_1 + \bar{D}_3 D_2 D_0 + D_3 \bar{D}_2 \bar{D}_1 + \bar{D}_2 \bar{D}_1 \bar{D}_0$$

S_b	$D_{3:2}$	00	01	11	10
$D_{1:0}$	00	1	1	X	1
00	1	0	X	1	
01	1	1	X	X	
11	1	1	X	X	
10	1	0	X	X	

$$S_b = \bar{D}_2 + D_1 D_0 + \bar{D}_1 \bar{D}_0$$

$$S_b = \bar{D}_3 \bar{D}_2 + \bar{D}_2 \bar{D}_1 + \bar{D}_3 D_1 D_0 + \bar{D}_3 \bar{D}_1 \bar{D}_0$$

Επιλεγμένες ασκήσεις

■ Άσκηση 2.34

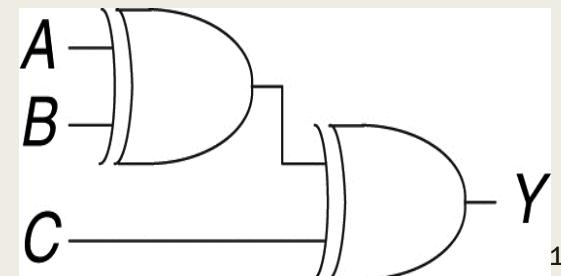
Ολοκληρώστε τη σχεδίαση των **τμημάτων S_c έως S_g** για τον αποκωδικοποιητή απεικόνισης επτά τμημάτων

- *Βρείτε εξισώσεις Boole για τις εξόδους S_c έως S_g , υποθέτοντας ότι οι είσοδοι με τιμή μεγαλύτερη από 9 είναι «αδιάφορες».*
- **Σχεδιάστε μια ευλόγως απλή υλοποίηση του αποκωδικοποιητή απεικόνισης επτά τμημάτων σε επίπεδο πυλών για όλες τις εξόδους του**
 - *Όπου είναι απαραίτητο, επιτρέπεται η κοινή χρήση πυλών από πολλές εξόδους*

Μείωση υλικού με τη χρήση πυλών XOR

- Παραδείγματα χαρτών Karnaugh που υλοποιούνται και με τη χρήση πυλών XOR των δύο εισόδων:
 - Βρείτε αρχικά την **εξίσωση Boole** στην απλοποιημένη μορφή **αθροίσματος γινομένων δύο επιπέδων**
 - Στη συνέχεια προχωρήστε την ελαχιστοποίηση, ώστε να χρησιμοποιήσετε την **πύλη XOR/XNOR δύο εισόδων**
 - Τέλος, σχεδιάστε το **σχηματικό διάγραμμα**
- Παράδειγμα: **Πύλη XOR με 3 εισόδους**
 - **Εξίσωση Boole:**
$$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$
 - Διαδικασία ελαχιστοποίσης με **XOR**:
$$\begin{aligned} Y &= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC) \\ &= \bar{A}(B \oplus C) + A(\bar{B} \oplus \bar{C}) \\ &= A \oplus B \oplus C = (A \oplus B) \oplus C \end{aligned}$$
 - Υλοποίηση με 2 πύλες XOR των 2 εισόδων

		<i>AB</i>	Domino			
		00	01	11	10	
<i>C</i>	0	0	1	0	1	
	1	1	0	1	0	



Μείωση υλικού με τη χρήση πυλών XOR

- Παραδείγματα χαρτών Karnaugh που υλοποιούνται και με τη χρήση πυλών XOR των δύο εισόδων. Να βρείτε τις πιο ελαχιστοποιημένες εξισώσεις

AB	$Y = \overline{A \oplus B \oplus C \oplus D}$			
CD	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

(α)

AB	$Y = \overline{B \oplus D}$			
CD	00	01	11	10
00	1	0	X	1
01	0	1	X	0
11	0	1	X	0
10	1	0	X	1

(β)

AB	$Y = B \oplus (CD)$			
CD	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	1	0	X	1
10	0	1	X	0

(γ)

AB	$Y = B \oplus C \oplus D$			
CD	00	01	11	10
00	0	1	X	0
01	1	0	X	1
11	0	1	X	0
10	1	0	X	1

(δ)

AB	$Y = \overline{A}(\overline{B} \oplus \overline{C})$			
CD	00	01	11	10
00	1	0	X	0
01	1	0	X	0
11	0	1	X	0
10	0	1	X	0

(ε)

AB	$Y = (AB) \oplus (CD)$			
CD	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	1	0	1
10	0	0	1	0

(στ)

Επιλεγμένες ερωτήσεις

■ Ερώτηση 2.2

Σχεδιάστε ένα κύκλωμα το οποίο να προσδιορίζει αν ένας δεδομένος μήνας έχει 31 μέρες

- Ο μήνας καθορίζεται από μια είσοδο $A_{3:0}$ των 4 bit
 - Για παράδειγμα, αν η είσοδος είναι 0001, ο μήνας είναι ο Ιανουάριος, και αν η είσοδος είναι 1100, ο μήνας είναι ο Δεκέμβριος.
- Η έξοδος Y του κυκλώματος πρέπει να έχει την τιμή '1' HIGH μόνο όταν ο μήνας που καθορίζεται από την είσοδο έχει 31 μέρες

Γράψτε την ελαχιστοποιημένη εξίσωση και σχεδιάστε το διάγραμμα του κυκλώματος χρησιμοποιώντας τον ελάχιστο δυνατό αριθμό πυλών

$$Y = \bar{A}_3 A_0 + A_3 \bar{A}_0 = A_3 \oplus A_0$$

	$A_{3:2}$	$A_{1:0}$	00	01	11	10	
00	X	0	1	1			
01	1	1	X	0			
11	1	1	X	0			
10	0	0	X	1			

Month	A_3	A_2	A_1	A_0	Y
Jan	0	0	0	1	1
Feb	0	0	1	0	0
Mar	0	0	1	1	1
Apr	0	1	0	0	0
May	0	1	0	1	1
Jun	0	1	1	0	0
Jul	0	1	1	1	1
Aug	1	0	0	0	1
Sep	1	0	0	1	0
Oct	1	0	1	0	1
Nov	1	0	1	1	0
Dec	1	1	0	0	1

Κώδικας Gray

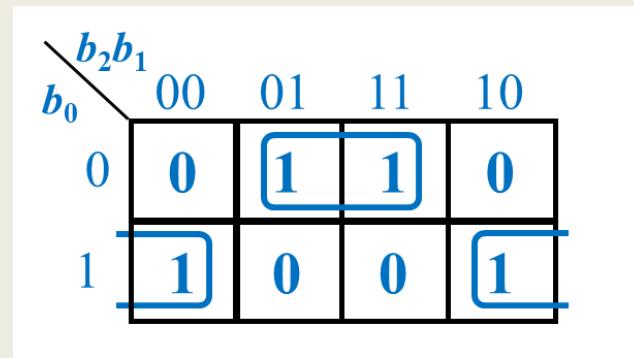
- Ο κώδικας Gray των 3 bit (g_2, g_1, g_0) παράγεται από τον δυαδικό κώδικα των 3 bit (b_2, b_1, b_0) ως εξής:
 - Εάν $b_1 = b_0$, τότε $g_0 = 0$, αλλιώς $g_0 = 1$.
 - Εάν $b_2 = b_1$, τότε $g_1 = 0$, αλλιώς $g_1 = 1$.
 - Για το περισσότερο σημαντικό ψηφίο $g_2 = b_2$
- Η ίδια διαδικασία παραγωγής του κώδικα Gray από το δυαδικό κώδικα ακολουθείται ανεξάρτητα από το πλήθος των ψηφίων.
- Το σημαντικό χαρακτηριστικό του κώδικα Gray είναι ότι δύο διαδοχικές κωδικές λέξεις του διαφέρουν μόνο κατά ένα ψηφίο
 - Χρησιμοποιείται στους **ADCs**, για να αποφευχθούν ενδιάμεσες τιμές, όπου τα προκύπτοντα ψηφιακά δεδομένα αυξάνονται ή μειώνονται κατά 1.
 - Χρησιμοποιείται στην κωδικοποίηση καταστάσεων των ακολουθιακών κυκλωμάτων για **μείωση κατανάλωσης ισχύος**

Κύκλωμα μετατροπής από τον δυαδικό στον Gray

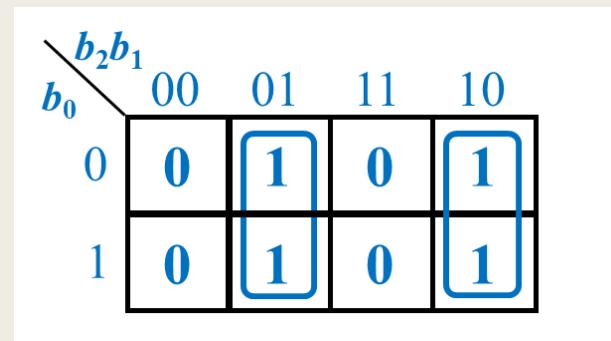
- Πίνακας αλήθειας

$b_2 b_1 b_0$	$g_2 g_1 g_0$
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

- Χάρτες Karnaugh και εξισώσεις Boole



$$- \quad g_0 = b_1 \oplus b_0$$



$$- \quad g_1 = b_2 \oplus b_1$$

$$- \quad g_2 = b_2$$

Λογικοί συνθετητές

- Η áλγεβρα Boole και οι χάρτες Karnaugh αποτελούν δύο μεθόδους ελαχιστοποίησης λογικών εξισώσεων.
- Μια σύγχρονη πρακτική των σχεδιαστών ψηφιακών συστημάτων είναι να χρησιμοποιούν προγράμματα υπολογιστή, τα οποία ονομάζονται **λογικοί συνθετητές** (**logic synthesizers**), για να παράγουν ελαχιστοποιημένα κυκλώματα από μια περιγραφή της λογικής συνάρτησης.
- Για σύνθετα προβλήματα σχεδίασης, οι λογικοί συνθετητές είναι πολύ πιο αποδοτικοί από τους ανθρώπους.

Δομικά στοιχεία συνδυαστικής λογικής

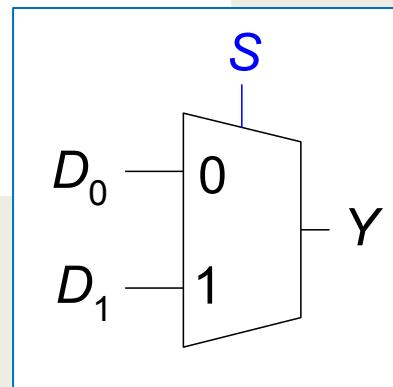
- Η συνδυαστική λογική συχνά ομαδοποιείται σε μεγαλύτερα δομικά στοιχεία για την κατασκευή πιο σύνθετων συστημάτων
- Αυτή είναι μια εφαρμογή της αρχής της «αφαίρεσης»
 - αποκρύπτονται οι περιττές λεπτομέρειες σε επίπεδο πυλών ώστε να δοθεί έμφαση στη συνάρτηση του δομικού στοιχείου
- Τέτοια κυκλώματα που έχουμε ήδη αναφέρει είναι:
 - ο **πλήρης αθροιστής**
 - το **κύκλωμα και ο κωδικοποιητής προτεραιότητας**
 - ο **αποκωδικοποιητής απεικόνισης επτά τμημάτων**
- Στη συνέχεια θα επικεντρωθούμε:
 - στους **πολυπλέκτες**
 - στους **δυαδικούς αποκωδικοποιητές**, και
 - στους **αποπλέκτες**

Πολυπλέκτης 2 σε 1

- Ο πολυπλέκτης 2 σε 1 επιλέγει τη μία από τις δύο εισόδους δεδομένων D_0 και D_1 με βάση την τιμή του σήματος S (select)
 - αν $S = 0$, τότε $Y = D_0$
 - αλλιώς, αν $S = 1$, τότε $Y = D_1$

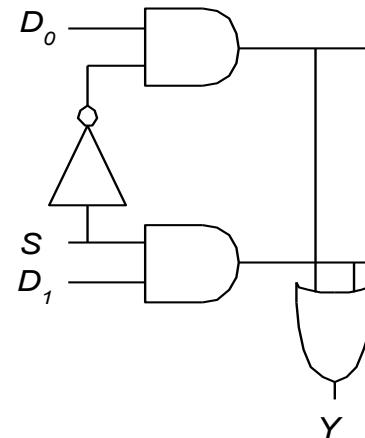
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

S	Y
0	D_0
1	D_1



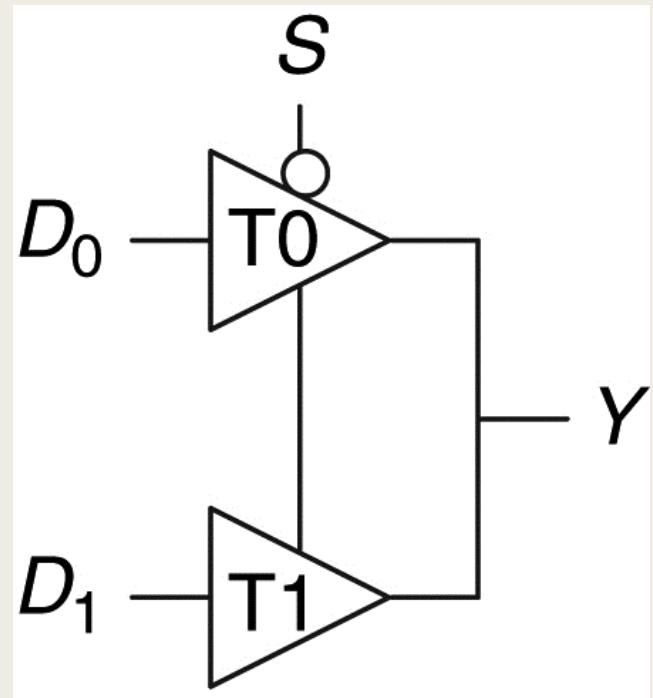
S	D_0	D_1	Y
0	0	0	0
1	0	1	1

$$Y = D_0 \bar{S} + D_1 S$$



Πολυπλέκτης 2 σε 1

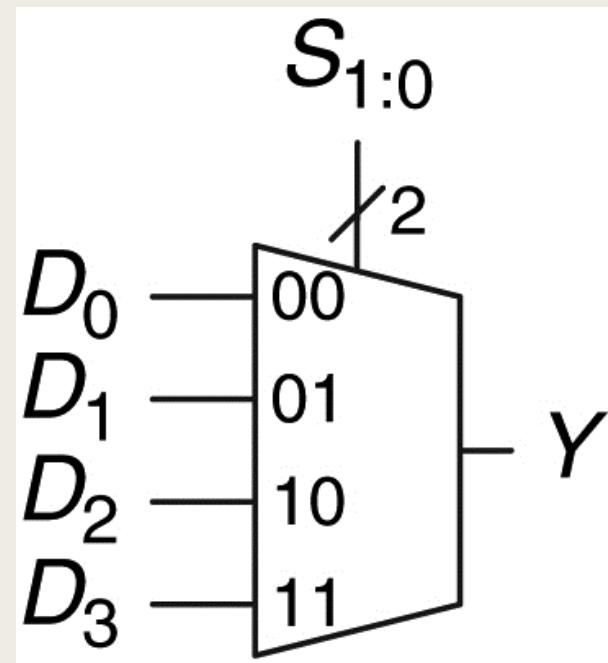
- Εναλλακτικά, πολυπλέκτες μπορούν να κατασκευαστούν από **απομονωτές τριών καταστάσεων**
- Τα σήματα έγκρισης (enable) τριών καταστάσεων είναι διατεταγμένα έτσι ώστε, σε όλες τις περιπτώσεις, να είναι ενεργός ακριβώς ένας απομονωτής τριών καταστάσεων
- Όταν ισχύει **$S = 0$** , είναι ενεργοποιημένος ο απομονωτής **T0**, γεγονός που επιτρέπει στην τιμή του D_0 να μεταφερθεί στην έξοδο Y
- Όταν ισχύει **$S = 1$** , είναι ενεργοποιημένος ο απομονωτής **T1**, γεγονός που επιτρέπει στην τιμή του D_1 να μεταφερθεί στην έξοδο Y



$$Y = D_0 \bar{S} + D_1 S$$

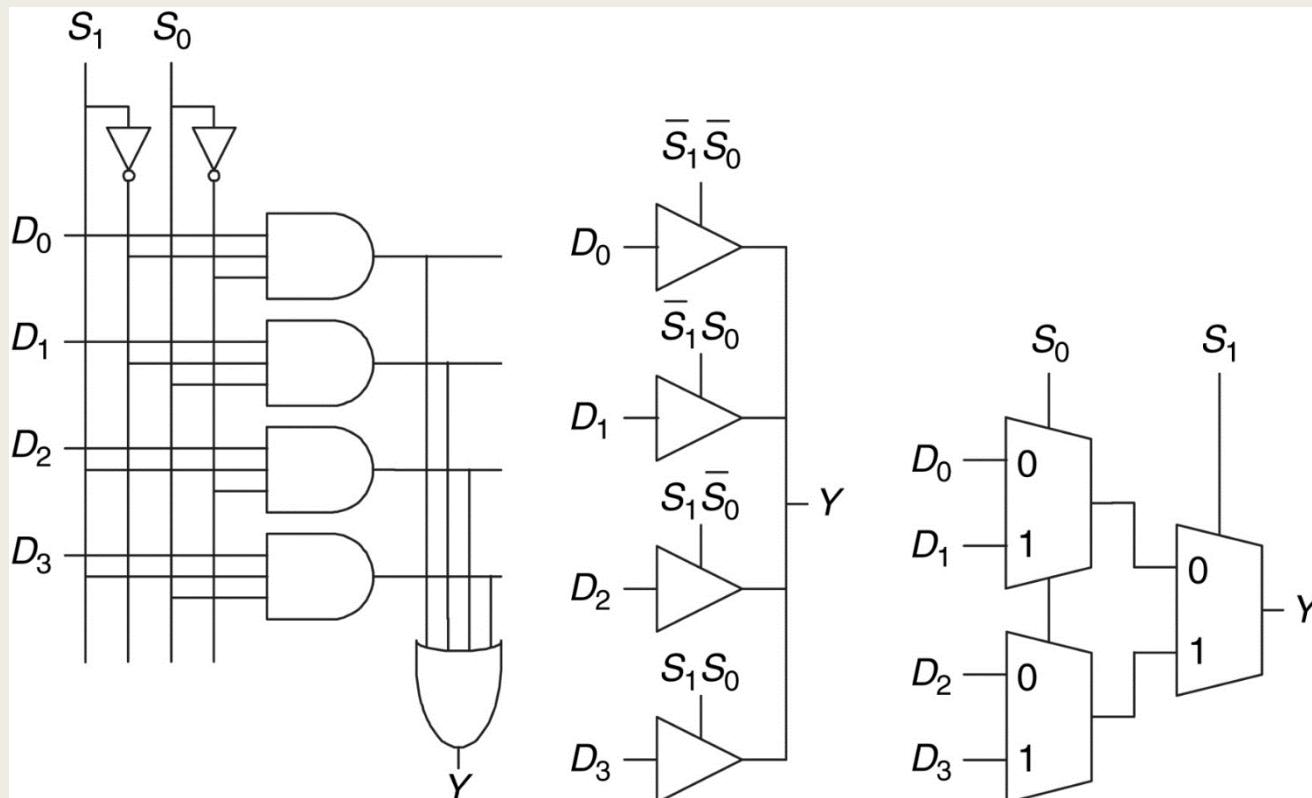
Πολυπλέκτης 4 σε 1

- Ο πολυπλέκτης 4 σε 1 επιλέγει τη μία από τις τέσσερεις εισόδους δεδομένων D_0, D_1, D_2 και D_3 με βάση την τιμή του διαύλου επιλογής $S_{1:0}$
 - αν $S_{1:0} = 00$, τότε $Y = D_0$
 - αν $S_{1:0} = 01$, τότε $Y = D_1$
 - αν $S_{1:0} = 10$, τότε $Y = D_2$
 - αν $S_{1:0} = 11$, τότε $Y = D_3$
- Εξίσωση Boole του πολυπλέκτη 4 σε 1:
$$Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$$



Πολυπλέκτης 4 σε 1

- Μπορούμε να κατασκευάσουμε τον πολυπλέκτη 4 σε 1 χρησιμοποιώντας
 - (α) λογική αθροίσματος γινομένων, (β) απομονωτές τριών καταστάσεων
 - ή (γ) δένδρο πολυπλεκτών 2 σε 1



Έχουμε τη δυνατότητα να κατασκευάσουμε μεγαλύτερους πολυπλέκτες, επεκτείνοντας αυτές τις τρεις μεθόδους. Ενας πολυπλέκτης $N = 2^n$ σε 1 απαιτεί $\log_2 N = n$ σήματα επιλογής (select).

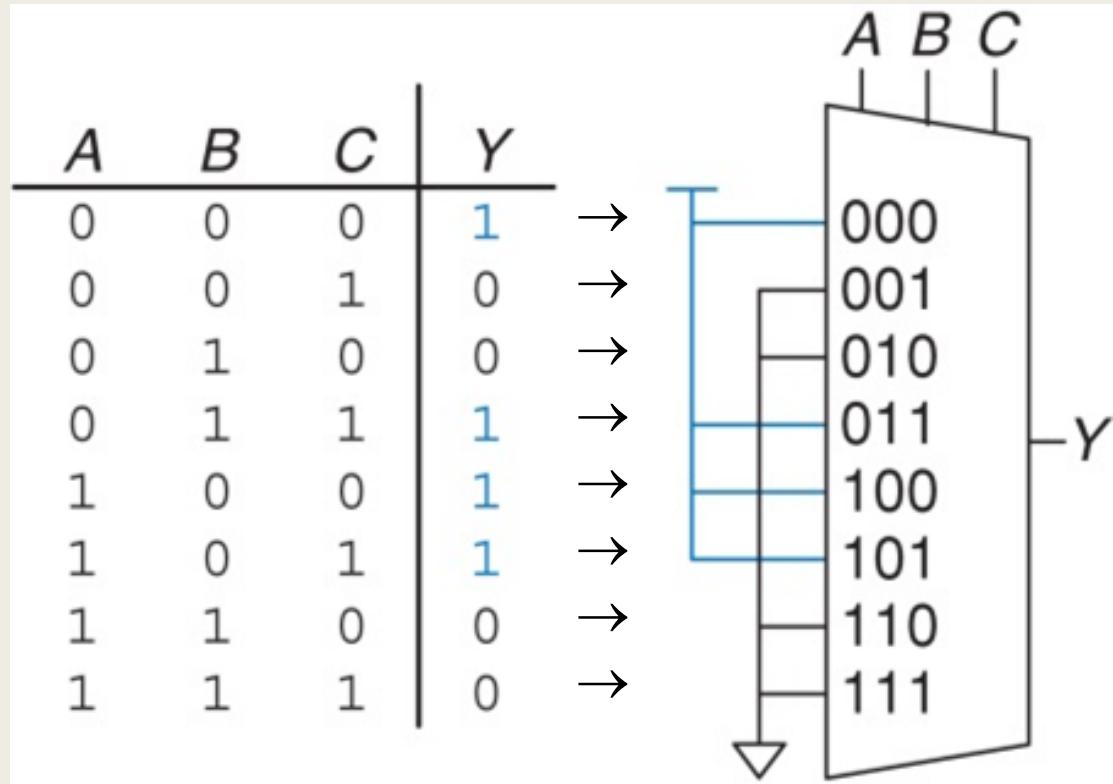
Υλοποίηση συνδυαστικής λογικής

■ Παράδειγμα 4.12

- Υλοποιήστε την συνάρτηση

$$Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$$

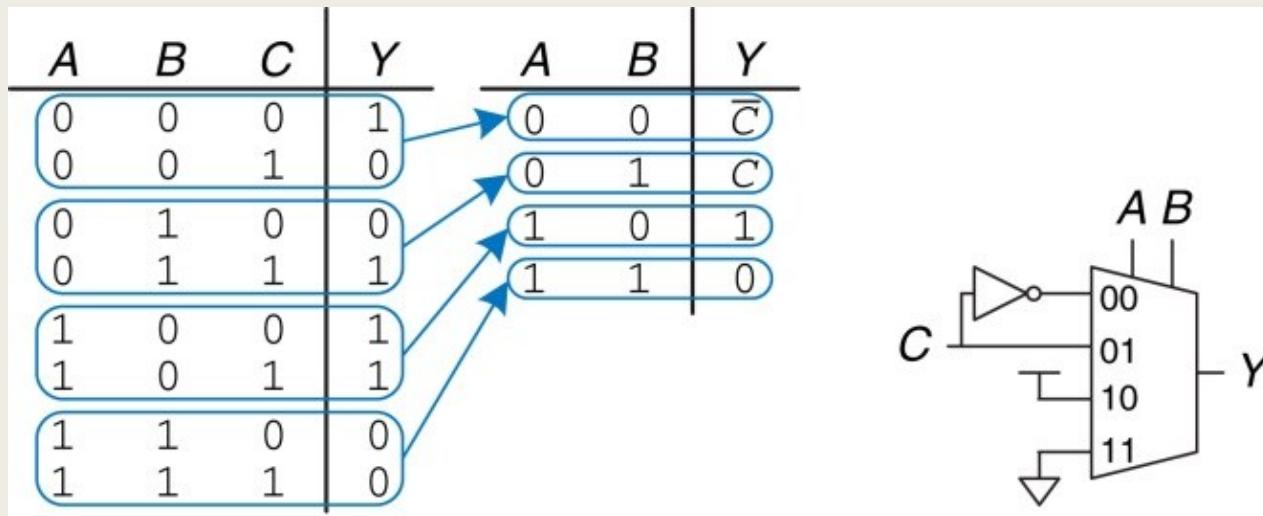
με την χρήση μονάχα
ενός πολυπλέκτη 8 σε 1



- Ο πολυπλέκτης χρησιμεύει ως **πίνακας αναζήτησης (Look-Up Table – LUT)**,
όπου κάθε γραμμή του πίνακα αληθείας (κάθε ελαχιστόρος) αντιστοιχεί
σε μια είσοδο του πολυπλέκτη
 - Οι είσοδοι του πολυπλέκτη ταυτίζονται με την έξοδο Y του πίνακα

Υλοποίηση συνδυαστικής λογικής

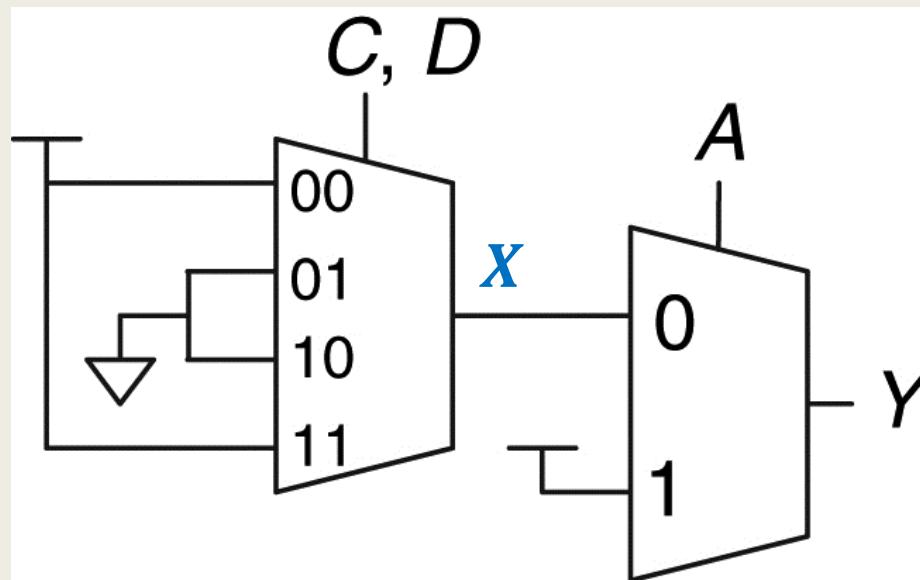
- **Παράδειγμα 4.12 (μείωση του πολυπλέκτη στο μισό)**
- Υλοποιήστε την συνάρτηση $Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$
 - Με την χρήση μονάχα ενός **πολυπλέκτη 4 σε 1** και μίας **πύλης NOT**
- Μειώνουμε το μέγεθος του πίνακα αληθείας στο μισό, από 8 γραμμές σε 4 γραμμές επιτρέποντας στην έξοδο να εξαρτάται από την τιμή του C
 - Για κάθε δύο γραμμές του πίνακα αλήθειας εξετάζουμε την τιμή της εξόδου Y με βάση την τιμή της εισόδου C (του **LSB**)
 - Υπάρχουν 4 πιθανές περιπτώσεις: $Y = C$, $Y = \bar{C}$, $Y = 1$ και $Y = 0$
- Υλοποιούμε τον πολυπλέκτη με βάση τον μειωμένο πίνακα αλήθειας



Επιλεγμένες ασκήσεις

■ Άσκηση 2.39

Γράψτε μια ελαχιστοποιημένη εξίσωση Boole για τη συνάρτηση που εκτελείται από το κύκλωμα της Εικόνας 2.87



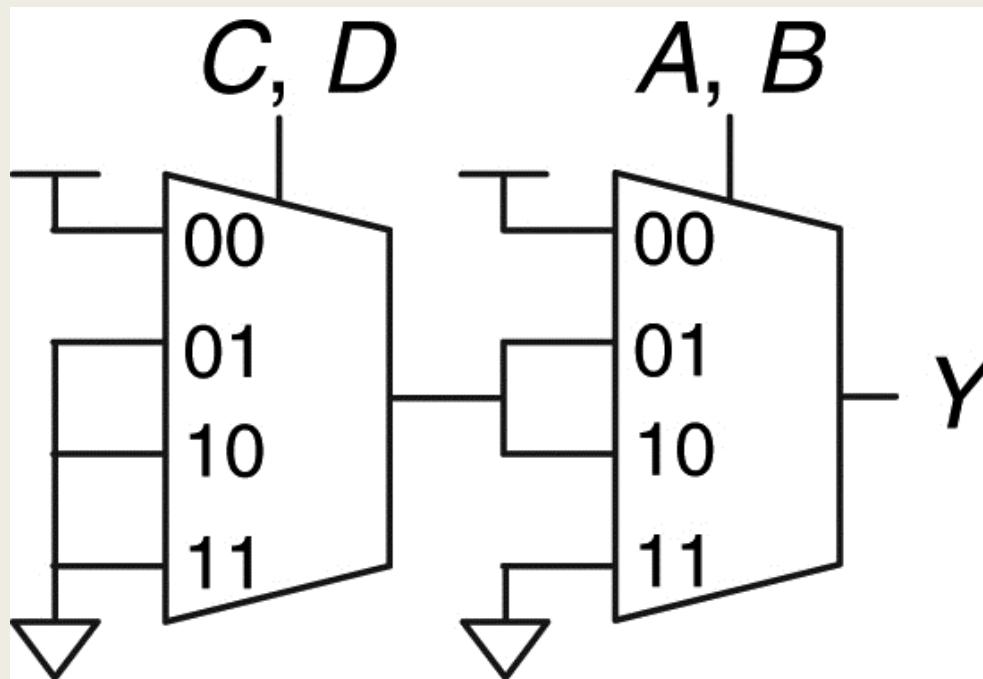
$$X = \bar{C}\bar{D}(1) + \bar{C}D(0) + C\bar{D}(0) + CD(1) = \bar{C}\bar{D} + CD = \overline{C \oplus D}$$

$$\begin{aligned} Y &= \bar{A}(\overline{C \oplus D}) + A(1) = \bar{A}(\overline{C \oplus D}) + A = (\bar{A} + A)(\overline{C \oplus D} + A) \\ &= (1)(\overline{C \oplus D} + A) = \overline{C \oplus D} + A = \bar{C}\bar{D} + CD + A \end{aligned}$$

Επιλεγμένες ασκήσεις

■ Άσκηση 2.40

Γράψτε μια ελαχιστοποιημένη εξίσωση Boole για τη συνάρτηση που εκτελείται από το κύκλωμα της Εικόνας 2.88



$$Y = \overline{CD}(\overline{A} \oplus \overline{B}) + \overline{AB} = \overline{ACD} + \overline{BCD} + \overline{AB}$$

Επιλεγμένες ασκήσεις

■ Άσκηση 2.41

Υλοποιήστε τη συνάρτηση της Εικόνας 2.81(β) χρησιμοποιώντας

(α) έναν πολυπλέκτη 8 σε 1.

(β) έναν πολυπλέκτη 4 σε 1 και έναν αντιστροφέα.

(γ) έναν πολυπλέκτη 2 σε 1 και δύο άλλες λογικές πύλες

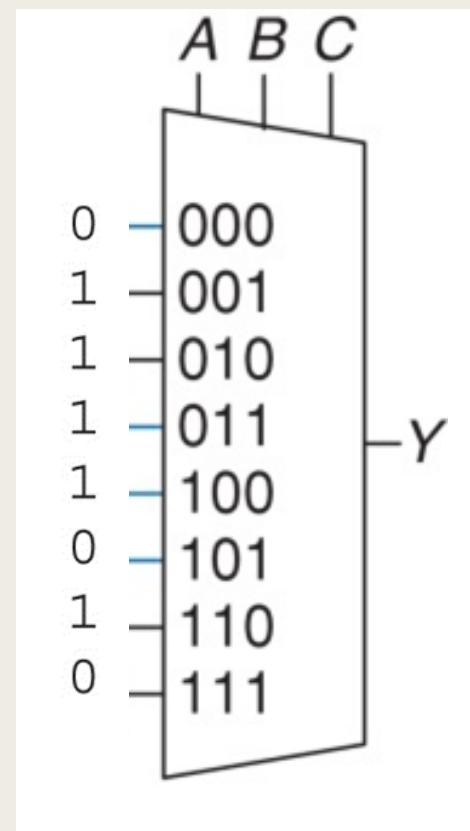
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Επιλεγμένες ασκήσεις

■ Άσκηση 2.41

Υλοποιήστε τη συνάρτηση της Εικόνας 2.81(β) χρησιμοποιώντας
(α) έναν πολυπλέκτη 8 σε 1.

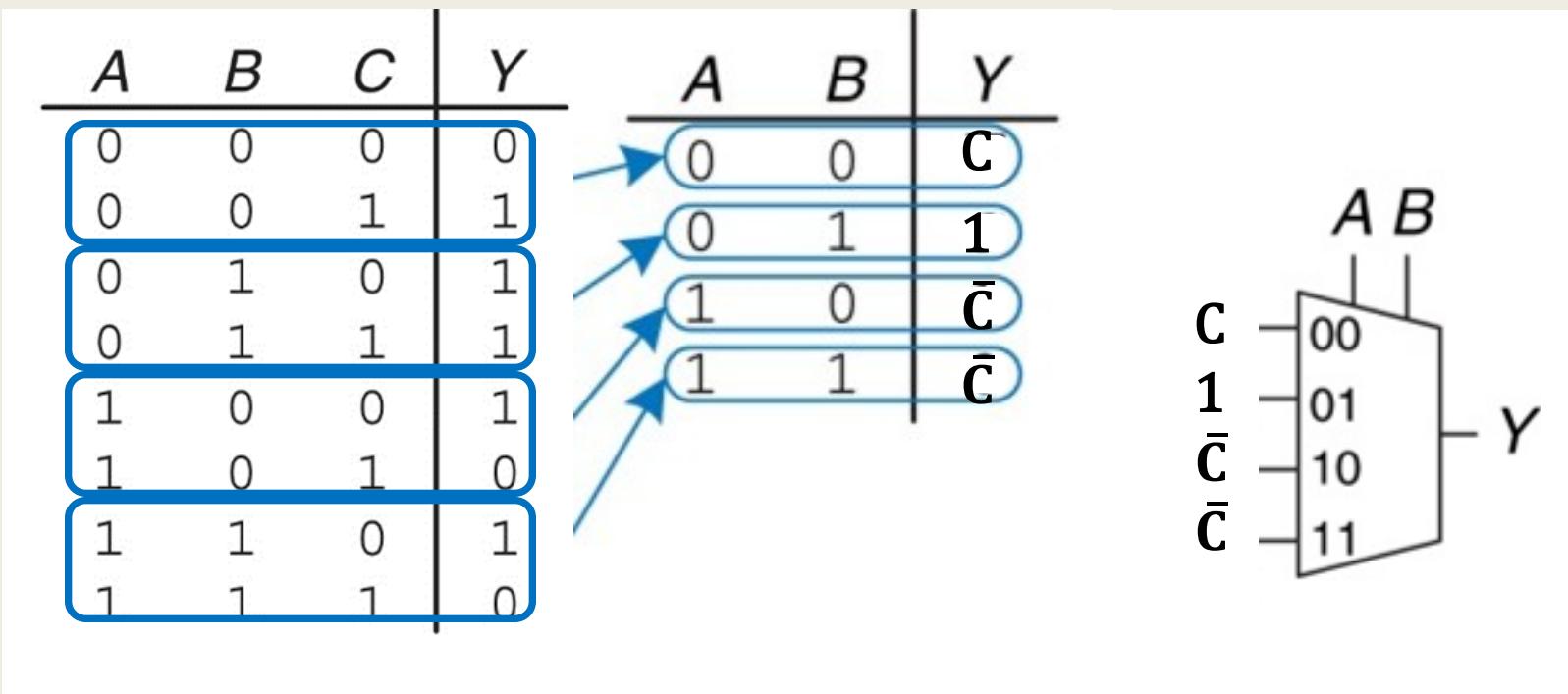
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



Επιλεγμένες ασκήσεις

■ Άσκηση 2.41

Υλοποιήστε τη συνάρτηση της Εικόνας 2.81(β) χρησιμοποιώντας
(β) έναν πολυπλέκτη 4 σε 1 και έναν αντιστροφέα



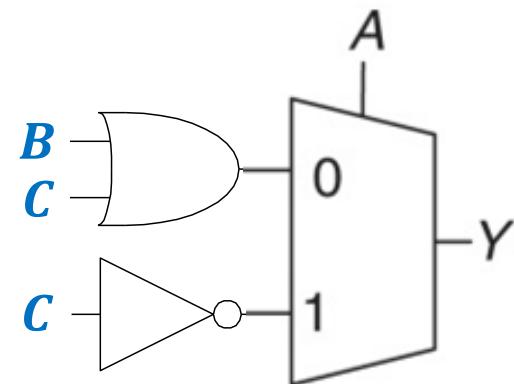
Επιλεγμένες ασκήσεις

■ Άσκηση 2.41

Υλοποιήστε τη συνάρτηση της Εικόνας 2.81(β) χρησιμοποιώντας (γ) έναν πολυπλέκτη 2 σε 1 και δύο άλλες λογικές πύλες

- Μειώνουμε το μέγεθος του πίνακα αληθείας από 8 γραμμές σε 2 γραμμές επιτρέποντας στην έξοδο να εξαρτάται από τα B και C
- Για κάθε 4 γραμμές του πίνακα αλήθειας βρίσκουμε τη συνάρτηση δύο μεταβλητών B και C που αντιστοιχεί στην έξοδο Y
 - Υπάρχουν 16 πιθανές συναρτήσεις δύο μεταβλητών
- Υλοποιούμε τις δύο προκύπτουσες συναρτήσεις με λογικές πύλες και συνδέουμε τις εξόδους τους στις αντίστοιχες εισόδους του πολυπλέκτη 2 σε 1 με βάση τον μειωμένο πίνακα αλήθειας

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



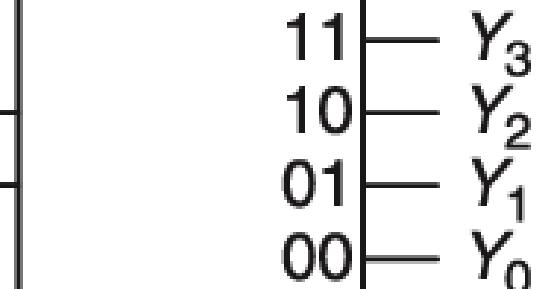
Δυαδικός αποκωδικοποιητής N σε 2^N

- Ένας δυαδικός αποκωδικοποιητής (binary decoder) N σε 2^N
 - δέχεται δυαδικούς αριθμούς των N bit ως εισόδους A_m , ($m = 0, \dots, N-1$)
 - και παράγει 2^N εξόδους Y_n , ($n = 0, \dots, 2^N-1$)
 - Ενεργοποιεί ακριβώς μία από τις εξόδους του Y_n , εκείνη που έχει ως δείκτη n τη δεκαδική τιμή του δυαδικού αριθμού στην είσοδο
- Οι έξοδοι αποκαλούνται έξοδοι «μοναδικού σημαντικού» (one-hot), ή «μονόθερμες», επειδή ακριβώς μία από αυτές είναι «θερμή» (HIGH) σε οποιαδήποτε δεδομένη στιγμή

Παράδειγμα: Δυαδικός αποκωδικοποιητής 2 σε 4

A_1	A_0	Y_3	Y_2	Y_1	Y_0	n
0	0	0	0	0	1	0
0	1	0	0	1	0	1
1	0	0	1	0	0	2
1	1	1	0	0	0	3

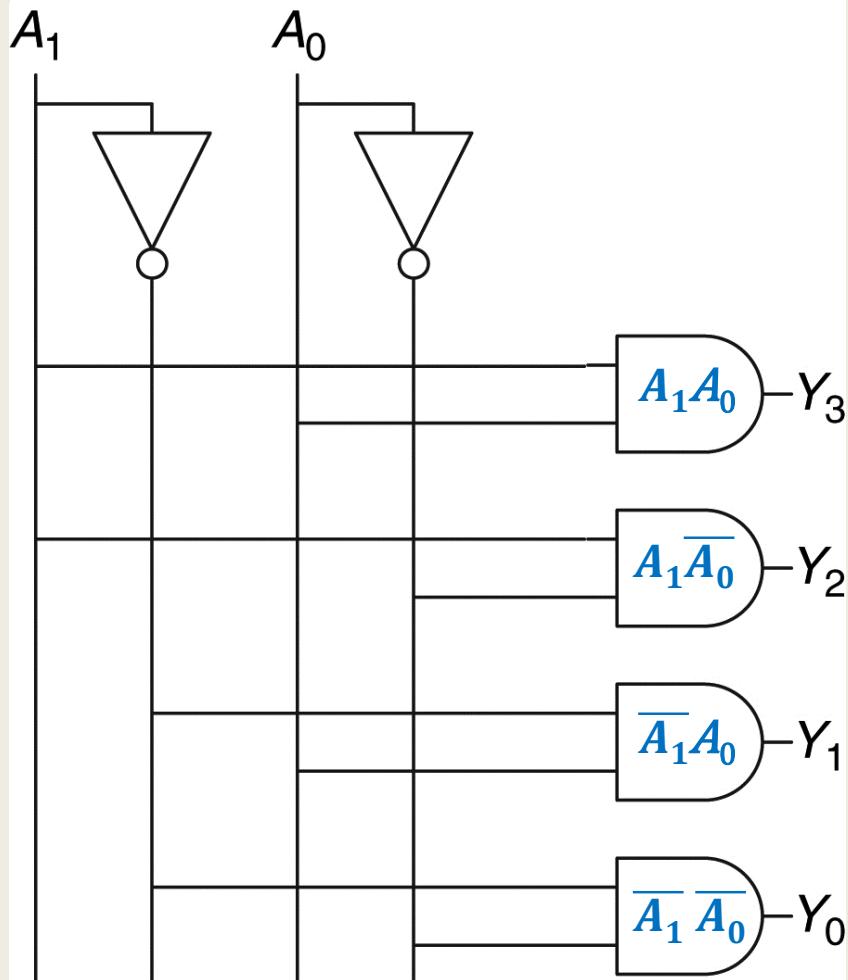
Αποκωδικοποιητής 2:4



Υλοποίηση του δυαδικού αποκωδικοποιητή

- Υλοποιείται με πύλες AND
- Κάθε πύλη AND εξαρτάται είτε από την αληθινή είτε από τη συμπληρωματική μορφή κάθε εισόδου A_m , ($m = 0,..,N-1$)
- Ένας αποκωδικοποιητής N σε 2^N μπορεί να κατασκευαστεί από **2^N πύλες AND των N εισόδων**
 - που δέχονται τους πιθανούς συνδυασμούς των αληθινών ή συμπληρωματικών εισόδων
- Κάθε **έξοδος Y_n** ($n = 0,.., 2^N-1$) αναπαριστά τον **ελαχιστόρο m_n** (π.χ., $Y_3 = m_3 = A_0 A_1$)

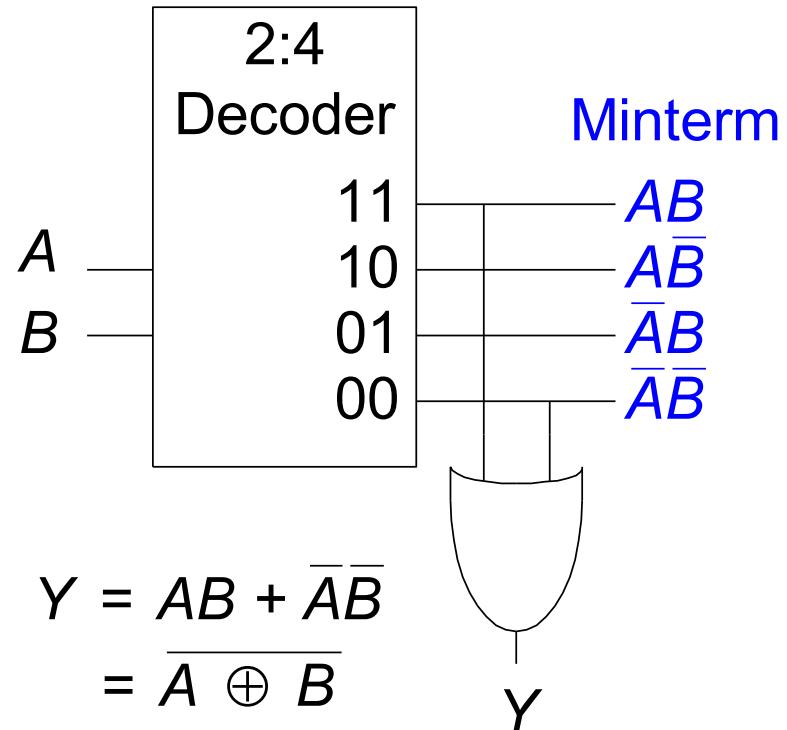
Παράδειγμα: Δυαδικός αποκωδικοποιητής 2 σε 4



Υλοποίηση συνδυαστικής λογικής

- Οι δυαδικοί αποκωδικοποιητές μπορούν να συνδυαστούν με **πύλες OR** για την υλοποίηση συνδυαστικής λογικής
- Επειδή κάθε έξοδος Y_n ενός αποκωδικοποιητή ($n = 0, \dots, 2^N-1$) αναπαριστά τον **ελαχιστόρο m_n**, η συνάρτηση υλοποιείται ως **η πράξη OR όλων των ελαχιστόρων της συνάρτησης**
 - **που έχουν την τιμή 1** στον πίνακα αληθείας
- Η ίδια ιδέα εφαρμόζεται και στην υλοποίηση συνδυαστικής λογικής με **διατάξεις μνήμης**

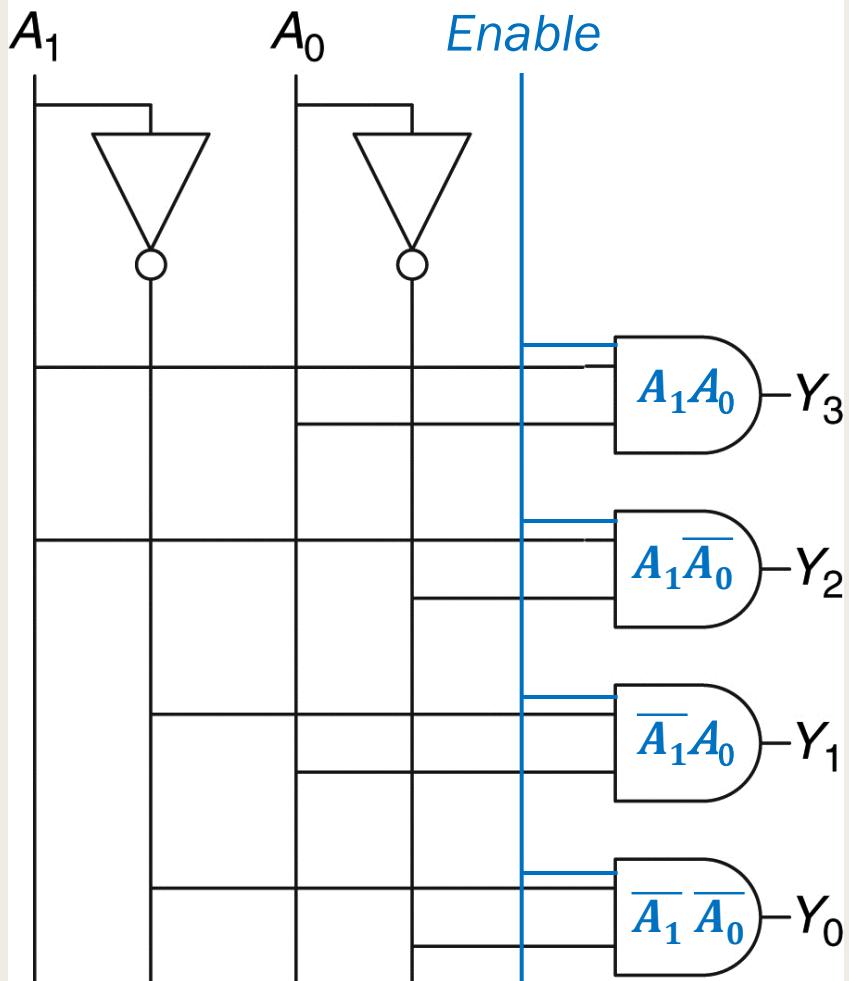
Παράδειγμα: Υλοποίηση της πύλης XNOR δύο εισόδων με δυαδικό αποκωδικοποιητή 2 σε 4



Δυαδικός αποκωδικοποιητής με έγκριση

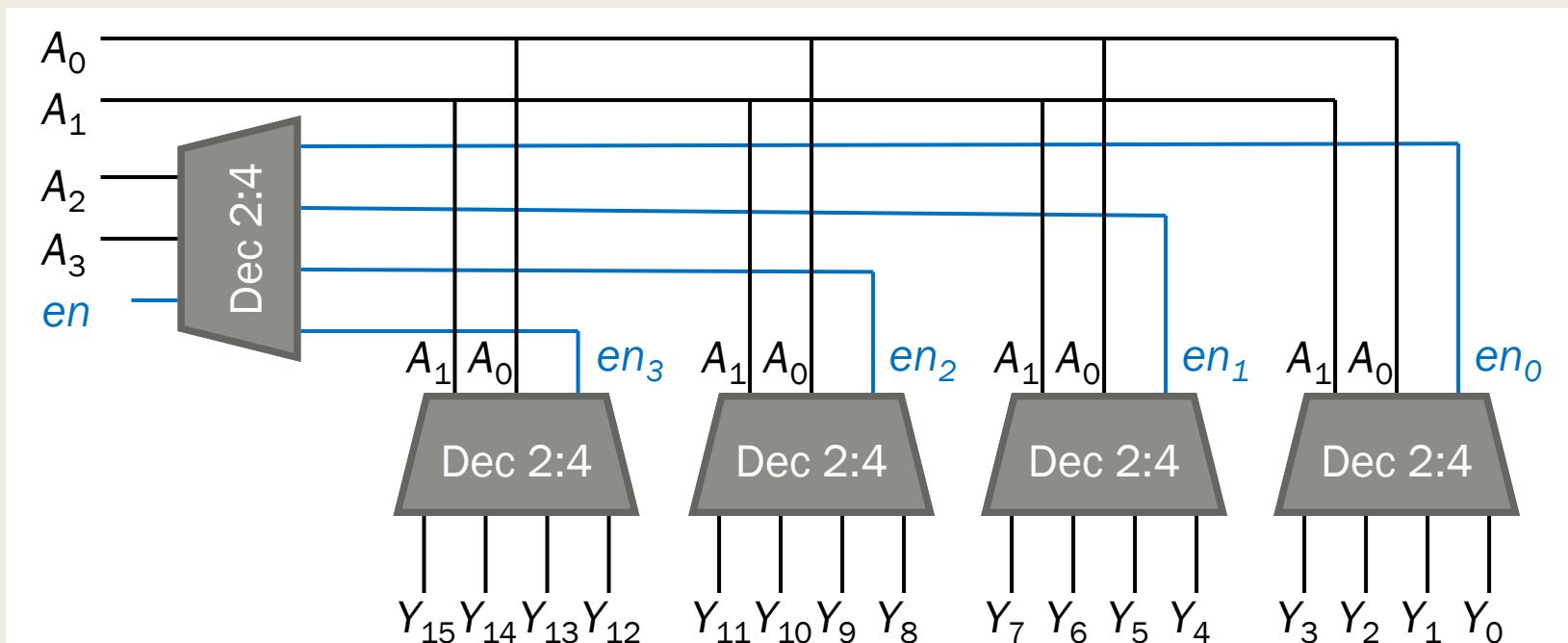
- Το **σήμα enable (έγκρισης)** είναι μία επιπλέον είσοδος του δυαδικού αποκωδικοποιητή
 - όταν το **σήμα enable** έχει τιμή '1' (**TRUE**) ο αποκωδικοποιητής λειτουργεί κανονικά
 - όταν το **σήμα enable** έχει τιμή '0' (**FALSE**) ο αποκωδικοποιητής έχει όλες τις εξόδους μηδενισμένες
- Ένας **αποκωδικοποιητής N σε 2^N** μπορεί να κατασκευαστεί από **2^N πύλες AND** των $N+1$ εισόδων
 - που δέχονται τους πιθανούς συνδυασμούς των αληθινών ή συμπληρωματικών εισόδων και το **σήμα enable**

Παράδειγμα: Δυαδικός αποκωδικοποιητής 2 σε 4 με enable



Δυαδικός αποκωδικοποιητής με έγκριση

- Το **σήμα enable (έγκρισης)** είναι μία επιπλέον είσοδος του δυαδικού αποκωδικοποιητή που χρησιμοποιείται όταν κατασκευάζουμε μεγαλύτερους αποκωδικοποιητές από πολλούς μικρότερους
 - *Η χρήση του σήματος enable για επέκταση του μεγέθους ενός κυκλώματος δεν περιορίζεται μόνο στους αποκωδικοποιητές, αλλά εφαρμόζεται και σε άλλα κυκλώματα (μετρητές, μνήμες, κ.α.)*
- Παράδειγμα: **Κατασκευή δυαδικού αποκωδικοποιητή 4 σε 16 με έγκριση χρησιμοποιώντας 5 αποκωδικοποιητές 2 σε 4 με enable**

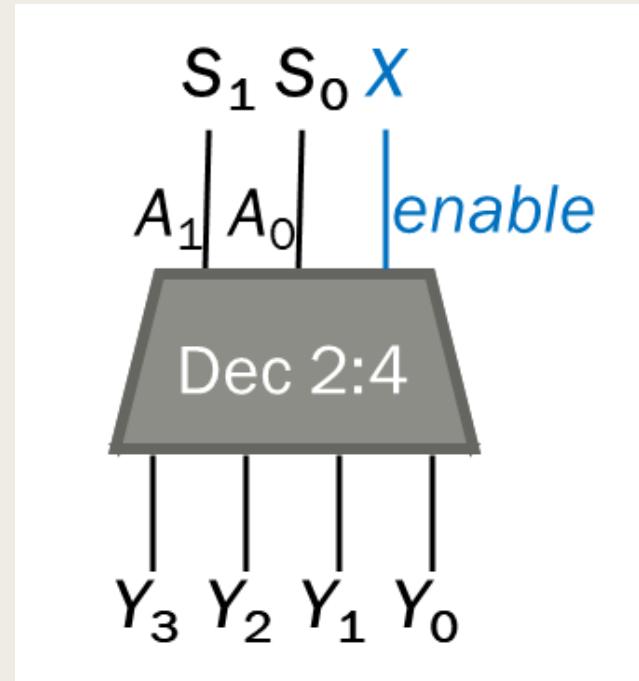


Αποπλέκτης 1 σε 2^N

- Ο αποπλέκτης (demultiplexer) 1 σε 2^N συνδέει τη μοναδική είσοδο δεδομένων X με μία από τις 2^N εξόδους δεδομένων με βάση την τιμή των N σημάτων επιλογής S (select)
- Ο δυαδικός αποκωδικοποιητής N σε 2^N με έγκριση χρησιμοποιείται και ως αποπλέκτης (demultiplexer) 1 σε 2^N ως εξής:
 - Το σήμα enable του αποκωδικοποιητή είναι η είσοδος δεδομένων X του αποπλέκτη
 - Οι N είσοδοι δεδομένων του αποκωδικοποιητή είναι τα N σήματα επιλογής του αποπλέκτη
 - Οι 2^N εξοδοί «μοναδικού σημαντικού» του αποκωδικοποιητή είναι οι 2^N εξοδοί δεδομένων του αποπλέκτη

Αποπλέκτης 1 σε 4

- Ο αποπλέκτης 1 σε 4 συνδέει την είσοδο δεδομένων X με μία από τις τέσσερεις εξόδους δεδομένων Y_0 , Y_1 , Y_2 και Y_3 με βάση την τιμή του διαύλου επιλογής $S_{1:0}$
 - αν $S_{1:0} = 00$, τότε $Y_0 = X$
 - αν $S_{1:0} = 01$, τότε $Y_1 = X$
 - αν $S_{1:0} = 10$, τότε $Y_2 = X$
 - αν $S_{1:0} = 11$, τότε $Y_3 = X$
- Υλοποιείται με έναν δυαδικό αποκωδικοποιητή 2 σε 4 με έγκριση (enable)

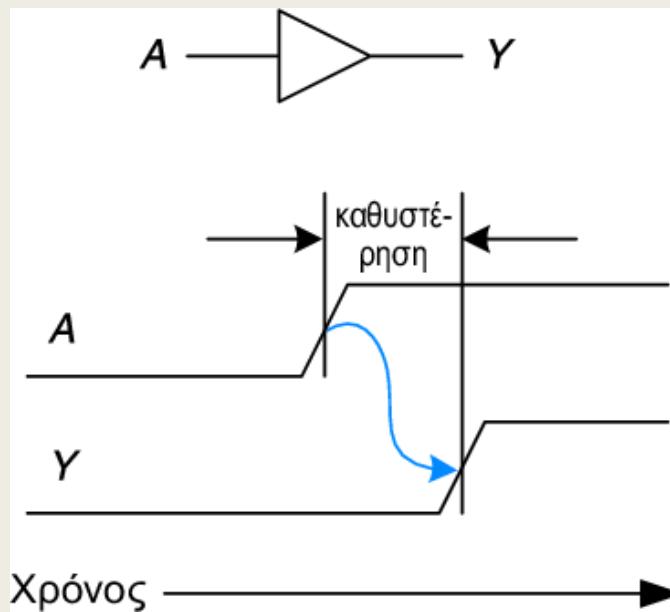


Χρονισμός

- Στη σχεδίαση ενός ψηφιακού κυκλώματος είναι σημαντικά:
 - η **μείωση του κόστους υλοποίησης** με τη χρήση όσο το δυνατόν λιγότερων δομικών στοιχείων.
 - η **αύξηση της ταχύτητας λειτουργίας**, δηλαδή ο **χρονισμός (timing)**

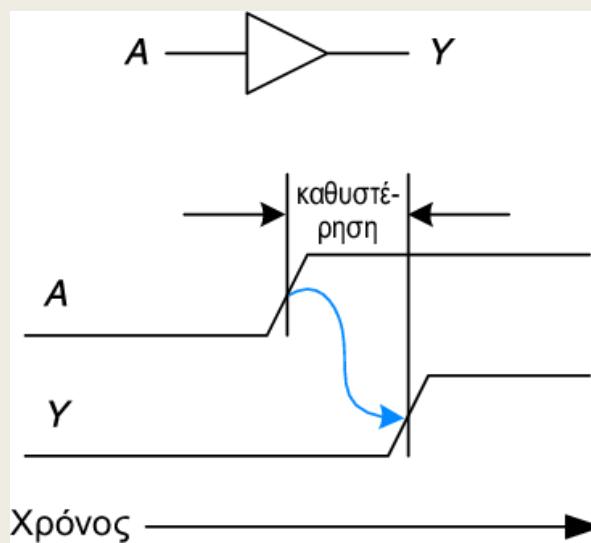
Χρονισμός: καθυστέρηση

- Μια έξοδος χρειάζεται χρόνο για να μεταβληθεί αντιδρώντας στη μεταβολή μιας εισόδου
- Στην εικόνα φαίνεται **η καθυστέρηση (delay)** (ο χρόνος) που μεσολαβεί από τη μεταβολή μιας εισόδου έως την επακόλουθη μεταβολή της εξόδου για έναν απομονωτή



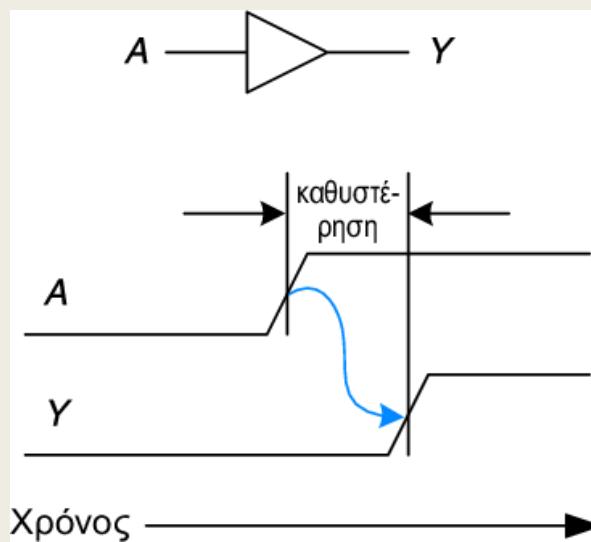
Χρονισμός: διάγραμμα χρονισμού

- Το παρακάτω σχήμα ονομάζεται **διάγραμμα χρονισμού**
 - απεικονίζει τη **μεταβατική απόκριση** (*transient response*) του κυκλώματος του απομονωτή όταν μεταβάλλεται μια είσοδος
 - Η μετάβαση από την τιμή *LOW* στην τιμή *HIGH* ονομάζεται **ανερχόμενη ακμή** (*rising edge*)
 - Η μετάβαση από την τιμή *HIGH* στην τιμή *LOW* ονομάζεται **κατερχόμενη ακμή** (*falling edge*)
(δεν φαίνεται στο σχήμα)



Χρονισμός: διάγραμμα χρονισμού

- Το **μπλε βέλος** υποδεικνύει ότι η ανερχόμενη ακμή της εξόδου Y προκαλείται από την ανερχόμενη ακμή της εισόδου A
 - Μετράμε την καθυστέρηση από το σημείο 50% του σήματος εισόδου A έως το σημείο 50% του σήματος εξόδου Y
 - Το σημείο 50% είναι εκείνο στο οποίο το σήμα βρίσκεται ακριβώς στη μέση (50%) της μετάβασης από την τιμή *LOW* στην τιμή *HIGH* και αντίστροφα



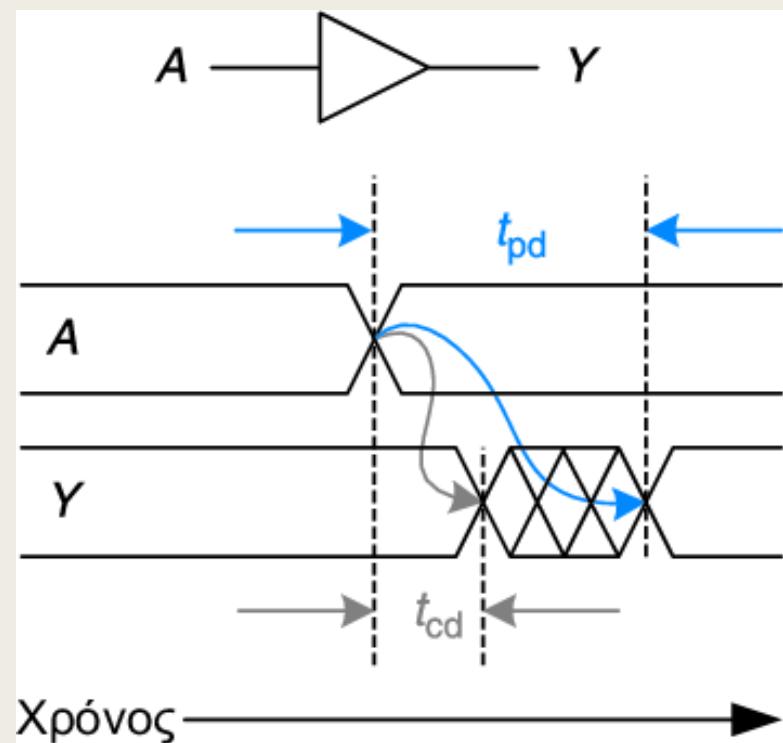
Καθυστέρηση διάδοσης / μόλυνσης

- Η συνδυαστική λογική χαρακτηρίζεται από
 - την **καθυστέρηση διάδοσης** t_{pd} (*propagation delay*) και
 - την **καθυστέρηση μόλυνσης** t_{cd} (*contamination delay*)

Η είσοδος A έχει αρχική τιμή είτε HIGH είτε LOW, και μεταβαίνει στην άλλη τιμή σε μια συγκεκριμένη στιγμή

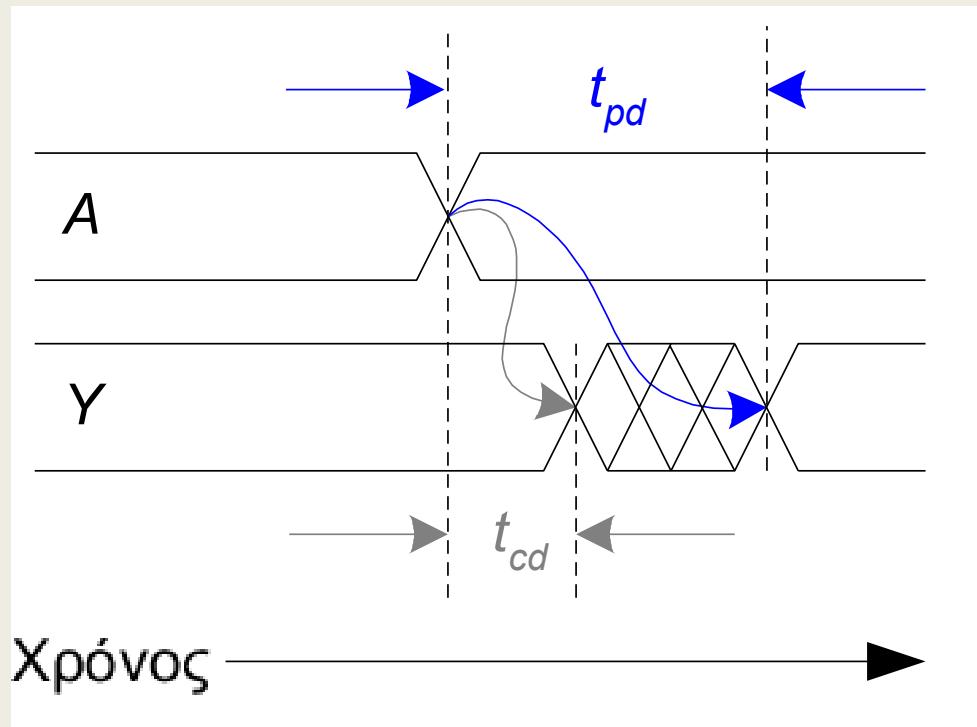
- δεν μας ενδιαφέρει η ακριβή τιμή

Αντιδρώντας στη μεταβολή, η έξοδος Y αλλάζει μετά από κάποιο χρονικό διάστημα.



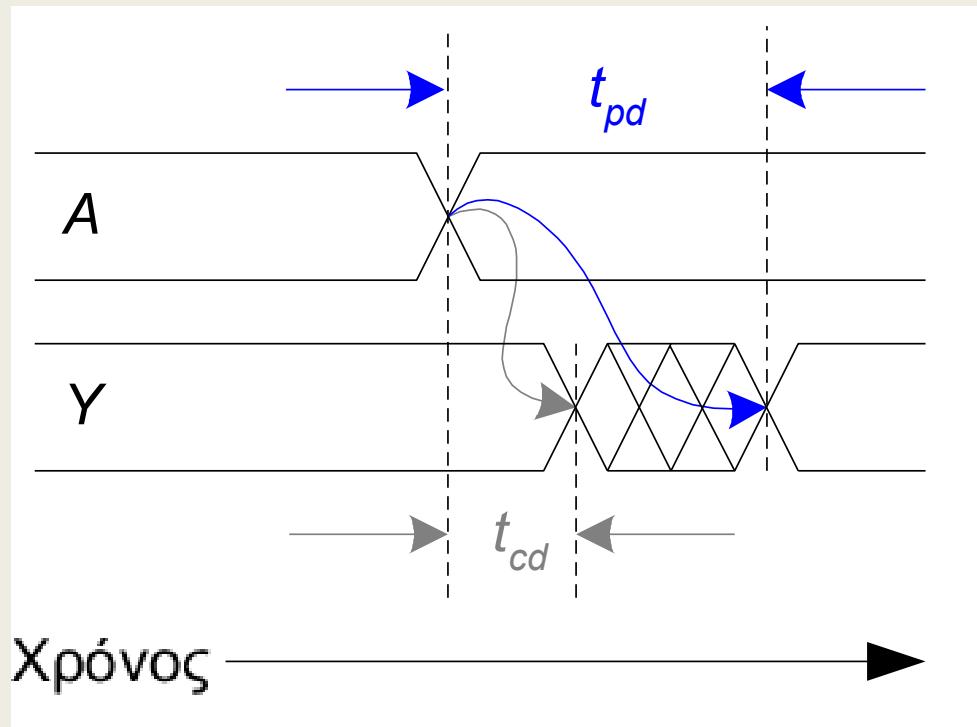
Καθυστέρηση διάδοσης

- Η **καθυστέρηση διάδοσης t_{pd}** είναι **ο μέγιστος χρόνος** που μεσολαβεί από τη στιγμή που μεταβάλλεται μια είσοδος έως τη στιγμή που η έξοδος ή οι έξοδοι αποκτούν την τελική τιμή τους
 - είναι η **χειρότερη περίπτωση καθυστέρησης** (αυτό που ονομάζουμε απλά καθυστέρηση)



Καθυστέρηση μόλυνσης

- Η **καθυστέρηση μόλυνσης t_{cd}** είναι ο **ελάχιστος χρόνος** που μεσολαβεί από τη στιγμή που μεταβάλλεται μια είσοδος έως τη στιγμή που οποιαδήποτε έξοδος αρχίζει να μεταβάλλει την τιμή της
 - είναι η **καλλίτερη περίπτωση καθυστέρησης**
(τη χρησιμοποιούμε σε συγκεκριμένες περιπτώσεις)

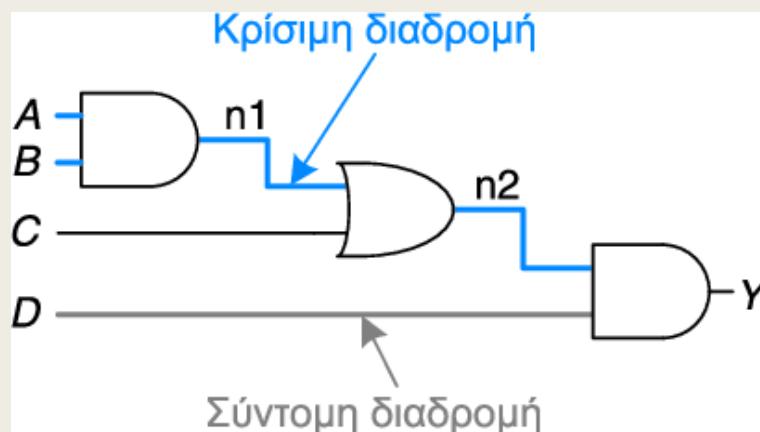


Καθυστέρηση διάδοσης / μόλυνσης

- **Αίτια καθυστέρησης σε επίπεδο πύλης**
 - ο χρόνος που απαιτείται για τη φόρτιση της χωρητικότητας
 - περισσότερα τρανζίστορ στη σειρά μεγαλύτερη καθυστέρηση
 - περισσότερες είσοδοι στην πύλη μεγαλύτερη καθυστέρηση
- **Οι τιμές των t_{pd} και t_{cd} ενδέχεται να είναι διαφορετικές**
 - διαφορετικές καθυστερήσεις ανερχόμενης και κατερχόμενης ακμής
 - Διαφορετική καθυστέρηση στα n-MOS από τα p-MOS τρανζίστορ
 - Διαφορετικό πλήθος τρανζίστορ στη σειρά στα n-MOS από τα p-MOS δίκτυα
 - παρουσία περισσότερων από μίας εισόδων και εξόδων, κάποιες από τις οποίες εμφανίζουν πιο αργές ή πιο κρίσιμες διαδρομές
 - αύξηση και μείωση της καθυστέρησης όταν τα κυκλώματα θερμαίνονται και ψύχονται, αντίστοιχα
- **Αίτια καθυστέρησης σε επίπεδο κυκλώματος**
 - η **διαδρομή** που ακολουθεί το σήμα μέσα από τις πύλες, όπως διαδίδεται από τις εισόδους στις εξόδους του κυκλώματος
 - τα σήματα διαδίδονται κατά μήκος των συρμάτων με την **ταχύτητα του φωτός**, με τη μορφή ηλεκτρομαγνητικών κυμάτων ($\sim 15 \text{ cm/ns}$)

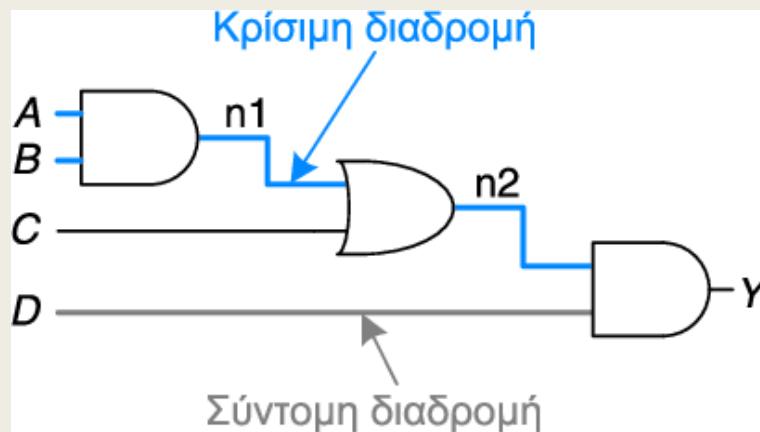
Κρίσιμη (αργή) διαδρομή

- Η **κρίσιμη (αργή) διαδρομή** είναι η μεγαλύτερη σε μήκος και συνεπώς η πιο αργή σε ταχύτητα
 - αυτή που διέρχεται από τις περισσότερες πύλες, που έχουν και τις περισσότερες εισόδους
 - ονομάζεται κρίσιμη επειδή **περιορίζει τη συχνότητα λειτουργίας του κυκλώματος**
- Στο σχήμα ξεκινάει από την είσοδο A ή B και καταλήγει στην έξοδο Y περνώντας μέσα από τρεις πύλες



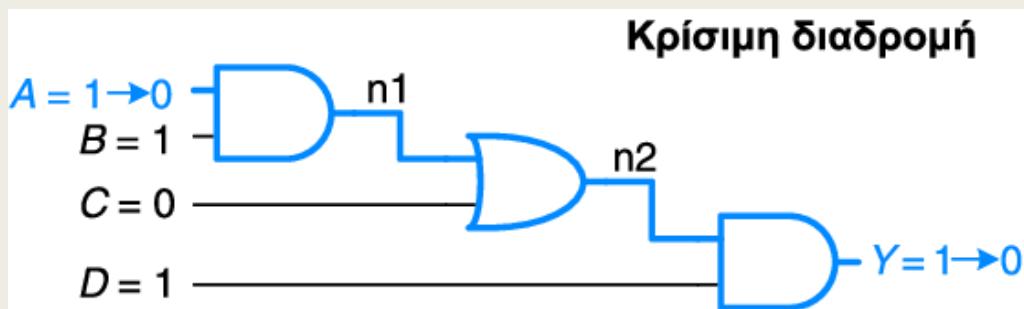
Σύντομη διαδρομή

- Η **σύντομη διαδρομή** είναι η μικρότερη σε μήκος και συνεπώς η πιο γρήγορη σε ταχύτητα
 - αυτή που διέρχεται από τις λιγότερες πύλες, που έχουν και τις λιγότερες εισόδους
- Στο σχήμα ξεκινάει από την είσοδο D και καταλήγει στην έξοδο Y περνώντας μέσα από μία πύλη

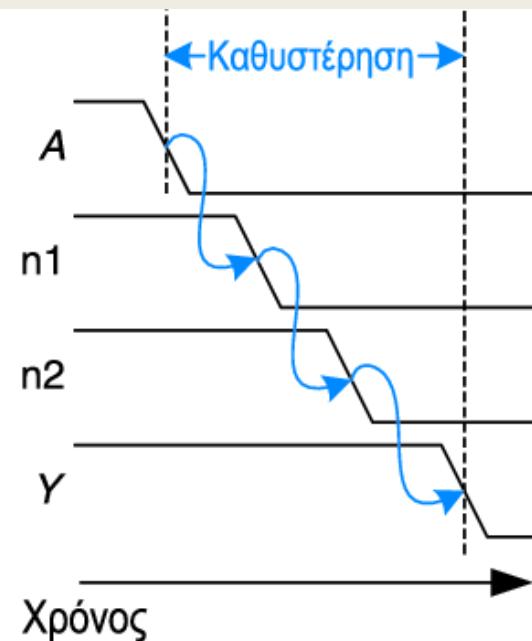


Υπολογισμός της καθυστέρησης t_{pd}

- Η **καθυστέρηση διάδοσης** t_{pd} ενός συνδυαστικού κυκλώματος είναι το άθροισμα των καθυστερήσεων διάδοσης για τη διέλευση από κάθε στοιχείο της **κρίσιμης (αργής) διαδρομής**
 - Στο σχήμα: $t_{pd} = 2t_{pd_AND} + t_{pd_OR}$
 - Εμφανίζεται όταν οι είσοδοι του κυκλώματος μεταβάλλονται από $ABCD = 1101$ σε $ABCD = 0101$

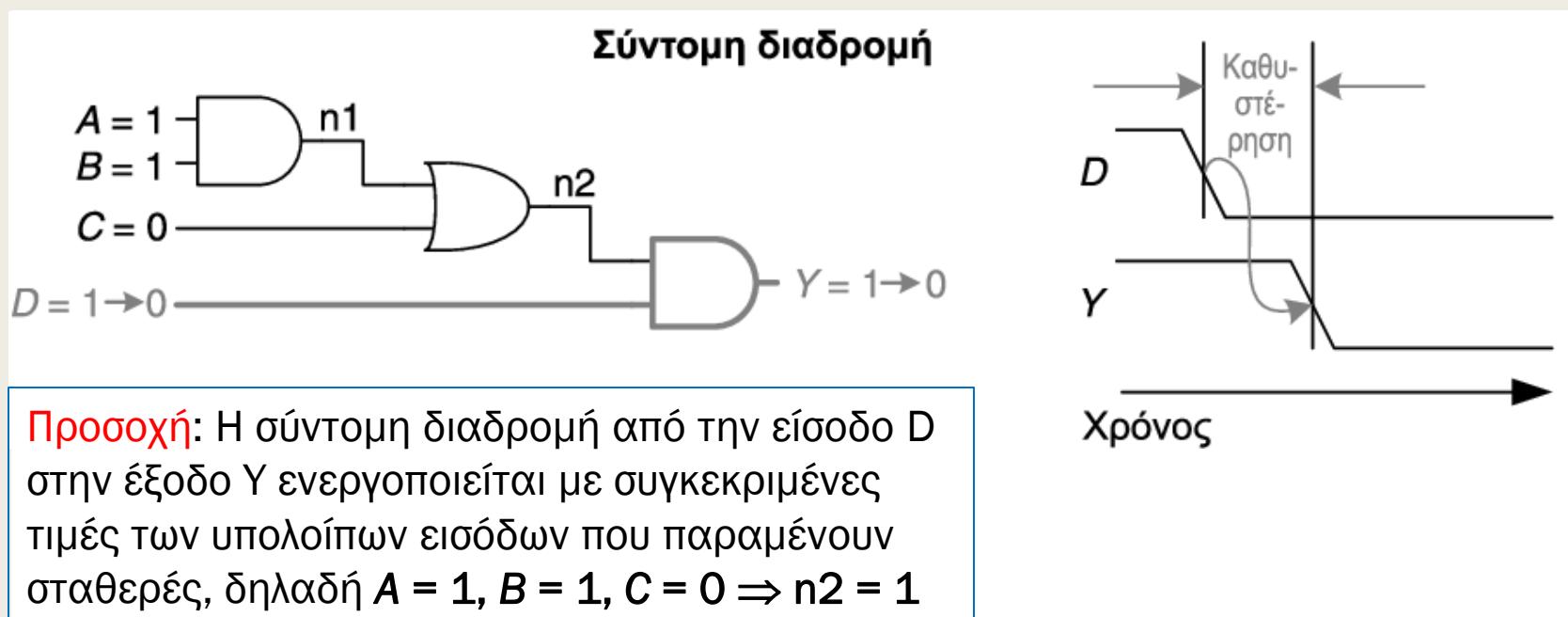


Προσοχή: Η κρίσιμη διαδρομή από την είσοδο A στην έξοδο Y ενεργοποιείται με συγκεκριμένες τιμές των υπολοίπων εισόδων που παραμένουν σταθερές, δηλαδή $B=1$, $C = 0$, $D = 1$



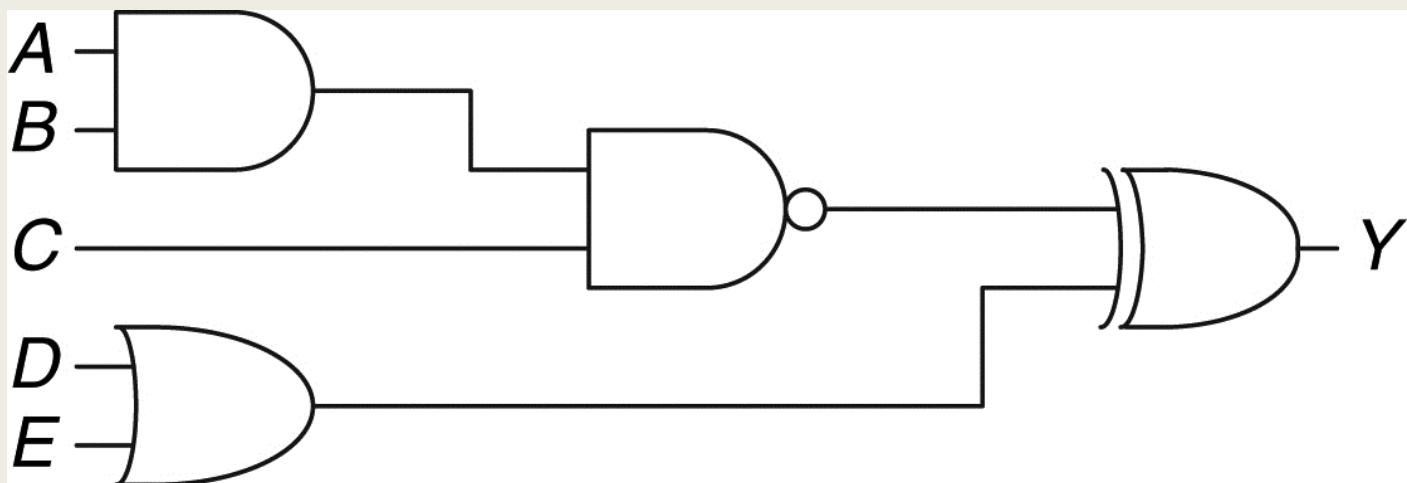
Υπολογισμός της καθυστέρησης t_{cd}

- Η **καθυστέρηση μόλυνσης t_{cd}** ενός συνδυαστικού κυκλώματος είναι το άθροισμα των καθυστερήσεων μόλυνσης για τη διέλευση από κάθε στοιχείο **της σύντομης διαδρομής**
 - Στο σχήμα: $t_{cd} = t_{cd_AND}$
 - Εμφανίζεται όταν οι είσοδοι του κυκλώματος μεταβάλλονται από $ABCD = 1101$ σε $ABCD = 1100$



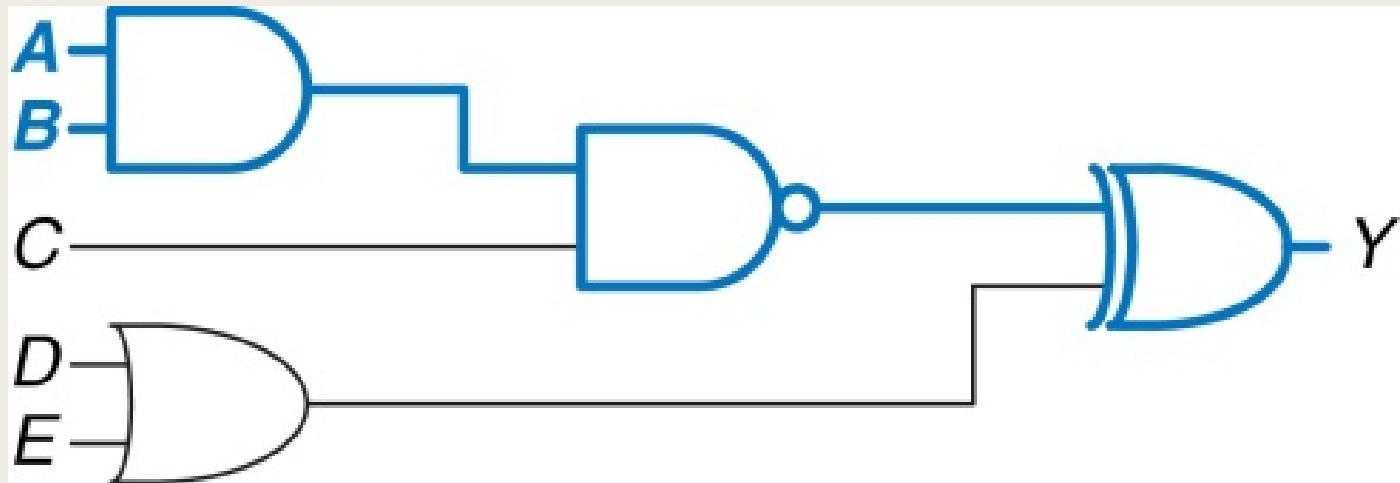
Παράδειγμα 2.15

- **Υπολογισμός καθυστερήσεων διάδοσης και μόλυνσης**
- Υπολογίστε την καθυστέρηση διάδοσης και την καθυστέρηση μόλυνσης του κυκλώματος
- Κάθε πύλη έχει καθυστέρηση διάδοσης ίση με **100 ps** και καθυστέρηση μόλυνσης ίση με **60 ps** (**ps = picoseconds**)



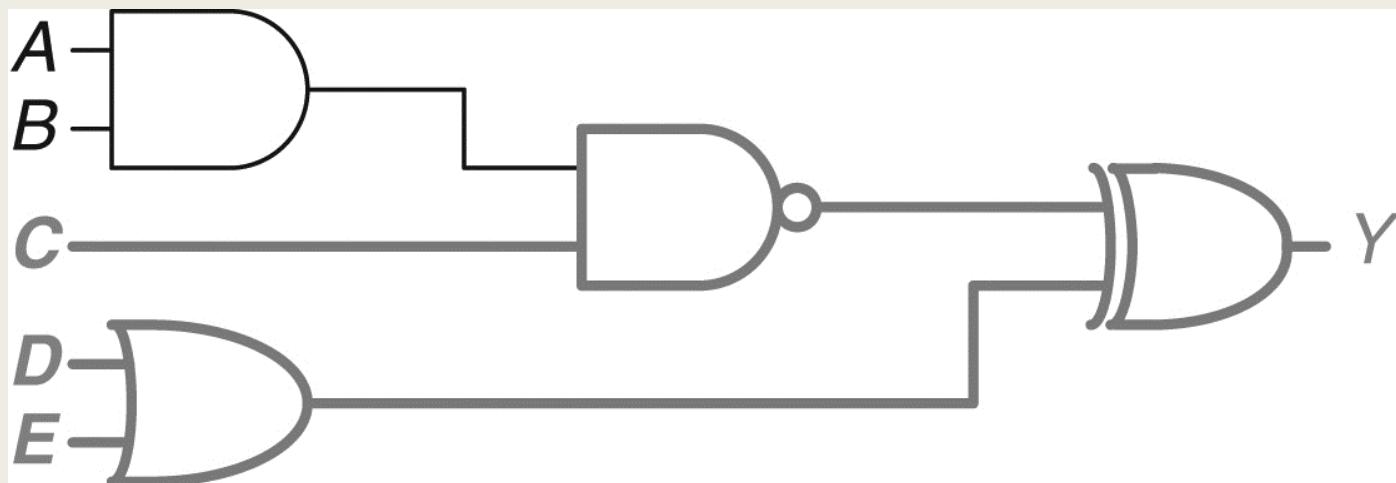
Παράδειγμα 2.15

- **Υπολογισμός καθυστερήσεων διάδοσης και μόλυνσης**
- Υπολογισμός της καθυστέρησης διάδοσης του κυκλώματος
 - Κάθε πύλη έχει καθυστέρηση διάδοσης ίση με **100 ps**
 - Η κρίσιμη διαδρομή αρχίζει από την είσοδο A ή B, διέρχεται από τρεις πύλες και καταλήγει στην έξοδο Y
 - Άρα, η καθυστέρηση διάδοσης είναι: $t_{pd} = 3 \times 100 = 300 \text{ ps}$



Παράδειγμα 2.15

- **Υπολογισμός καθυστερήσεων διάδοσης και μόλυνσης**
- Υπολογισμός της καθυστέρησης μόλυνσης του κυκλώματος
 - Κάθε πύλη έχει καθυστέρηση μόλυνσης ίση με **60 ps**
 - Η σύντομη διαδρομή αρχίζει από την είσοδο C ή D ή E, διέρχεται από δύο πύλες και καταλήγει στην έξοδο Y
 - Άρα, η καθυστέρηση μόλυνσης είναι: $t_{cd} = 2 \times 60 = 120 \text{ ps}$



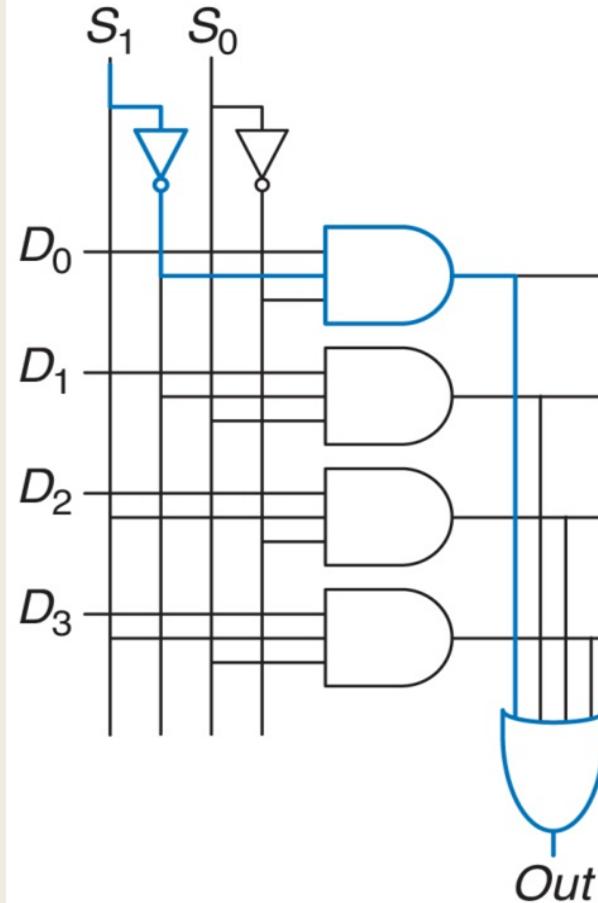
Παράδειγμα 2.16

- **Χρονισμός πολυπλεκτών: κρισιμότητα από πλευράς ελέγχου και κρισιμότητα από πλευράς δεδομένων**
- Συγκρίνετε τον χρονισμό της χειρότερης περίπτωσης (εύρεση κρίσιμης διαδρομής και υπολογισμός καθυστέρησης διάδοσης) για τις τρεις διαφορετικές υλοποιήσεις των πολυπλεκτών 4 σε 1
 - (α) λογική αθροίσματος γινομένων,
 - (β) απομονωτές τριών καταστάσεων,
 - (γ) δένδρο πολυπλεκτών 2 σε 1
- Δίδονται οι τιμές της καθυστέρησης διάδοσης για τα επιμέρους στοιχεία

Πύλη	t_{pd} (ps)
NOT	30
AND με 2 εισόδους	60
AND με 3 εισόδους	80
OR με 4 εισόδους	90
απομονωτής τριών καταστάσεων (από A έως Y)	50
απομονωτής τριών καταστάσεων (από enable έως Y)	35

Παράδειγμα 2.16

- **Χρονισμός πολυπλεκτών:** κρισιμότητα από πλευράς ελέγχου και κρισιμότητα από πλευράς δεδομένων
- Λογική αθροίσματος γινομένων
- Η κρίσιμη διαδρομή αρχίζει από την είσοδο επιλογής S_1 ή S_0 , διέρχεται από 3 πύλες (NOT, AND-3, OR-4) και καταλήγει στην έξοδο Y
- Η υλοποίηση είναι κρίσιμη από πλευράς ελέγχου
 - Έλεγχος-έξοδος 200 ps
 - Δεδομένα-έξοδος 170 ps

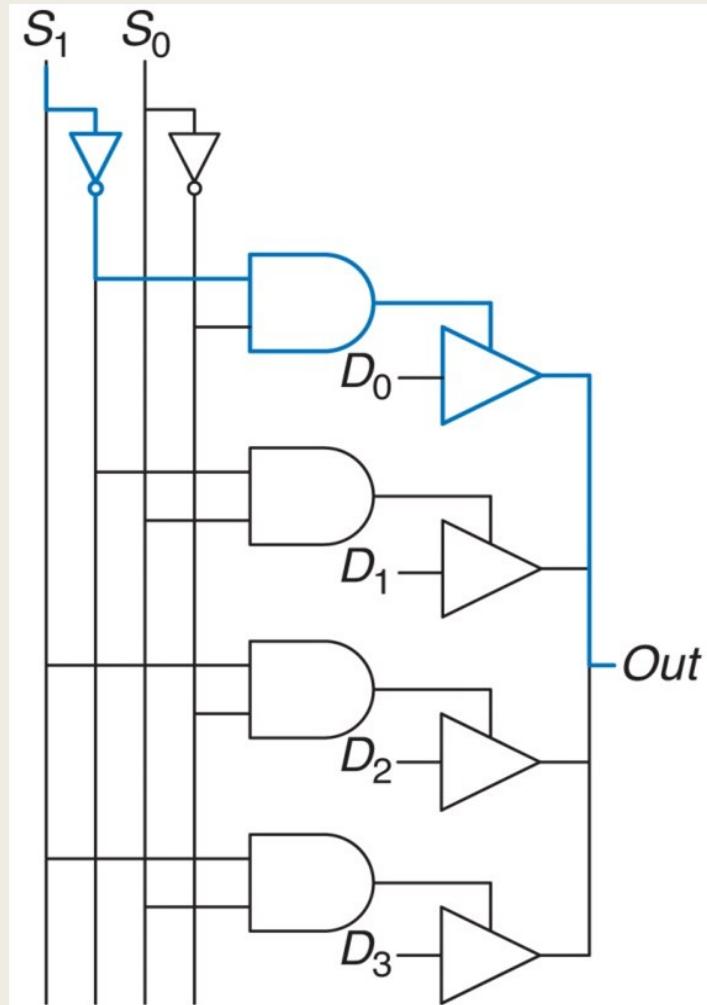


$$\begin{aligned}t_{pd_sy} &= t_{pd_INV} + t_{pd_AND3} + t_{pd_OR4} \\&= 30 \text{ ps} + 80 \text{ ps} + 90 \text{ ps} \\&= \mathbf{200 \text{ ps}}\end{aligned}$$

$$\begin{aligned}t_{pd_dy} &= t_{pd_AND3} + t_{pd_OR4} \\&= \mathbf{170 \text{ ps}}\end{aligned}$$

Παράδειγμα 2.16

- **Χρονισμός πολυπλεκτών:** κρισιμότητα από πλευράς ελέγχου και κρισιμότητα από πλευράς δεδομένων
- Απομονωτές τριών καταστάσεων
- Η κρίσιμη διαδρομή αρχίζει από την είσοδο επιλογής S_1 ή S_0 , διέρχεται από 2 πύλες (NOT, AND-2) και τον απομονωτή τριών καταστάσεων (enable to Y), και καταλήγει στην έξοδο Y
- Η υλοποίηση είναι κρίσιμη από πλευράς ελέγχου
 - Έλεγχος-έξοδος **125 ps**
 - Δεδομένα-έξοδος **50 ps**

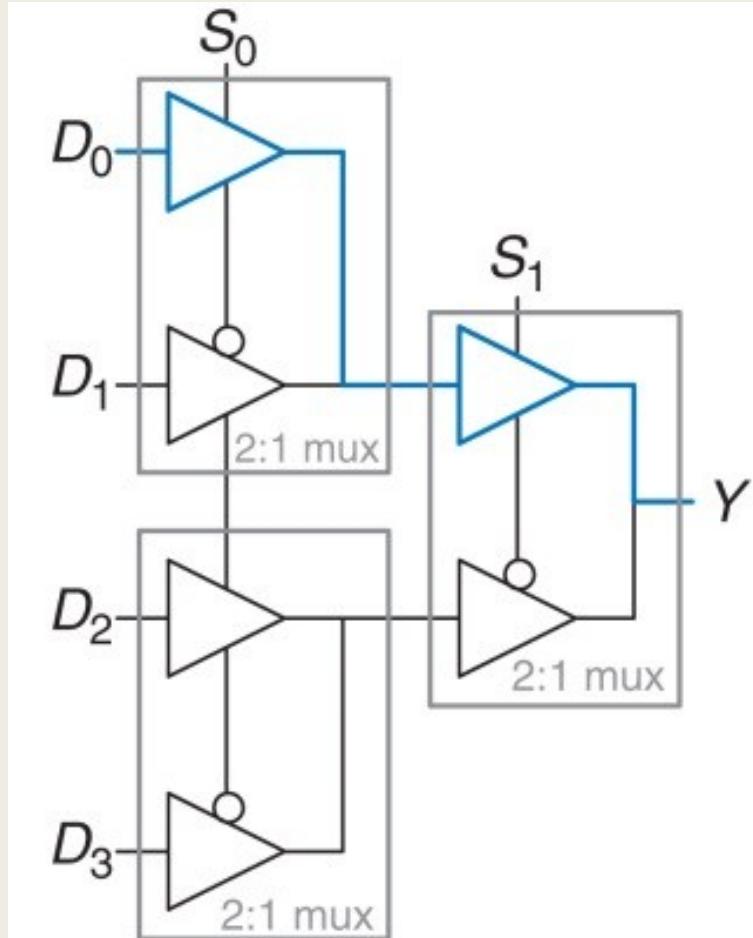


$$\begin{aligned}t_{pd_sy} &= t_{pd_INV} + t_{pd_AND2} + t_{pd_TRI_sy} \\&= 30 \text{ ps} + 60 \text{ ps} + 35 \text{ ps} \\&= \mathbf{125 \text{ ps}}\end{aligned}$$

$$\begin{aligned}t_{pd_dy} &= t_{pd_TRI_ay} \\&= \mathbf{50 \text{ ps}}\end{aligned}$$

Παράδειγμα 2.16

- Χρονισμός πολυπλεκτών:
κρισιμότητα από πλευράς
ελέγχου και κρισιμότητα
από πλευράς δεδομένων
- Δένδρο πολυπλεκτών 2 σε 1
- Οι πολυπλέκτες 2 σε 1
υλοποιούνται με απομονωτές
τριών καταστάσεων
- Η κρίσιμη διαδρομή αρχίζει από
οποιαδήποτε είσοδο D,
διέρχεται από 2 απομονωτές
τριών καταστάσεων (A to Y), και
καταλήγει στην έξοδο Y
- Η υλοποίηση είναι κρίσιμη
από πλευράς δεδομένων
 - Έλεγχος-έξοδος 85 ps
 - Δεδομένα-έξοδος 100 ps



$$t_{pd_s0y} = t_{pd_TRI_sy} + t_{pd_TRI_ay} = 85 \text{ ps}$$
$$t_{pd_dy} = 2 t_{pd_TRI_ay} = 100 \text{ ps}$$

Παράδειγμα 2.16

- **Χρονισμός πολυπλεκτών: κρισιμότητα από πλευράς ελέγχου και κρισιμότητα από πλευράς δεδομένων**
- Συμπεράσματα:
 - Αν τα σήματα από τις εισόδους δεδομένων φτάνουν αρκετά πριν από εκείνα των εισόδων ελέγχου, επιλέγουμε την υλοποίηση με τη μικρότερη καθυστέρηση διάδοσης ελέγχου-προς-έξοδο (**85 ps**)
 - Την υλοποίηση του **δένδρου πολυπλεκτών 2 σε 1**
 - Αν τα σήματα από τις εισόδους ελέγχου φτάνουν αρκετά πριν από εκείνα των εισόδων δεδομένων, επιλέγουμε την υλοποίηση με τη μικρότερη καθυστέρηση δεδομένων-προς-έξοδο (**50 ps**)
 - Την υλοποίηση των **απομονωτών τριών καταστάσεων**
- Η βέλτιστη επιλογή δεν εξαρτάται μόνο από την κρίσιμη διαδρομή μέσω του κυκλώματος και τους χρόνους άφιξης των σημάτων εισόδου, αλλά και από την κατανάλωση ισχύος, το κόστος και τη διαθεσιμότητα των στοιχείων

Επιλεγμένες ασκήσεις

■ Άσκηση 2.44

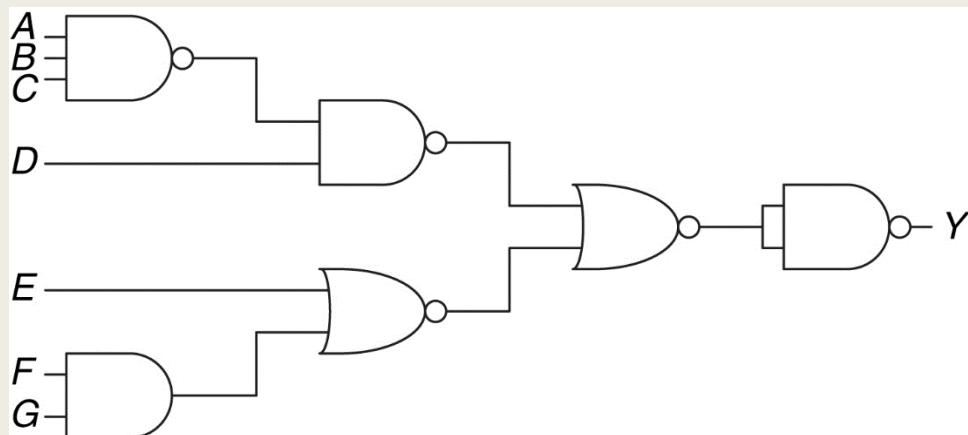
Υπολογίστε την καθυστέρηση διάδοσης και την καθυστέρηση μόλυνσης του κυκλώματος της εικόνας. Χρησιμοποιήστε τις καθυστερήσεις πυλών που δίνονται στον Πίνακα

- Κρίσιμη διαδρομή: F-Y

$$\begin{aligned} t_{pd} &= t_{pd_AND2} + 2t_{pd_NOR2} + t_{pd_NAND2} \\ &= [30 + 2(30) + 20] \text{ ps} \\ &= 110 \text{ ps} \end{aligned}$$

- Σύντομη διαδρομή: D-Y

$$\begin{aligned} t_{cd} &= 2t_{cd_NAND2} + t_{cd_NOR2} \\ &= [2(15) + 25] \text{ ps} \\ &= 55 \text{ ps} \end{aligned}$$



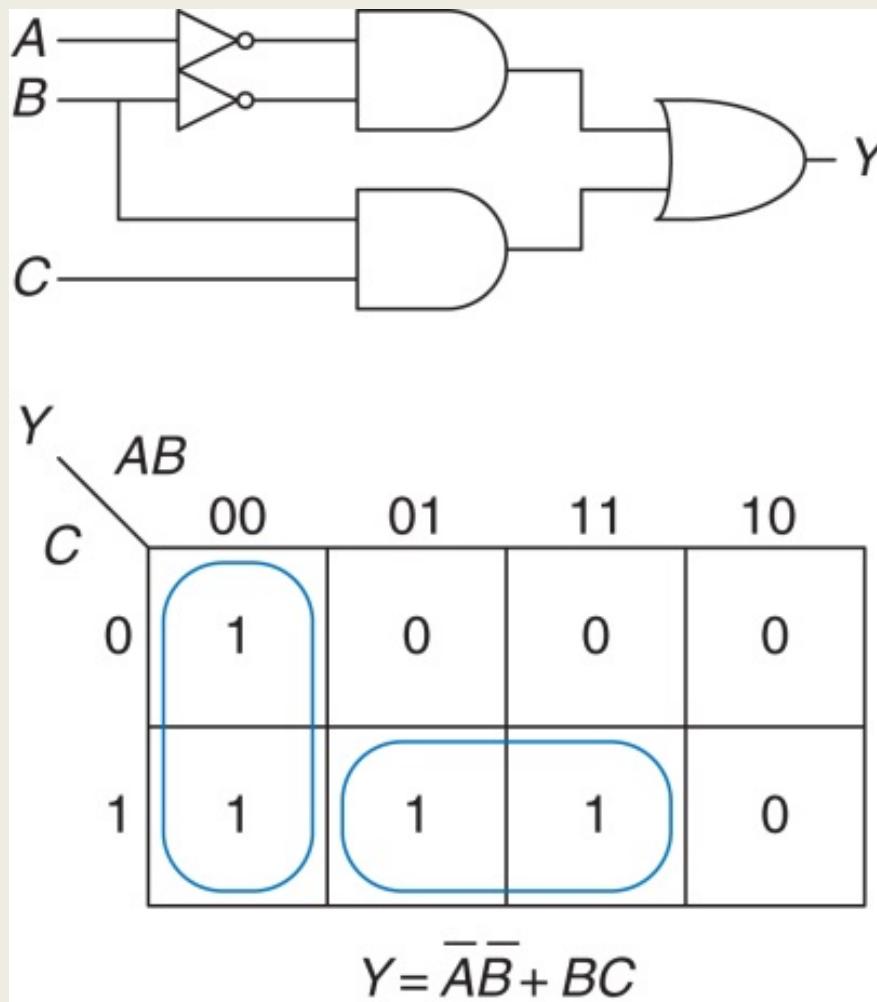
Πύλη	t_{pd} (ps)	t_{cd} (ps)
NOT	15	10
NAND με 2 εισόδους	20	15
NAND με 3 εισόδους	30	25
NOR με 2 εισόδους	30	25
NOR με 3 εισόδους	45	35
AND με 2 εισόδους	30	25
AND με 3 εισόδους	40	30
OR με 2 εισόδους	40	30
OR με 3 εισόδους	55	45
XOR με 2 εισόδους	60	40

Μεταβατικοί παλμοί (Glitches)

- Μέχρι τώρα έχουμε μελετήσει την περίπτωση μία μεταβολή εισόδου να προκαλεί μονάχα μία μεταβολή στην έξοδο
- Είναι όμως πιθανό μία μεταβολή εισόδου να προκαλέσει **πολλαπλές μεταβολές στην έξοδο**
 - Αυτές ονομάζονται **μεταβατικοί παλμοί** (*glitches*) ή **κίνδυνοι** (*hazards*)
- Με την προϋπόθεση ότι περιμένουμε να παρέλθει η **κρίσιμη καθυστέρηση διάδοσης** προτού βρεθούμε εξαρτημένοι από την έξοδο, οι μεταβατικοί παλμοί δεν αποτελούν πρόβλημα επειδή η έξοδος θα έχει προλάβει να σταθεροποιηθεί στη σωστή λογική τιμή
- Μας ενδιαφέρει να γνωρίζουμε ότι υπάρχουν στα περισσότερα ψηφιακά κυκλώματα και να τα διακρίνουμε όταν εξετάζουμε διαγράμματα χρονισμού σε προσομοιωτές ή παλμογράφους

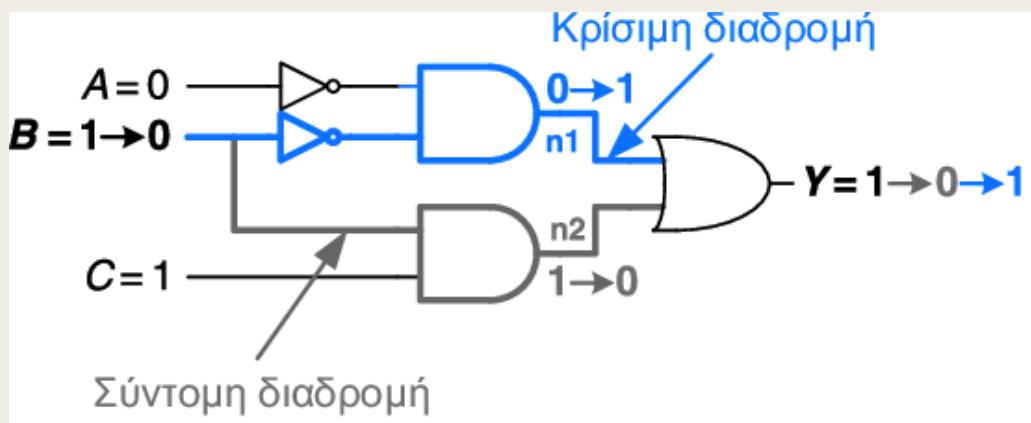
Μεταβατικοί παλμοί (Glitches)

- Τι θα συμβεί όταν $A = 0$, $C = 1$ και B μεταβαίνει από το 1 στο 0



Μεταβατικοί παλμοί (Glitches)

- Η είσοδος B συνδέεται με την έξοδο Y μέσω δύο διαδρομών:
 - Την **κρίσιμη διαδρομή** που διέρχεται από τρείς πύλες (NOT , $AND-2$ και $OR-2$)
 - Τη **σύντομη διαδρομή** που διέρχεται από δύο πύλες ($AND-2$ και $OR-2$)
 - Η κρίσιμη διαδρομή διαφέρει από τη σύντομη διαδρομή στην καθυστέρηση διάδοσης **κατά μία πύλη NOT**
- Λόγω αυτής της διαφοράς εμφανίζεται ένας **μεταβατικός παλμός** στην έξοδο Y , όταν $A = 0$, $C = 1$ και B μεταβαίνει από το 1 στο 0



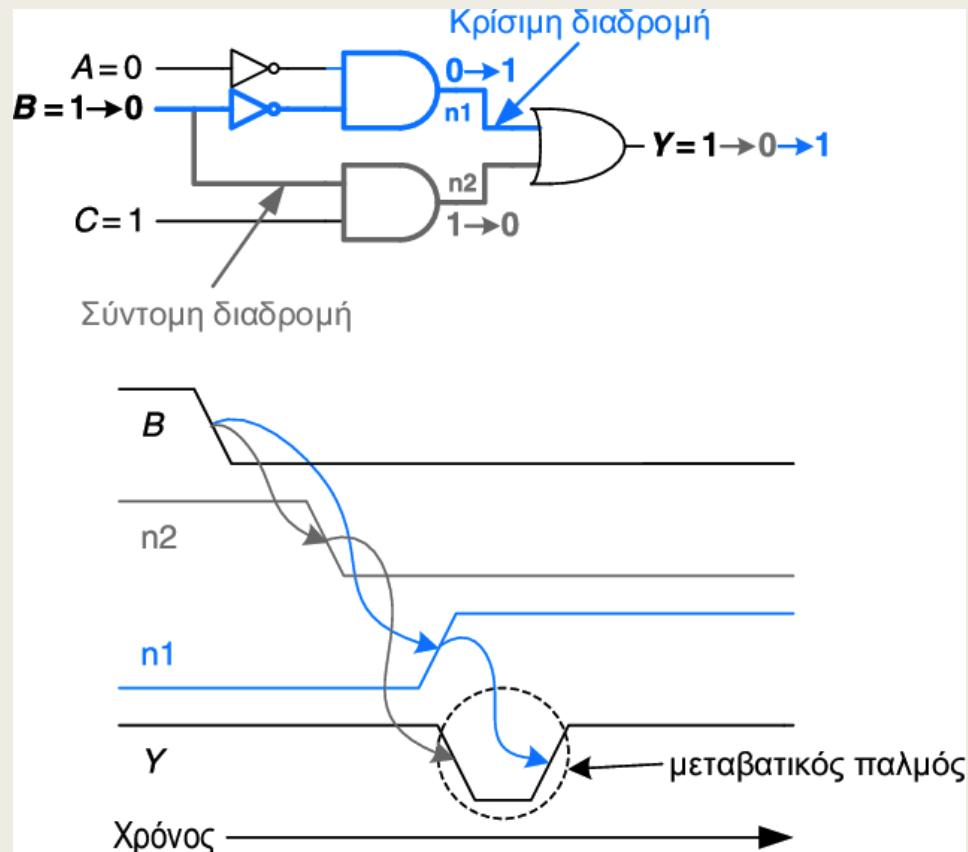
Μεταβατικοί παλμοί (Glitches)

Καθώς το B μεταβαίνει από το 1 στο 0, ο κόμβος $n2$ (στη σύντομη διαδρομή) υφίσταται κάθιδο προτού προλάβει να ολοκληρωθεί η άνοδος του κόμβου $n1$ (στην κρίσιμη διαδρομή)

Μέχρι να ολοκληρωθεί η άνοδος του $n1$, οι δύο είσοδοι της πύλης OR έχουν τιμή 0, και η έξοδος Y «πέφτει» στο 0

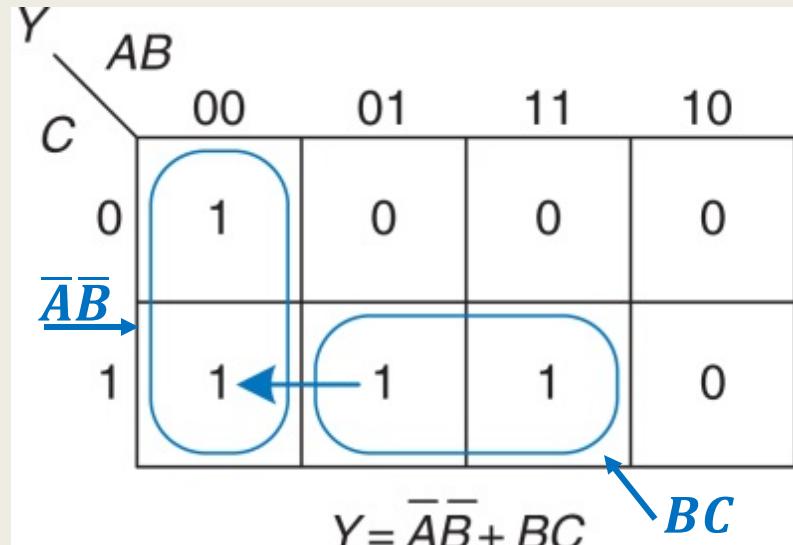
Μόλις ολοκληρωθεί η άνοδος του $n1$, το Y επιστρέφει στο 1

Η διάρκεια του μεταβατικού παλμού είναι η καθυστέρηση διάδοσης μίας πύλης NOT



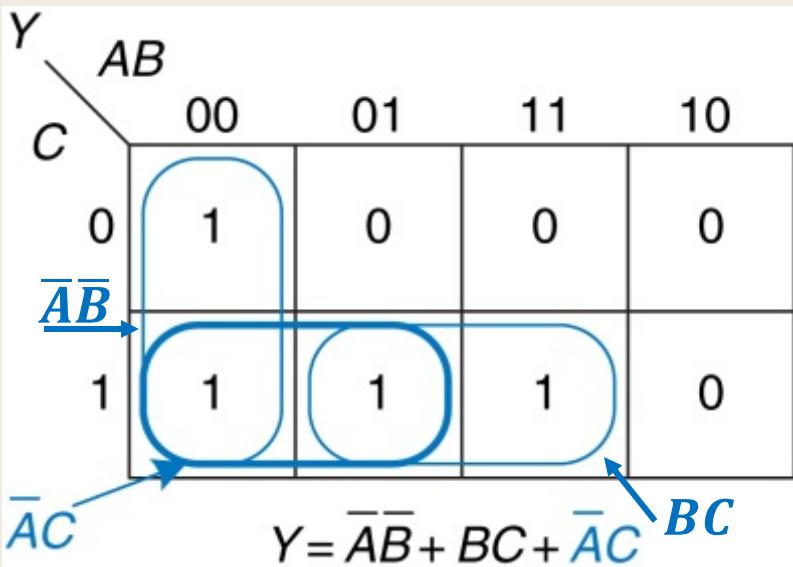
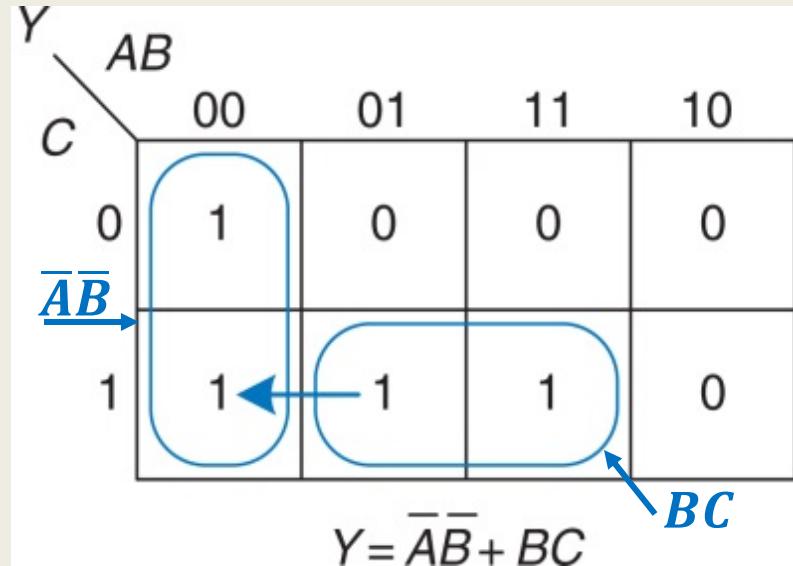
Αποφυγή μεταβατικού παλμού

- Κατά τη μετάβαση των εισόδων ABC από το 011 στο 001, αν το κύκλωμα του πρώτου όρου (BC) απενεργοποιηθεί προτού το κύκλωμα του άλλου πρώτου όρου ($\bar{A}\bar{B}$) ενεργοποιηθεί, τότε παρατηρείται ένας μεταβατικός παλμός
- Η μετάβαση που διασχίζει τα όρια δύο πρώτων όρων στον χάρτη Karnaugh αποτελεί ένδειξη πιθανού μεταβατικού παλμού



Αποφυγή μεταβατικού παλμού

- Ο μεταβατικός παλμός αποφεύγεται εάν προσθέσουμε έναν επιπλέον κύκλο του οποίου ο όρος $\bar{A}C$ είναι **ανάμεσα** στους πρώτους όρους BC και $\bar{A}\bar{B}$
- Πρόκειται για το **Θεώρημα της ομοφωνίας**, όπου ο επιπλέον προστιθέμενος όρος, το $\bar{A}C$, είναι ο **πλεονάζων όρος**



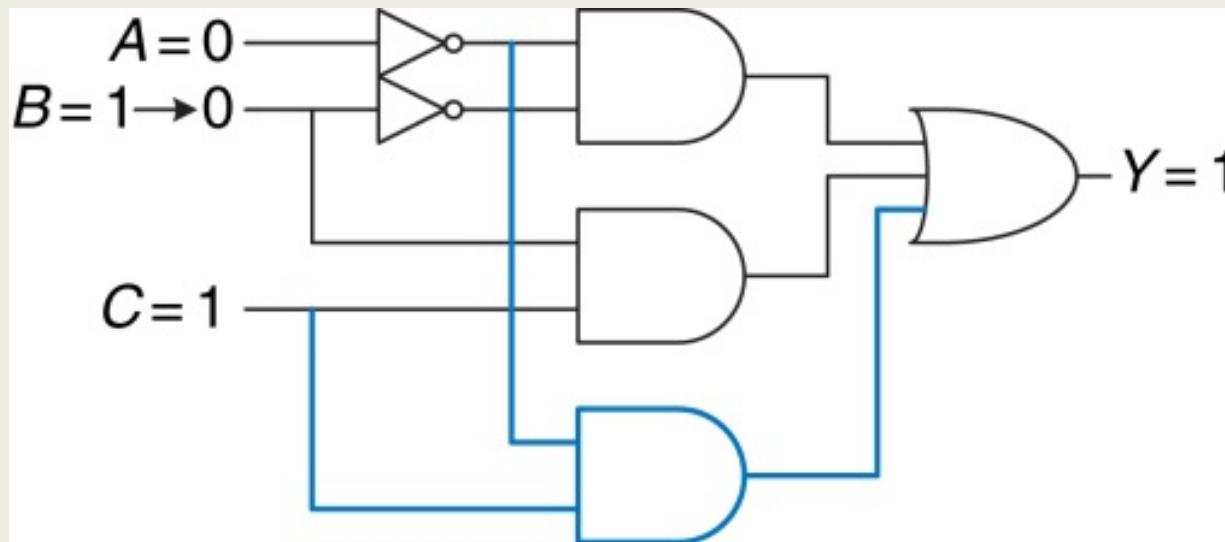
Αποφυγή μεταβατικού παλμού

- Ο μεταβατικός παλμός αποφεύγεται γιατί η πύλη AND που υλοποιεί τον **πλεονάζοντα όρο $\bar{A}C$** παράγει στην έξοδό της την τιμή 1 καθ' όλη τη διάρκεια της μεταβολής του B

Y	AB	00	01	11	10
C	0	1	0	0	0
$\bar{A}\bar{B}$	1	1	1	1	0
$\bar{A}C$	1	1	1	1	0

$$Y = \bar{A}\bar{B} + BC + \bar{A}C$$

BC



Κεφάλαιο 3

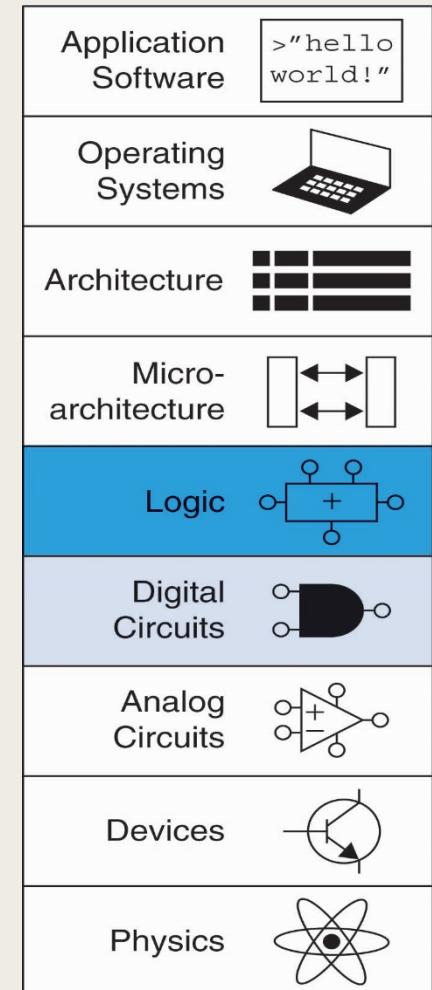
Σχεδίαση ακολουθιακής λογικής

Γιώργος Παπαδημητρίου, Αντώνης Πασχάλης,
Διονύσης Βασιλόπουλος



Περιεχόμενα κεφαλαίου 3

- Εισαγωγή
- Διασυζευγμένοι αντιστροφείς
- Latches και Flip-Flops
- Σύγχρονη ακολουθιακή λογική
- Μηχανές πεπερασμένων καταστάσεων (FSM)
 - Μηχανές Moore-Mealy
- Χρονισμός ακολουθιακής λογικής
- Παραλληλισμός

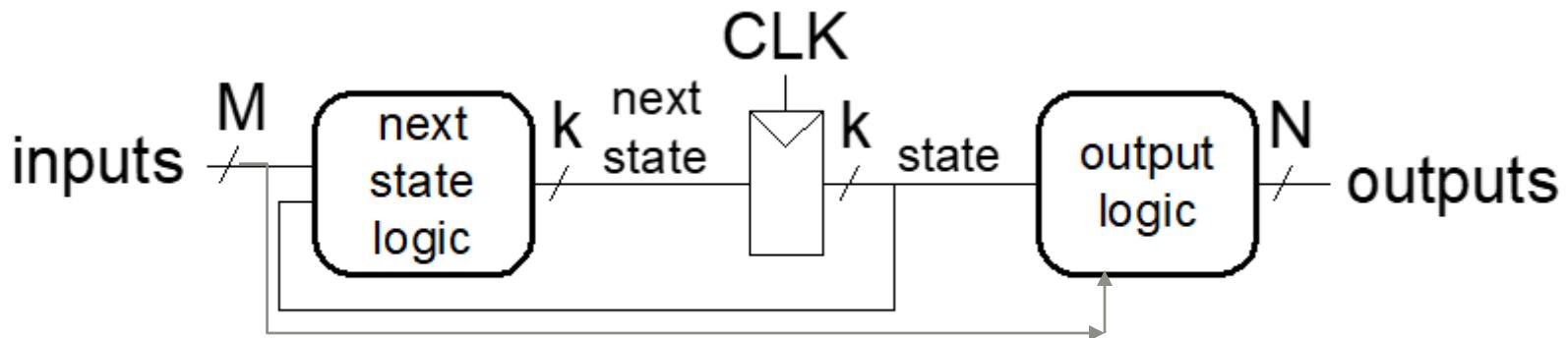


Ακολουθιακή λογική

- Οι έξοδοι των **ακολουθιακών κυκλωμάτων** εξαρτώνται όχι μόνο από τις τρέχουσες τιμές των εισόδων, αλλά και από τις προηγούμενες τιμές αυτών
- Τα ακολουθιακά κυκλώματα **έχουν μνήμη**, δηλαδή
 - «διυλίζουν» τις προηγούμενες τιμές εισόδων ώστε να διατηρούν αποθηκευμένη μία μικρότερη ποσότητα πληροφοριών, η οποία ονομάζεται **κατάσταση** (state)
 - για να επιτευχθεί η αποθήκευση μίας κατάστασης απαιτείται **ανάδραση από την έξοδο στην είσοδο**
- Ένα ακολουθιακό κύκλωμα με **N πιθανές καταστάσεις** διατηρεί αποθηκευμένη μία κατάσταση μεγέθους **από $\log_2 N$ μέχρι N bit**
 - *Tα latches και τα flip-flops* διατηρούν αποθηκευμένη μία κατάσταση μεγέθους ενός bit, δηλαδή έχουν δύο πιθανές καταστάσεις (0 και 1)
- Οι έξοδοι είναι συναρτήσεις των εισόδων και της αποθηκευμένης κατάστασης του κυκλώματος

Ακολουθιακή λογική

- Ακολουθιακά κυκλώματα
 - Οι έξοδοι εξαρτώνται από τις τρέχουσες και από τις προηγούμενες εισόδους
 - Αποθήκευση κατάστασης (*state*): μια αφαίρεση του ιστορικού των εισόδων
- Συνήθως, ελέγχεται από σήμα ρολογιού (*clock*)

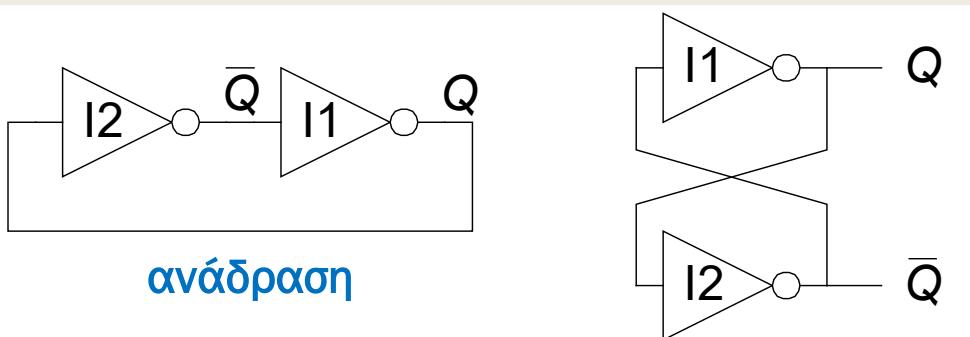


Ιδιότητες των κυκλωμάτων με μνήμη

- Ένα κύκλωμα που έχει μνήμη πρέπει να έχει τις εξής 3 ιδιότητες:
 - Πρέπει να μπορεί να **διατηρεί αποθηκευμένη μία κατάσταση**
 - Πρέπει να μπορούμε να **διαβάζουμε την αποθηκευμένη κατάσταση**
 - Πρέπει να μπορούμε να **αλλάζουμε την αποθηκευμένη κατάσταση σε μία νέα κατάσταση**
 - Η νέα κατάσταση είναι συνάρτηση της προηγούμενης κατάστασης και ενδεχομένως και των εισόδων του ακολουθιακού κυκλώματος

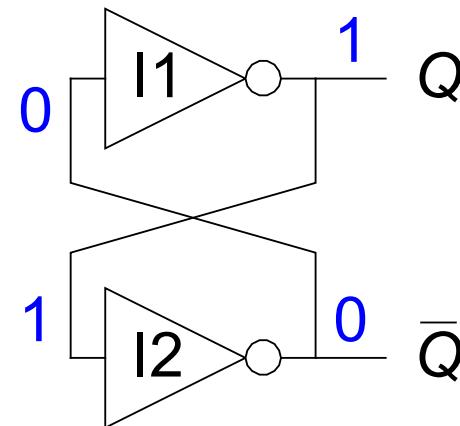
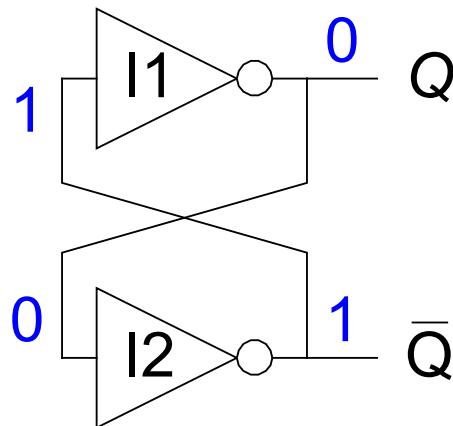
Διασυζευγμένοι αντιστροφείς

- Το θεμελιώδες δομικό στοιχείο μνήμης είναι ένα **δισταθές** (bistable) στοιχείο, δηλαδή ένα στοιχείο με δύο σταθερές καταστάσεις, το 0 και το 1
- Οι αντιστροφείς είναι **διασυζευγμένοι** (cross-coupled), που σημαίνει ότι η είσοδος του αντιστροφέα I_1 αποτελεί έξοδο του αντιστροφέα I_2 και αντίστροφα, ώστε να δημιουργείται ανάδραση
- Το κύκλωμα δεν διαθέτει εισόδους, αλλά έχει **δύο εξόδους, τις Q και \bar{Q}** (διαβάζεται Q bar) που είναι **συμπληρωματικές**
- Το κύκλωμα διατηρεί αποθηκευμένη μία **κατάσταση Q** μεγέθους ενός bit, που ταυτίζεται με την έξοδο Q
 - Μπορούμε να αναφερόμαστε και στην μεταβλητή κατάστασης Q που έχει ως τιμή την τιμή της εξόδου Q



Λειτουργία διασυζευγμένων αντιστροφέων

- Όταν βρίσκεται στην σταθερή κατάσταση $Q = 0$
τότε οι έξοδοι είναι: $Q = 0$ και $\bar{Q} = 1$
- Όταν βρίσκεται στην σταθερή κατάσταση $Q = 1$
τότε οι έξοδοι είναι: $Q = 1$ και $\bar{Q} = 0$

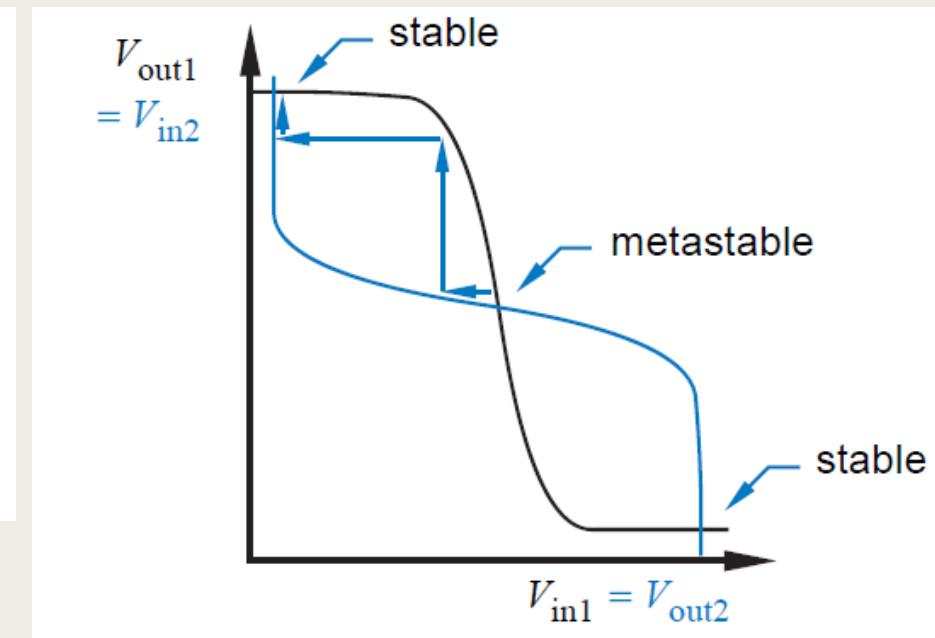
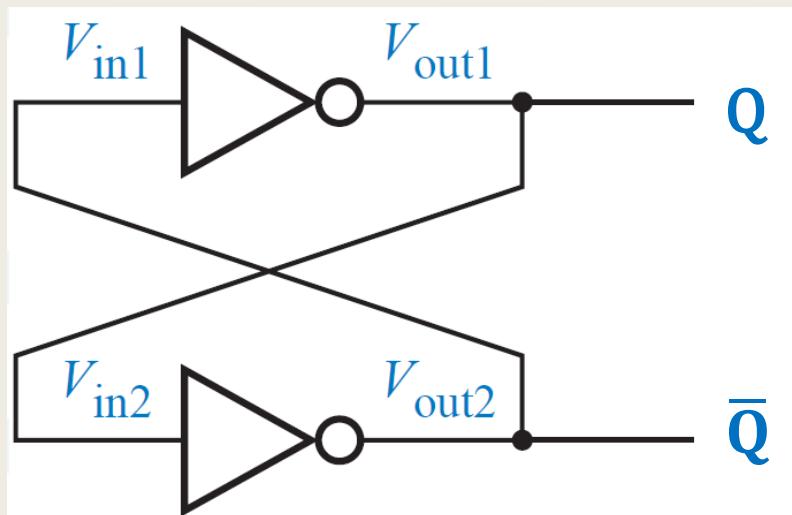


Λειτουργία διασυζευγμένων αντιστροφέων

- Το κύκλωμα των διασυζευγμένων αντιστροφέων έχει τις εξής ιδιότητες:
 - *Μπορεί να διατηρεί αποθηκευμένη μία κατάσταση μεγέθους ενός bit*
 - Την σταθερή κατάσταση 0 ή την σταθερή κατάσταση 1
 - *Μπορούμε να διαβάσουμε την αποθηκευμένη κατάσταση*
 - Εξετάζοντας την τιμή της **εξόδου Q**
 - *Δεν μπορούμε να αλλάξουμε την αποθηκευμένη κατάσταση σε μία νέα κατάσταση*
 - Το κύκλωμα **δεν έχει εισόδους**, αλλά όταν τίθεται σε λειτουργία πηγαίνει **ανεξέλεγκτα** στην σταθερή κατάσταση 0 ή στην σταθερή κατάσταση 1
 - Υπάρχει ενδεχόμενο για ένα χρονικό διάστημα να παραμείνει σε μία μια τρίτη πιθανή κατάσταση, την **μετασταθερή (metastable) κατάσταση**

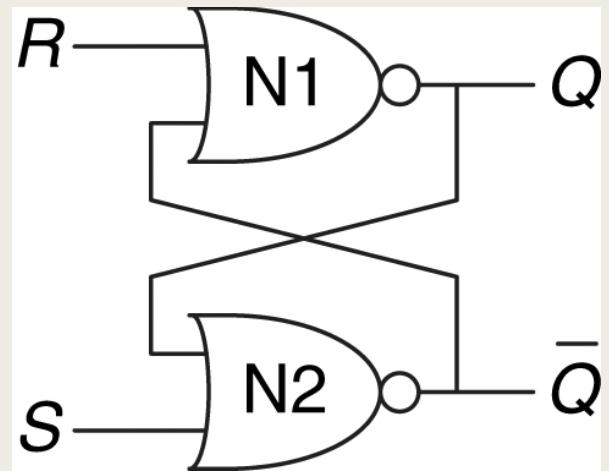
Μετασταθερή Κατάσταση

- Κατά την έναρξη της λειτουργίας του κυκλώματος των διασυζευγμένων αντιστροφέων, το κύκλωμα μπορεί να παραμείνει προσωρινά στην **μετασταθερή (metastable) κατάσταση**
 - όπου και οι δύο έξοδοι Q και \bar{Q} βρίσκονται κατά προσέγγιση στο μέσο μεταξύ του 0 και του 1



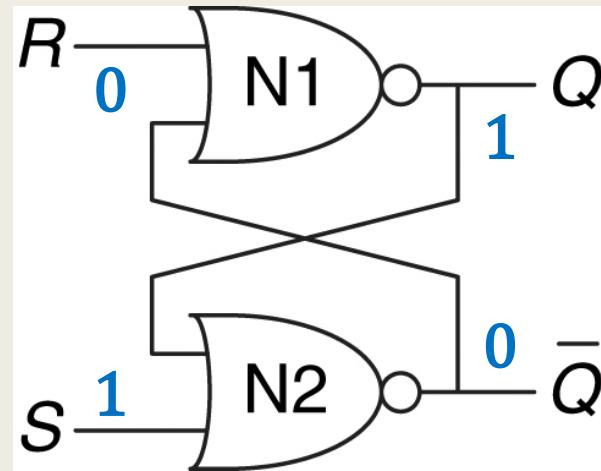
Set-Reset Latch (Μανδαλωτής SR)

- Το **Set-Reset latch** είναι το πιο απλό δισταθές ακολουθιακό κύκλωμα
- Αποτελείται από **δύο διασυζευγμένες πύλες NOR**
- Η διαφορά του με τους διασυζευγμένους αντιστροφείς είναι ότι η κατάσταση του μπορεί πλέον να ελεγχθεί μέσω των εισόδων **S (set)** και **R (reset)**
- Έχει **δύο εξόδους Q και \bar{Q}** που σε κανονική λειτουργία είναι **συμπληρωματικές**
- Το κύκλωμα διατηρεί αποθηκευμένη μία **κατάσταση Q** μεγέθους ενός bit, που ταυτίζεται με την έξοδο **Q**
 - την σταθερή κατάσταση 0 ή
 - την σταθερή κατάσταση 1



Ενεργοποίηση της εισόδου Set

- Η **είσοδος S (set)**, όταν ενεργοποιηθεί ($S = 1$) θέτει την έξοδο της πύλης NOR N2 στην τιμή 0 ($\bar{Q} = 0$)
- Σε κανονική λειτουργία η έξοδος \bar{Q} πρέπει να είναι συμπληρωματική της εξόδου Q ,
 - άρα η έξοδος της πύλης NOR N1 πρέπει να τεθεί στην τιμή 1 ($Q = 1$)
 - άρα η είσοδος **R (reset)** πρέπει να τεθεί στην τιμή 0 ($R = 0$)
- Σε κανονική λειτουργία το κύκλωμα τίθεται στην **σταθερή κατάσταση 1** εάν
 - $S = 1$ και $R = 0$
- Γενικά, η **είσοδος S (set)**, όταν ενεργοποιηθεί, δίνει στο κύκλωμα την εντολή:
 - **Πήγαινε στην κατάσταση 1**

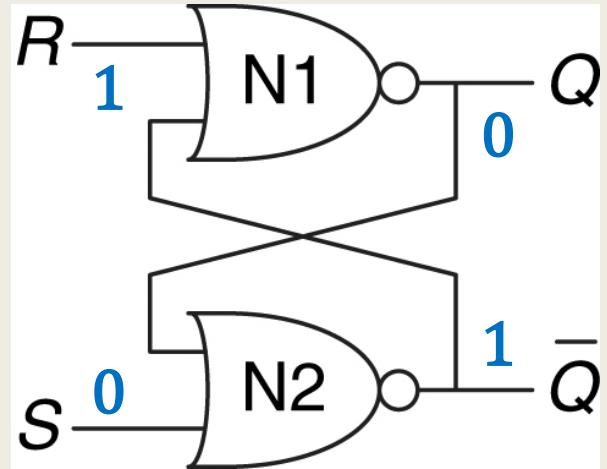


Πηγαίνει στην κατάσταση 1
NOR

A	B	Y	\bar{Y}
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Ενεργοποίηση της εισόδου Reset

- Η **είσοδος R (reset)**, όταν ενεργοποιηθεί ($R = 1$) θέτει την έξοδο της πύλης NOR N1 στην τιμή 0 ($Q = 0$)
- Σε κανονική λειτουργία η έξοδος \bar{Q} πρέπει να είναι συμπληρωματική της εξόδου Q ,
 - άρα η έξοδος της πύλης NOR N2 πρέπει να τεθεί στην τιμή 1 ($\bar{Q} = 1$)
 - άρα η είσοδος **S (set)** πρέπει να τεθεί στην τιμή 0 ($S = 0$)
- Σε κανονική λειτουργία το κύκλωμα τίθεται στην **σταθερή κατάσταση 0** εάν
 - $S = 0$ και $R = 1$
- Γενικά, η **είσοδος R (reset)**, όταν ενεργοποιηθεί, δίνει στο κύκλωμα την εντολή:
 - **Επανάφερε την κατάσταση 0**

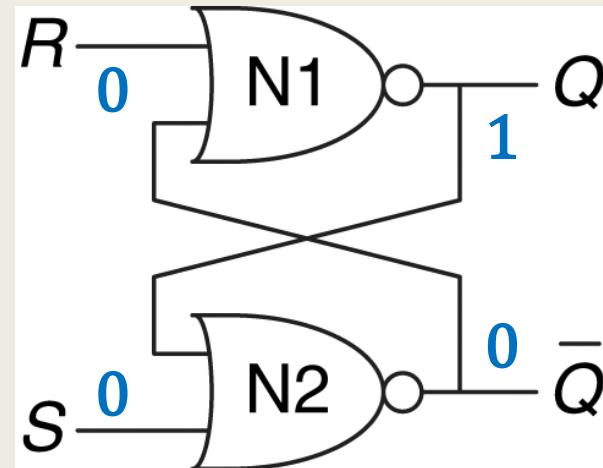


Πηγαίνει στην κατάσταση 0

NOR			
A	B	Y	\bar{Y}
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Απενεργοποίηση και των δύο εισόδων Set και Reset

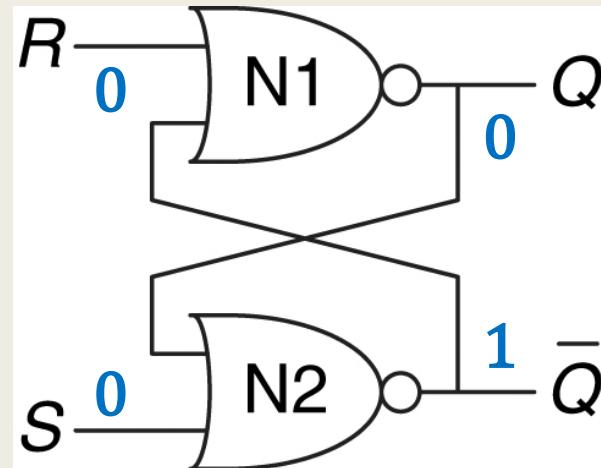
- Η **είσοδος S (set)**, όταν απενεργοποιηθεί ($S = 0$) εξαρτά την έξοδο της πύλης NOR N2 από την προηγούμενη τιμή της εξόδου Q ($\bar{Q} = \text{NOT } Q_{\text{prev}}$)
- Η **είσοδος R (reset)**, όταν απενεργοποιηθεί ($R = 0$) εξαρτά την έξοδο της πύλης NOR N1 από την προηγούμενη τιμή της εξόδου \bar{Q} ($Q = \text{NOT } \bar{Q}_{\text{prev}}$)
- Εάν το κύκλωμα βρισκόταν στην **σταθερή κατάσταση 1** (ήταν $S = 1$ και $R = 0$) τότε $Q_{\text{prev}} = 1$ και $\bar{Q}_{\text{prev}} = 0$
 - άρα $\bar{Q} = \text{NOT } 1 = 0$ και $Q = \text{NOT } 0 = 1$
 - δηλαδή διατηρεί αποθηκευμένη την προηγούμενη κατάσταση 1



Διατηρεί αποθηκευμένη την προηγούμενη κατάσταση 1

Απενεργοποίηση και των δύο εισόδων Set και Reset

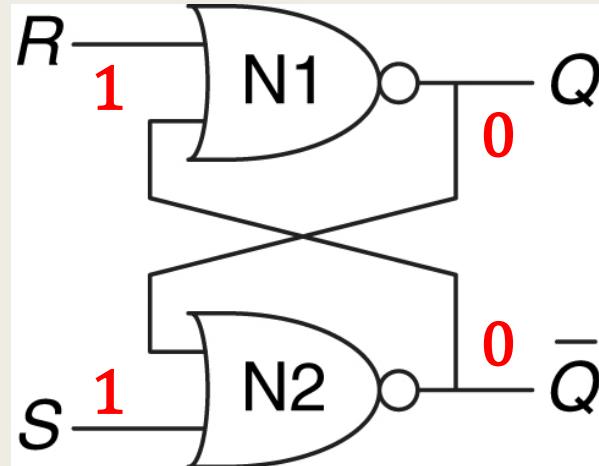
- Η **είσοδος S (set)**, όταν απενεργοποιηθεί ($S = 0$) εξαρτά την έξοδο της πύλης NOR N2 από την προηγούμενη τιμή της εξόδου Q ($\bar{Q} = \text{NOT } Q_{\text{prev}}$)
- Η **είσοδος R (reset)**, όταν απενεργοποιηθεί ($R = 0$) εξαρτά την έξοδο της πύλης NOR N1 από την προηγούμενη τιμή της εξόδου \bar{Q} ($Q = \text{NOT } \bar{Q}_{\text{prev}}$)
- Εάν το κύκλωμα βρισκόταν στην **σταθερή κατάσταση 0** (ήταν $S = 0$ και $R = 1$) τότε $Q_{\text{prev}} = 0$ και $\bar{Q}_{\text{prev}} = 1$
 - άρα $\bar{Q} = \text{NOT } 0 = 1$ και $Q = \text{NOT } 1 = 0$
 - δηλαδή διατηρεί αποθηκευμένη την προηγούμενη κατάσταση 0



Διατηρεί αποθηκευμένη την προηγούμενη κατάσταση 0

Ενεργοποίηση και των δύο εισόδων Set και Reset

- Η **είσοδος S (set)**, όταν ενεργοποιηθεί ($S = 1$) θέτει την έξοδο της πύλης NOR N2 στην τιμή 0 ($\bar{Q} = 0$)
- Η **είσοδος R (reset)**, όταν ενεργοποιηθεί ($R = 1$) θέτει την έξοδο της πύλης NOR N1 στην τιμή 0 ($Q = 0$)
- Η έξοδος \bar{Q} παύει να είναι συμπληρωματική της εξόδου Q και το κύκλωμα πηγαίνει σε μία μη έγκυρη κατάσταση ($Q = \bar{Q}$)

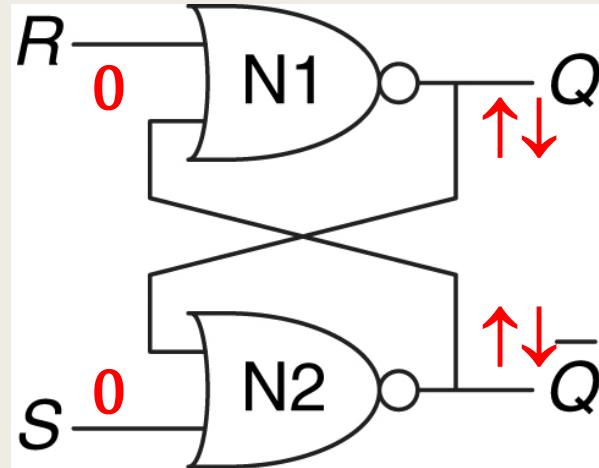


Πηγαίνει σε μη έγκυρη κατάσταση

NOR			
A	B	Y	\bar{Y}
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Απενεργοποίηση και των δύο εισόδων Set και Reset

- Η **είσοδος S (set)**, όταν απενεργοποιηθεί ($S = 0$) εξαρτά την έξοδο της πύλης NOR N2 από την προηγούμενη τιμή της εξόδου Q ($\bar{Q} = \text{NOT } Q_{\text{prev}}$)
- Η **είσοδος R (reset)**, όταν απενεργοποιηθεί ($R = 0$) εξαρτά την έξοδο της πύλης NOR N1 από την προηγούμενη τιμή της εξόδου \bar{Q} ($Q = \text{NOT } \bar{Q}_{\text{prev}}$)
- Εάν το κύκλωμα βρισκόταν στην **μη έγκυρη κατάσταση** (ήταν $S = 1$ και $R = 1$) τότε $Q_{\text{prev}} = 0$ και $\bar{Q}_{\text{prev}} = 0$
 - άρα $\bar{Q} = \text{NOT } 0 = 1$ και $Q = \text{NOT } 0 = 1$
 - το κύκλωμα θα εμφανίσει ταλαντώσεις ή θα βρεθεί στη μετασταθερή κατάσταση μέχρι ανεξέλεγκτα να ισορροπήσει σε μία σταθερή κατάσταση (0 ή 1)



Πηγαίνει σε ταλαντώσεις ή στη μετασταθερή κατάσταση

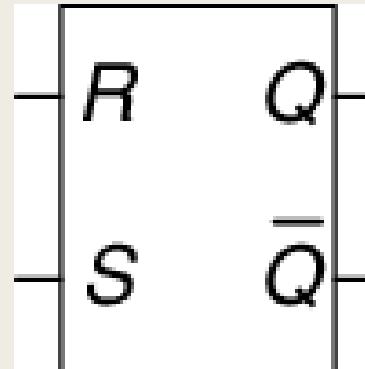
NOR			
A	B	Y	\bar{Y}
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Set-Reset (S-R) Latch

■ Πίνακας Αλήθειας

S	R	Q	\bar{Q}
0	0	Q_{prev}	\bar{Q}_{prev}
0	1	0	1
1	0	1	0
1	1	0	0

■ Σύμβολο



■ Σύνοψη

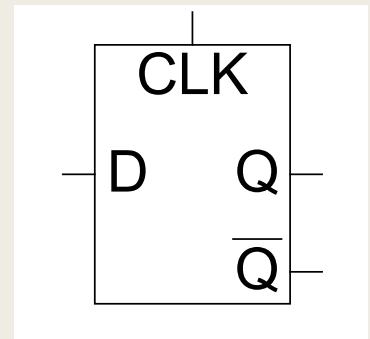
- To S-R Latch ελέγχει την κατάστασή του μέσω των εισόδων **S (set)** και **R (reset)**
- **S = 1** και **R = 0** ⇒ **Πηγαίνει στην κατάσταση $Q = 1$**
- **S = 0** και **R = 1** ⇒ **Πηγαίνει στην κατάσταση $Q = 0$**
- Το κύκλωμα διατηρεί αποθηκευμένη μία **κατάσταση Q** μεγέθους ενός bit, που ταυτίζεται με την έξοδο **Q**
- **S = 0** και **R = 0** ⇒ **Αποθηκεύει την προηγούμενη κατάσταση $Q = Q_{\text{prev}}$**
- **S = 1** και **R = 1** ⇒ **Μη έγκυρη κατάσταση ($Q = \bar{Q}$)**

Κάτι πρέπει να κάνουμε ώστε να την αποφύγουμε!

D (Data) Latch (Μανδαλωτής D)

- Το **D (Data) Latch** επιλύει το πρόβλημα της μη έγκυρης κατάστασης ($Q = \bar{Q}$)
- Το κύκλωμα έχει δύο εισόδους:
 - την είσοδο του ρολογιού **CLK** που ελέγχει **πότε** πρέπει να αλλάξει η κατάσταση
 - την είσοδος δεδομένων **D** που ελέγχει **τί τιμή** πρέπει να έχει η κατάσταση **Q**
- Έχει **δύο εξόδους Q και \bar{Q}** που είναι πάντα **συμπληρωματικές**
- Το κύκλωμα διατηρεί αποθηκευμένη μία **κατάσταση Q** μεγέθους ενός bit, που ταυτίζεται με την έξοδο **Q**
 - την σταθερή κατάσταση 0 ή
 - την σταθερή κατάσταση 1

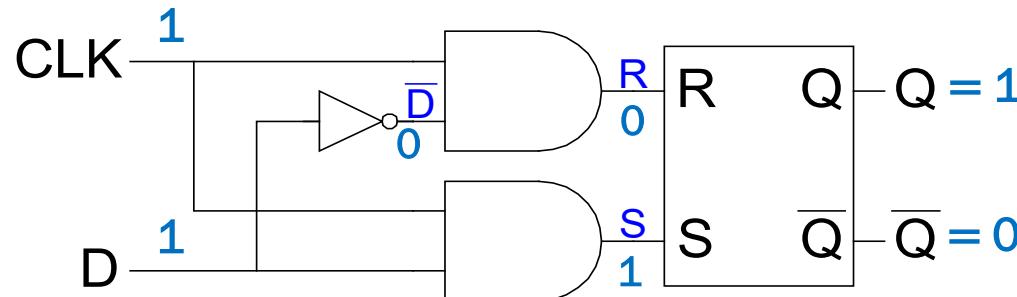
■ Σύμβολο



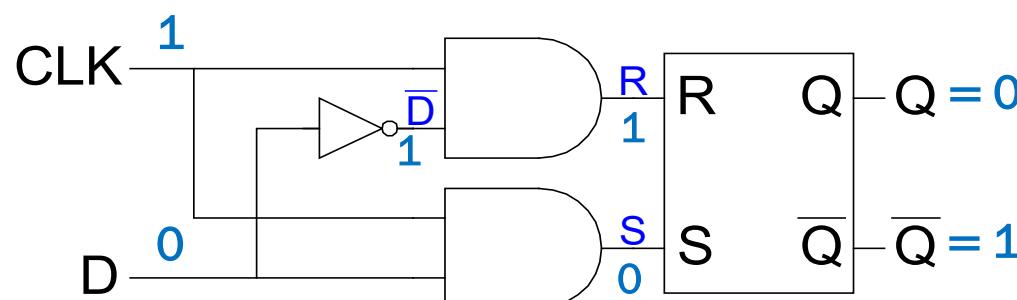
Λειτουργία D Latch ($CLK = 1$)

- Όταν $CLK = 1$

- $D = 1 \Rightarrow S = 1$ και $R = 0 \Rightarrow$ Πηγαίνει στην κατάσταση $Q = 1$

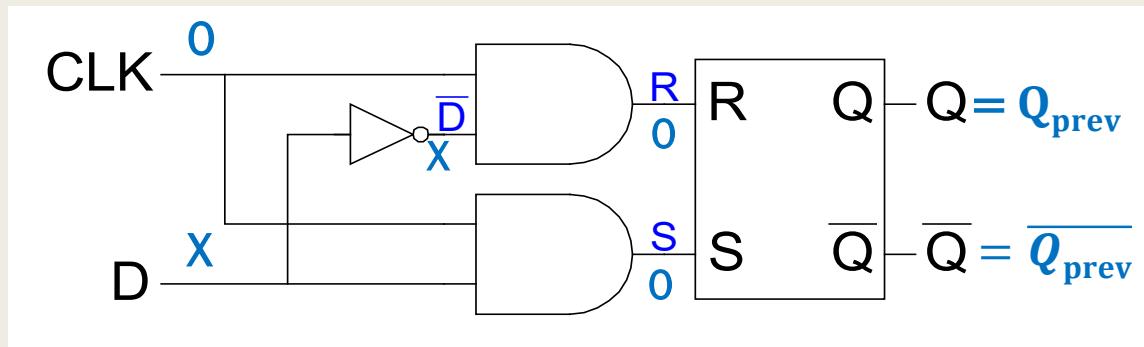


- $D = 0 \Rightarrow S = 0$ και $R = 1 \Rightarrow$ Πηγαίνει στην κατάσταση $Q = 0$



Λειτουργία D Latch (CLK = 0)

- Όταν $\text{CLK} = 0$
 - $D = X \Rightarrow S = 0$ και $R = 0 \Rightarrow$
Αποθηκεύει την προηγούμενη κατάσταση $Q = Q_{\text{prev}}$

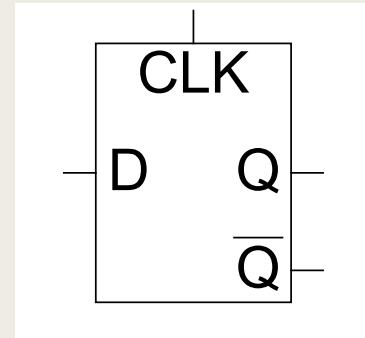


D Latch

■ Πίνακας Αλήθειας

CLK	D	D'	S	R	Q	\bar{Q}
0	X	X'	0	0	Q_{prev}	\bar{Q}_{prev}
1	0	1	0	1	0	1
1	1	0	1	0	1	0

■ Σύμβολο



■ Σύνοψη

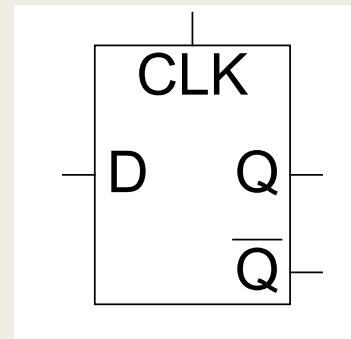
- Το D Latch εξαρτάται από το επίπεδο (την τιμή 0 ή 1) του σήματος CLK
- To D Latch ενημερώνει την κατάστασή του συνεχώς για όσο διάστημα ισχύει $CLK = 1$ (είναι διάφανο και συμπεριφέρεται ως απομονωτής)
- **CLK = 1 και D = 1** \Rightarrow Πηγαίνει στην κατάσταση $Q = 1$
- **CLK = 1 και D = 0** \Rightarrow Πηγαίνει στην κατάσταση $Q = 0$
- To D Latch διατηρεί αποθηκευμένη μία **κατάσταση Q** μεγέθους ενός bit, που ταυτίζεται με την έξοδο **Q** για όσο διάστημα ισχύει $CLK = 0$
- **CLK = 0 και D = X** \Rightarrow Αποθηκεύει την προηγούμενη κατάσταση $Q = Q_{prev}$
- **Δεν πηγαίνει σε μη έγκυρη κατάσταση ($Q = \bar{Q}$)**

D Latch

- Πίνακας Αλήθειας

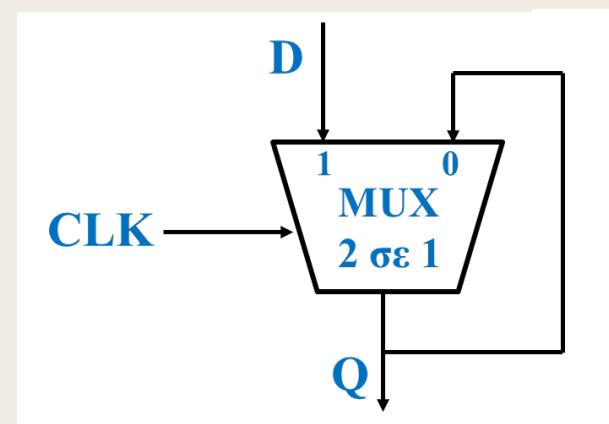
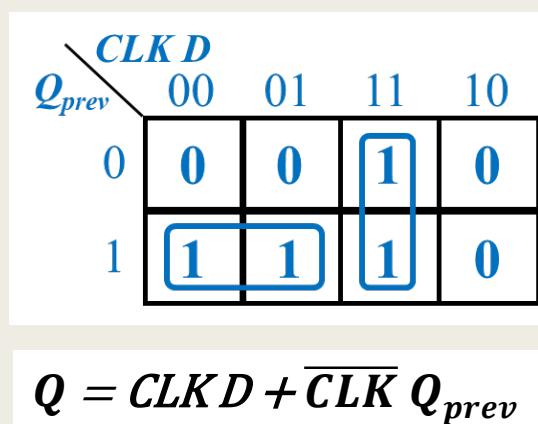
CLK	D	D'	S	R	Q	\bar{Q}
0	X	X'	0	0	Q_{prev}	\bar{Q}_{prev}
1	0	1	0	1	0	1
1	1	0	1	0	1	0

- Σύμβολο



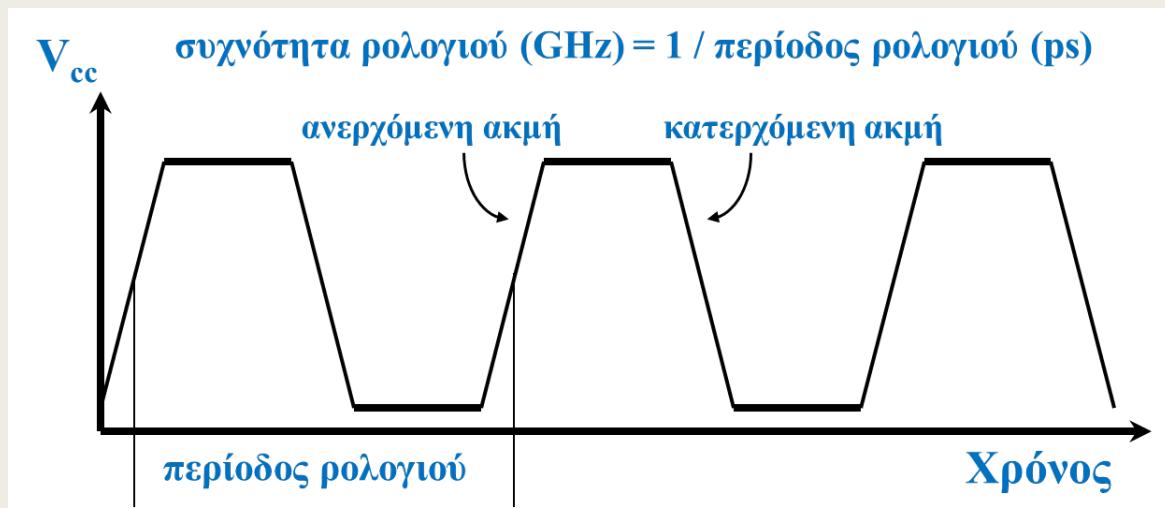
- Υλοποίηση με πολυπλέκτη 2 σε 1

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Το σήμα CLK: Latches vs Flip-Flops

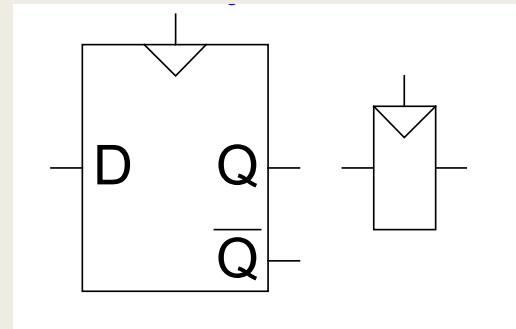
- Το **σήμα του ρολογιού CLK** είναι ένας τετραγωνικός παλμός με εναλλασσόμενα επίπεδα 0 και 1 και με ανερχόμενες (από το 0 στο 1) και κατερχόμενες (από το 1 στο 0) ακμές
 - Τα δισταθή στοιχεία που μπορούν να ενημερώνουν συνεχώς την κατάστασή τους σε ένα από τα επίπεδα (0 ή 1) του ρολογιού αποκαλούνται **Latches**
 - Τα δισταθή στοιχεία που ενημερώνουν την κατάστασή τους μόνο σε μια ακμή του ρολογιού, όπως για παράδειγμα στην ανερχόμενη ακμή του ρολογιού, αποκαλούνται **Flip-Flops**



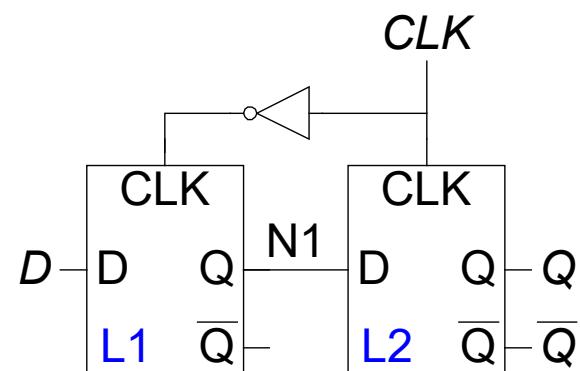
D (Data) Flip-Flop

- Ένα D Flip-Flop αποτελείται από δύο D Latches συνδεδεμένα στην σειρά που ελέγχονται από συμπληρωματικά CLK
 - Το πρώτο D Latch (L1) ονομάζεται αφέντης (master)
 - Το δεύτερο D Latch (L2) ονομάζεται σκλάβος (slave)
 - Ο N1 είναι ενδιάμεσος κόμβος
- Τα D Flip-Flops είναι επίσης γνωστά ως:
 - Flip-Flop αφέντη-σκλάβου
 - Flip-Flop ενεργοποιούμενα στις ακμές (του CLK).

■ Σύμβολο



■ Σχηματικό διάγραμμα



Λειτουργία D Flip-Flop

■ Όταν $CLK = 0$

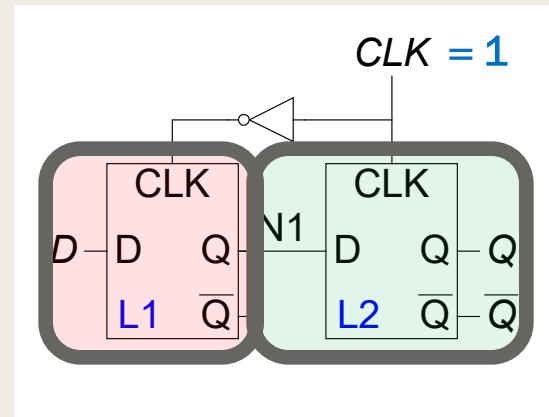
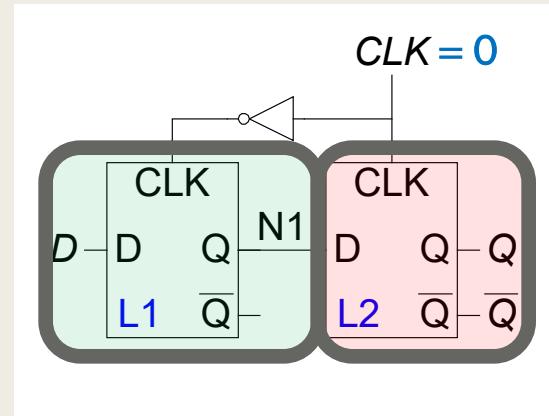
- το D-Latch αφέντης ($L1$) είναι **διαφανές**
- το D-Latch σκλάβος ($L2$) είναι **αδιαφανές**
- áρα, η τιμή στην είσοδο δεδομένων D μεταφέρεται μέσω του $L1$ στον κόμβο $N1$, χωρίς να επηρεάζεται η έξοδος Q του $L2$

■ Όταν $CLK = 1$

- το D-Latch αφέντης ($L1$) είναι **αδιαφανές**
- το D-Latch σκλάβος ($L2$) είναι **διαφανές**
- áρα, η τιμή στον κόμβο $N1$ μεταφέρεται στην έξοδος Q του $L2$, ενώ η τιμή στον κόμβο $N1$ παραμένει σταθερή γιατί παύει να ελέγχεται από την είσοδο δεδομένων D

■ Η τιμή που υπήρχε στην είσοδο δεδομένων D αμέσως πριν η τιμή του CLK μεταβεί από το 0 στο 1 αντιγράφεται στην έξοδο Q αμέσως μετά την ανερχόμενη ακμή του CLK (**το D Flip-Flop ενημερώνει την κατάστασή του**)

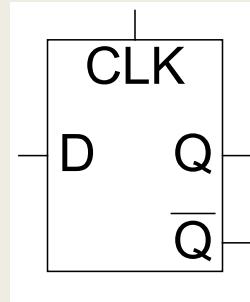
■ Σε όλες τις άλλες περιπτώσεις, **το D Flip-Flop διατηρεί την αποθηκευμένη κατάστασή του**, επειδή υπάρχει πάντα ένας αδιαφανές D Latch που μπλοκάρει τη διαδρομή μεταξύ της εισόδου D και της εξόδου Q



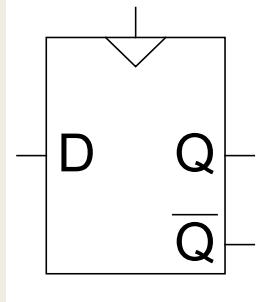
D Latch vs D Flip-Flop*

- D Latch: συνεχής ενημέρωση της εξόδου Q αντιγράφοντας την τιμή της εισόδου D, όσο **CLK = 1**
- D Flip-Flop: ενημέρωση της εξόδου Q αντιγράφοντας την τιμή της εισόδου D **μόνο στην ανερχόμενη ακμή του CLK**
- Σε όλες τις άλλες περιπτώσεις οι έξοδοι Q διατηρούν τη τιμή τους και τα κυκλώματα την κατάστασή τους
 - Η κατάστασή ταυτίζεται με την τιμή της εξόδου Q
- Η αρχική τιμή της εξόδου Q είναι απροσδιόριστη
 - Στην πράξη τα κυκλώματα πηγαίνουν ανεξέλεγκτα σε μία από τις δύο σταθερές καταστάσεις (0 ή 1)

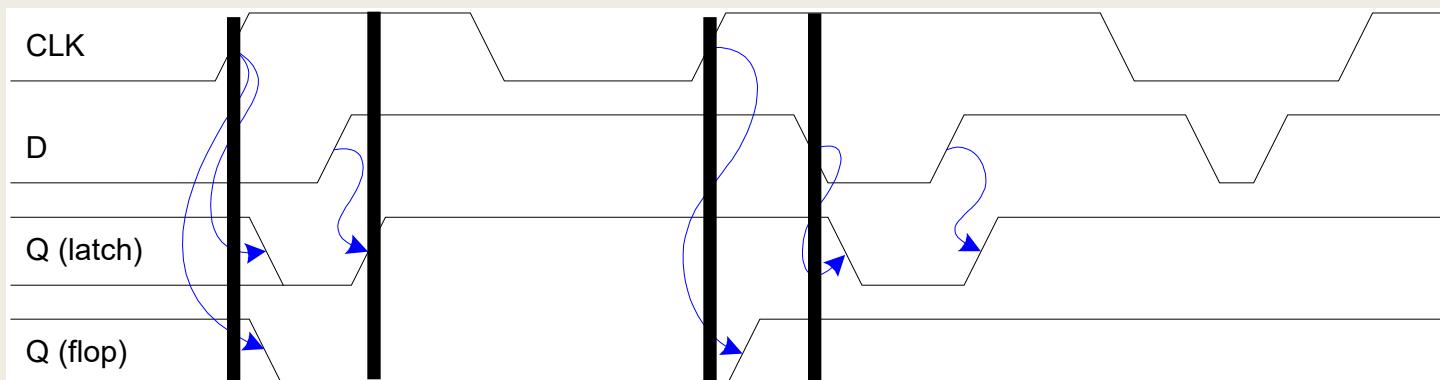
D Latch



D Flip-Flop

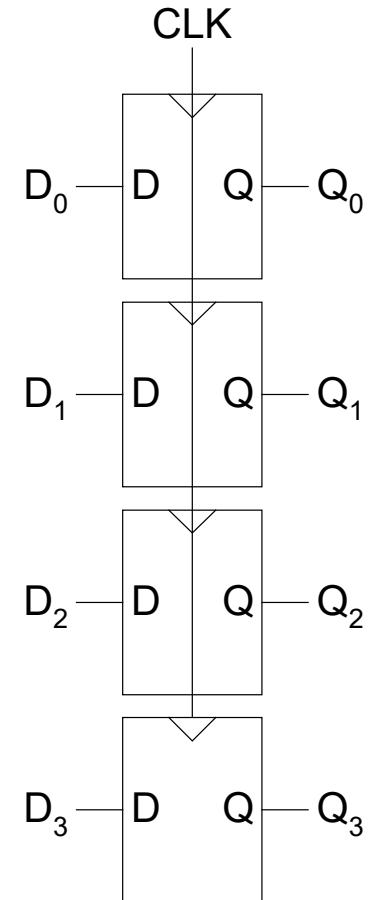
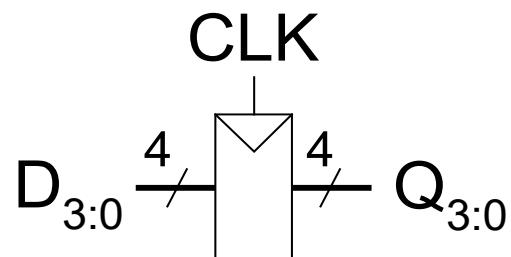


*Παράδειγμα 3.2



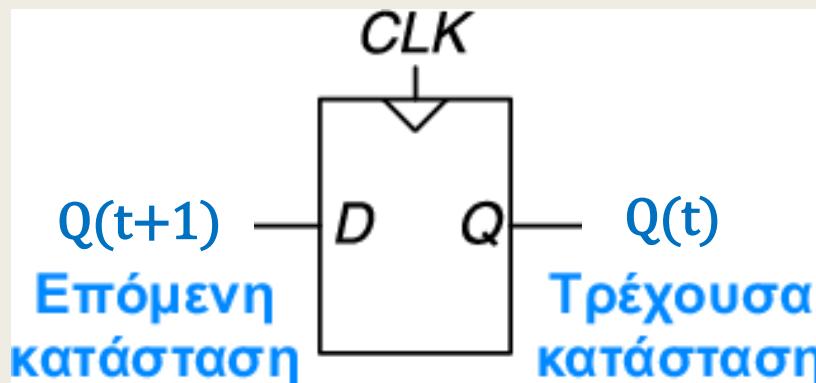
Καταχωρητές

- Ένας **καταχωρητής** (register) μεγέθους N bit είναι μια συστοιχία από N Flip-Flop τα οποία χρησιμοποιούν από κοινού μια είσοδο ρολογιού CLK, έτσι ώστε όλα τα bit του καταχωρητή να ενημερώνονται ταυτόχρονα
- Οι καταχωρητές αποτελούν το θεμελιώδες δομικό στοιχείο στα σύγχρονα ακολουθιακά κυκλώματα (ή μηχανές πεπερασμένων καταστάσεων – FSM)



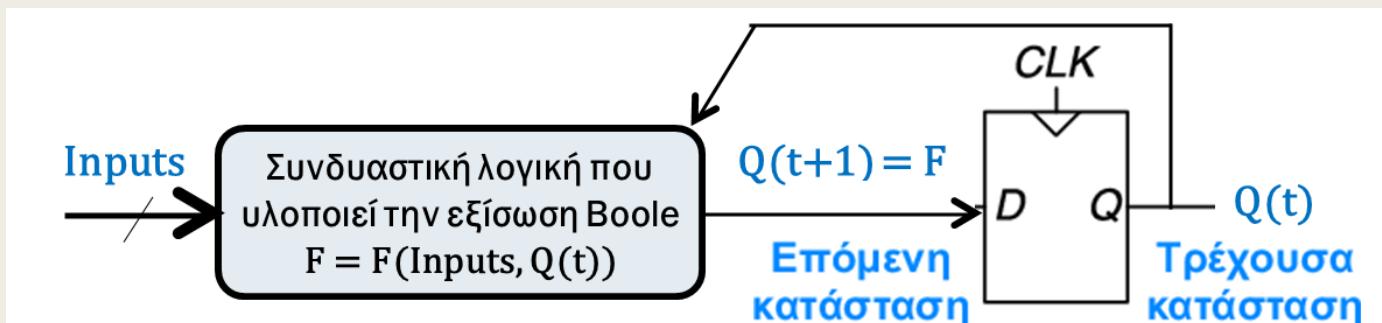
Σχεδίαση άλλων Flip-Flop

- Χρησιμοποιείται σαν βάση το **D Flip-Flop**, το οποίο
 - διατηρεί αποθηκευμένη την **τρέχουσα κατάστασή $Q(t)$** (που ταυτίζεται με την τιμή της εξόδου Q), όσο το CLK δεν βρίσκεται στην ανερχόμενη ακμή του
 - πηγαίνει στην **επόμενη κατάσταση $Q(t+1)$** , που είναι η τιμή που έχει η είσοδος D κατά την επόμενη ανερχόμενη ακμή του CLK
- $Q(t+1) = D$



Σχεδίαση άλλων Flip-Flop

- Προσδιορίζουμε τον **πίνακας αλήθειας** του νέου Flip-Flop
 - εξετάζοντας ποιες είναι οι τιμές των εισόδων του κατά την ανερχόμενη ακμή του CLK
 - στις εισόδους του πίνακα συμπεριλαμβάνεται η τρέχουσα κατάσταση $Q(t)$
- Προσδιορίζουμε την **εξίσωση Boole** για την επόμενη κατάσταση $Q(t+1)$
 - ως συνάρτηση των εισόδων και της τρέχουσας κατάστασης $Q(t)$
 - μετά από ελαχιστοποίηση με τον χάρτη Karnaugh
- Σχεδιάζουμε την **συνδυαστική λογική** με βάση την εξίσωση Boole
 - Η έξοδος F αυτού του συνδυαστικού κυκλώματος συνδέεται στην είσοδο δεδομένων D του D Flip-Flop, άρα $D = F$
- **$Q(t+1) = F(\text{Inputs}, Q(t))$**



D Flip-Flop με επαναφορά στο 0

- Ένα D Flip-Flop με δυνατότητα σύγχρονης επαναφοράς στο 0 (resettable Flip-Flop) διαθέτει άλλη μία είσοδο που ονομάζεται **RESET**
 - Η είσοδος **RESET** είναι **σύγχρονη** (δηλαδή επιδρά στο κύκλωμα στην ανερχόμενη ακμή του CLK)
 - Όταν η είσοδος **RESET** έχει την τιμή **FALSE**, συμπεριφέρεται ως απλό D Flip-Flop
 - Όταν η είσοδος **RESET** έχει την τιμή **TRUE**, **αγνοεί** την είσοδο D και επαναφέρει την κατάσταση και έξοδο Q στην **τιμή 0**
- Στα D Flip-Flop με δυνατότητα ασύγχρονης επαναφοράς στο 0, η επαναφορά στο 0 λαμβάνει χώρα αμέσως μόλις η είσοδος **RESET** πάρει την τιμή TRUE, ανεξάρτητα από την τιμή του CLK
 - Το ασύγχρονο σήμα επαναφοράς στο 0 ονομάζεται και **CLEAR**
- Αυτού του είδους τα D Flip-Flop είναι χρήσιμα στην περίπτωση που θέλουμε να επιβάλλουμε μια γνωστή κατάσταση στα Flip-Flop ενός συστήματος όταν το ενεργοποιούμε για πρώτη φορά

D Flip-Flop με επαναφορά στο 0

- Σχεδίαση ενός D Flip-Flop με δυνατότητα σύγχρονης επαναφοράς στο 0
 - Η είσοδος RESET (R) είναι ενεργή στο HIGH (active HIGH)

- Πίνακας Αλήθειας

RESET	D	$Q(t)$	$Q(t + 1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

LOAD

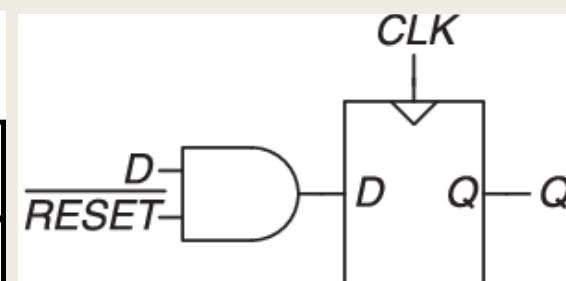
RESET

R	D	00	01	11	10
$Q(t)$	0	0	1	0	0
	1	0	1	0	0

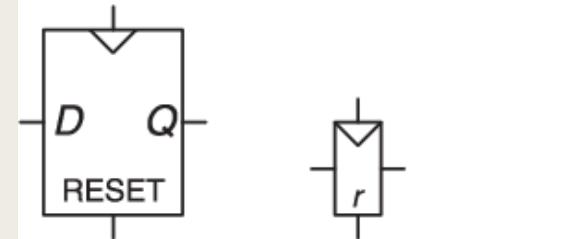
$$Q(t + 1) = \overline{RESET} D$$

Σε αυτή την περίπτωση
το $Q(t+1)$ δεν εξαρτάται
από το $Q(t)$

- K-μαρ και εξίσωση



- Σύμβολα



D Flip-Flop με τοποθέτηση στο 1

- Ένα D Flip-Flop με δυνατότητα σύγχρονης τοποθέτησης στο 1 (settable Flip-Flop) διαθέτει άλλη μία είσοδο που ονομάζεται **SET**
 - Η είσοδος **SET** είναι **σύγχρονη** (δηλαδή επιδρά στο κύκλωμα στην ανερχόμενη ακμή του CLK)
 - Όταν η είσοδος **SET** έχει την τιμή **FALSE**, συμπεριφέρεται ως απλό D Flip-Flop
 - Όταν η είσοδος **SET** έχει την τιμή **TRUE**, **αγνοεί** την είσοδο **D** και τοποθετεί την κατάσταση και έξοδο **Q** στην **τιμή 1**
- Στα D Flip-Flop με δυνατότητα ασύγχρονης τοποθέτησης στο 1, η τοποθέτηση στο 1 λαμβάνει χώρα αμέσως μόλις η είσοδος **SET** πάρει την τιμή TRUE, ανεξάρτητα από την τιμή του CLK
 - Το ασύγχρονο σήμα τοποθέτησης στο 1 ονομάζεται και **PRESET**
- Αυτού του είδους τα D Flip-Flop είναι χρήσιμα στην περίπτωση που θέλουμε να επιβάλλουμε μια γνωστή κατάσταση στα Flip-Flop ενός συστήματος όταν το ενεργοποιούμε για πρώτη φορά

D Flip-Flop με τοποθέτηση στο 1

- Σχεδίαση ενός D Flip-Flop με δυνατότητα σύγχρονης τοποθέτησης στο 1

– Η είσοδος SET (S) είναι ενεργή στο HIGH (active HIGH)

■ Πίνακας Αλήθειας

SET	D	$Q(t)$	$Q(t + 1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
<hr/>			
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

■ K-μαρ και εξίσωση

	$S \ D$	00	01	11	10
$Q(t)$	0	0	1	1	1
0	0	1	1	1	1
1	0	1	1	1	1

$Q(t + 1) = S + D$

Σε αυτή την περίπτωση το $Q(t+1)$ δεν εξαρτάται από το $Q(t)$

■ Σχηματικό διάγραμμα

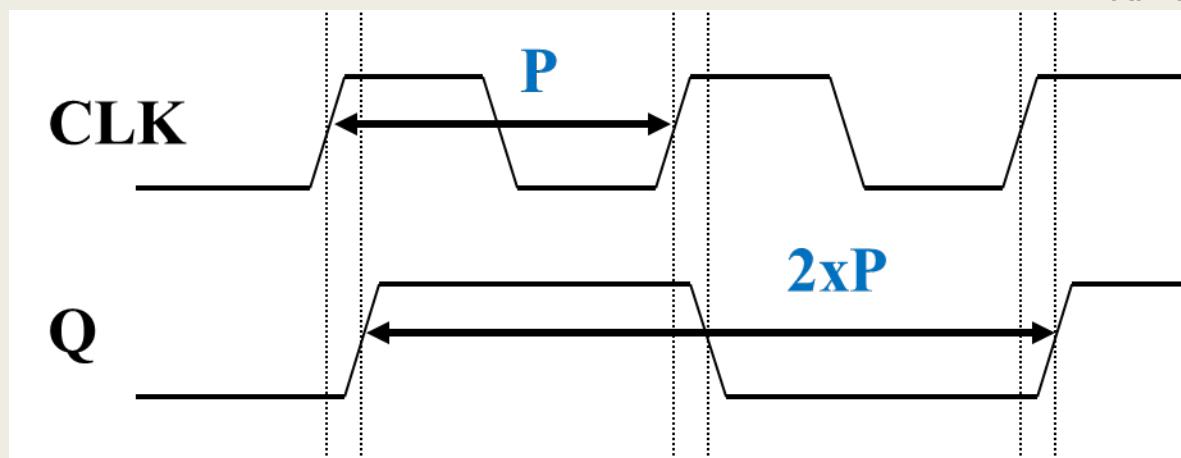
■ Σύμβολα

33

T (Toggle) Flip-Flop

- Το **T**(toggle, εναλλαγή) **Flip-Flop** διαθέτει μία είσοδο *CLK* και μία έξοδο *Q*
- Σε κάθε ανερχόμενη ακμή του *CLK*, η έξοδος *Q* εναλλάσσει την τιμή της με το συμπλήρωμα της προηγούμενης τιμής της
- Το T Flip-Flop χρησιμοποιείται κυρίως ως **διαιρέτης συχνότητας διά 2**
 - *N T Flip-Flops* στη σειρά απαρτίζουν έναν διαιρέτη συχνότητας διά 2^N

Σε τι θα το χρησιμοποιούσαμε;



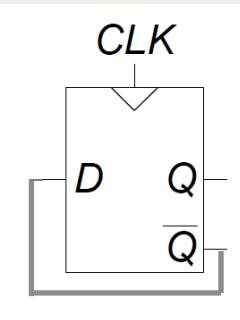
T Flip-Flop

- Σχεδίαση ενός T Flip-Flop
- Πίνακας Αλήθειας
- Εξίσωση Boole
- Σχηματικό διάγραμμα

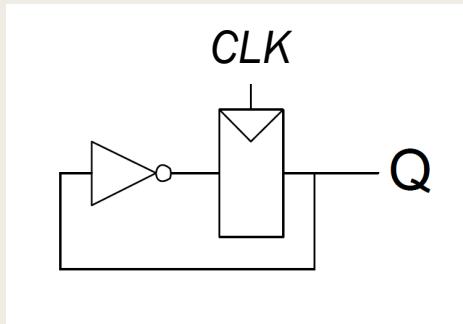
$Q(t)$	$Q(t + 1)$
0	1
1	0

TOGGLE

$$Q(t + 1) = \overline{Q(t)}$$



- Σύμβολο



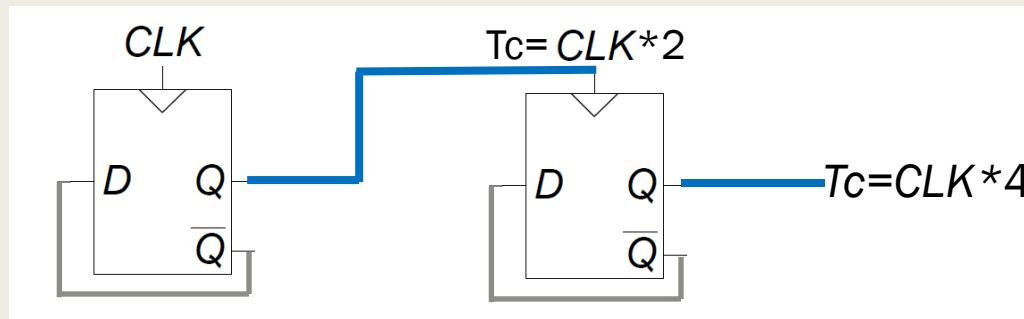
Διαιρέτης συχνότητας δια 4

- Σχηματικό διάγραμμα με τη χρήση δύο T Flip-Flops

$$F=1/CLK$$

$$F_1=F/2$$

$$F_2=F/4$$



D Flip-Flop με έγκριση (enabled)

- Ένα **D Flip-Flop με έγκριση (enabled flip-flop)** διαθέτει μία ακόμα είσοδο που ονομάζεται **EN** ή **ENABLE**
- Η επιπλέον είσοδος **EN**, χρησιμοποιείται για να καθορίζει αν τα δεδομένα θα φορτωθούν ή όχι κατά την επόμενη ακμή του CLK
 - Όταν η **είσοδος EN** έχει την τιμή **TRUE**, το D Flip-Flop με έγκριση συμπεριφέρεται ως απλό D Flip-Flop
 - Όταν η **είσοδος EN** έχει την τιμή **FALSE**, το D Flip-Flop με έγκριση **αγνοεί το CLK** και διατηρεί την κατάστασή του
- Αυτού του είδους τα κυκλώματα είναι χρήσιμα όταν θέλουμε να φορτώνουμε μια νέα τιμή σε ένα Flip-Flop μόνο σε κάποιες περιπτώσεις, και όχι σε κάθε ακμή του ρολογιού
- Το σήμα έγκρισης ονομάζεται και **write enable** ή **clock enable**

D Flip-Flop με έγκριση (enabled)

- **Σχεδίαση ενός D Flip-Flop με έγκριση**

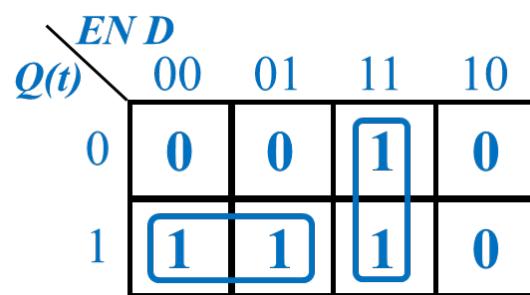
- Η είσοδος *ENABLE (EN)* είναι ενεργή στο *HIGH* (*active HIGH*)

- Πίνακας Αλήθειας

EN	D	$Q(t)$	$Q(t + 1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

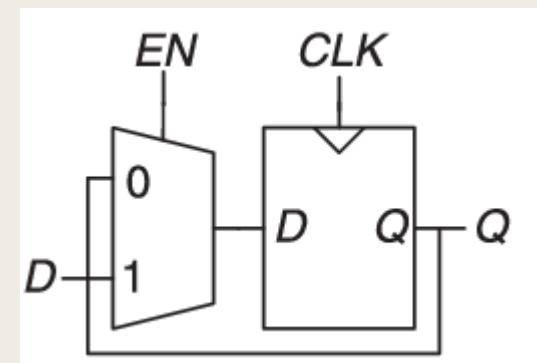
HOLD

- K-map και εξίσωση

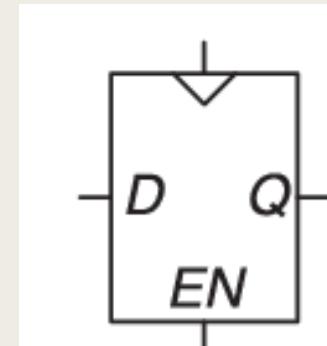


$$Q(t + 1) = \overline{EN} Q(t) + EN D$$

- Σχηματικό διάγραμμα

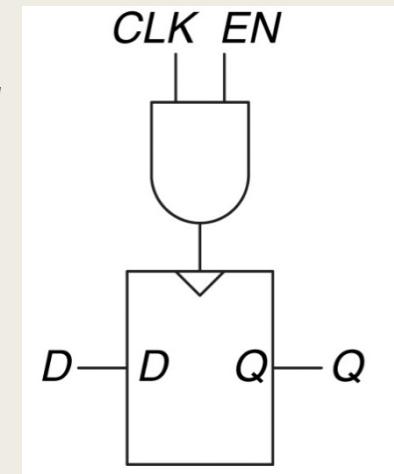


- Σύμβολο



D Flip-Flop με έγκριση (enabled)

- **Μία εναλλακτική σχεδίαση ενός D Flip-Flop με έγκριση**
 - Η είσοδος *ENABLE (EN)* είναι ενεργή στο *HIGH* (*active HIGH*)
- Το σήμα του ρολογιού *CLK* περνάει μέσα από μία πύλη AND πριν συνδεθεί στην είσοδο *CLK* του D Flip-Flop
 - Αν $EN = 1$, τότε η είσοδος *CLK* του D Flip-Flop εναλλάσσεται φυσιολογικά
 - Αν $EN = 0$, τότε η είσοδος *CLK* του D Flip-Flop παγώνει στην τιμή 0, και το D Flip-Flop διατηρεί την κατάστασή του
- Η συγκεκριμένη σχεδίαση εγκυμονεί κινδύνους
 - Η τιμή της εισόδου *EN* δεν πρέπει να μεταβάλλεται ενόσω ισχύει $CLK = 1$, ώστε να μην εμφανισθεί μεταβατικός παλμός (*glitch*) στην είσοδο *CLK* που θα οδηγήσει σε ανεξέλεγκτη αλλαγή κατάστασης
 - Η προσθήκη πύλης στο *CLK* το καθυστερεί και μπορεί να προκαλέσει και σφάλματα χρονισμού
 - Δεν συνιστάται η χρήση του *CLK* με πύλες



T Flip-Flop με έγκριση (enabled)

- Ένα T Flip-Flop με έγκριση (enabled flip-flop) διαθέτει μία ακόμα είσοδο που ονομάζεται **EN** ή **ENABLE**
- Η επιπλέον είσοδος **EN**, χρησιμοποιείται για να καθορίζει εάν θα εναλλαχθεί ή όχι η κατάστασή του T Flip-Flop κατά την επόμενη ακμή του CLK
 - Όταν η **είσοδος EN** έχει την τιμή **TRUE**, το T Flip-Flop με έγκριση εναλλάσσει την κατάστασή του (συμπεριφέρεται σαν T Flip-Flop)
 - Όταν η **είσοδος EN** έχει την τιμή **FALSE**, το T Flip-Flop με έγκριση διατηρεί την κατάστασή του
- Αυτού του είδους τα κυκλώματα είναι ιδιαίτερα χρήσιμα στην υλοποίηση **σύγχρονων μετρητών**

T Flip-Flop με έγκριση (enabled)

- Σχεδίαση ενός T Flip-Flop με έγκριση
 - Η είσοδος *ENABLE* (*EN*) είναι ενεργή στο *HIGH* (active *HIGH*)

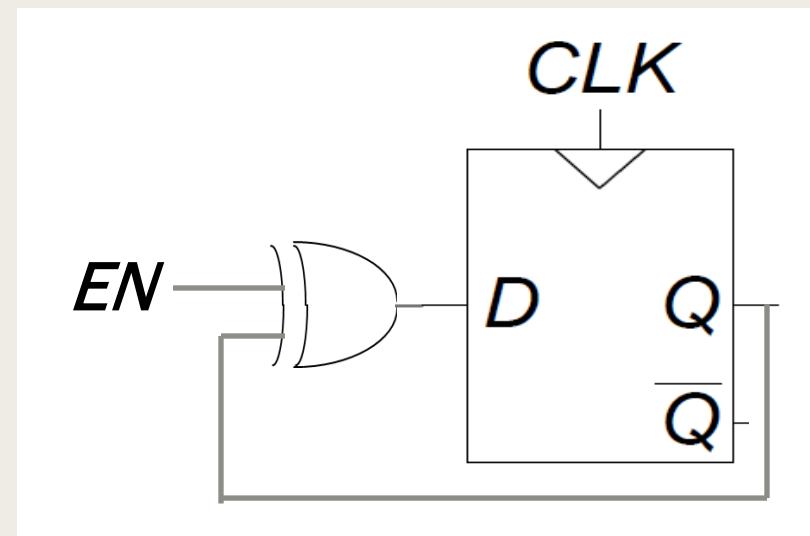
- Πίνακας Αλήθειας

EN	Q(t)	Q(t + 1)	
0	0	0	HOLD
0	1	1	
<hr/>			
1	0	1	TOGGLE
1	1	0	

- Εξίσωση Boole

$$Q(t + 1) = \overline{EN} Q(t) + EN \overline{Q(t)} = EN \oplus Q(t)$$

- Σχηματικό διάγραμμα



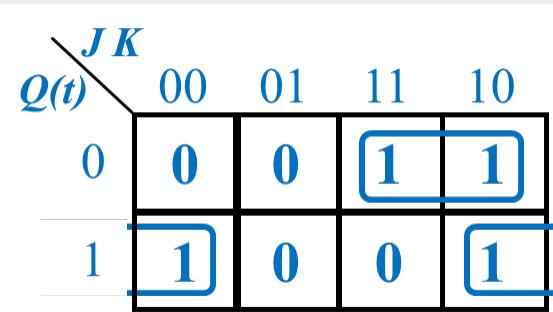
J-K Flip-Flop

- Το **J-K Flip-Flop** δέχεται ένα σήμα CLK και δύο σύγχρονες εισόδους (το *J* και το *K*).
- Κατά την ανερχόμενη ακμή του CLK ενημερώνει την έξοδο *Q*, σύμφωνα με τις τιμές που έχουν οι είσοδοι *J* και *K*, ως εξής:
 - Όταν οι είσοδοι *J* και *K* έχουν και οι δύο την τιμή 0, η έξοδος *Q* διατηρεί την προηγούμενη τιμή της (*hold*)
 - Όταν η είσοδος *J* έχει την τιμή 0 και η είσοδος *K* έχει την τιμή 1, η έξοδος *Q* παίρνει την τιμή 0 (*reset*)
 - Όταν η είσοδος *J* έχει την τιμή 1 και η είσοδος *K* έχει την τιμή 0, η έξοδος *Q* παίρνει την τιμή 1 (*set*)
 - Όταν οι είσοδοι *J* και *K* έχουν και οι δύο την τιμή 1, η έξοδος *Q* εναλλάσσει την τιμή της με το συμπλήρωμα της προηγούμενης τιμής της (*toggle*)
- Αυτό το κύκλωμα ήταν ιδιαίτερα δημοφιλές την εποχή που οι υλοποιήσεις βασίζονταν σε πλακέτες με SSI και MSI ολοκληρωμένα κυκλώματα

J-K Flip-Flop

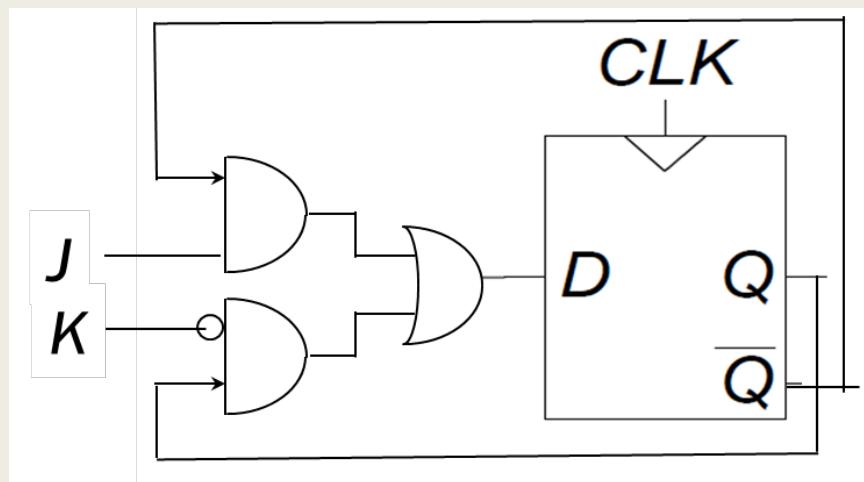
- Σχεδίαση ενός J-K Flip-Flop
- Πίνακας Αλήθειας
- K-map και εξίσωση

J	K	Q(t)	Q(t + 1)
0	0	0	0
0	0	1	1
.....			RESET HOLD
0	1	0	0
0	1	1	0
.....			TOGGLE SET
1	0	0	1
1	0	1	1
.....			
1	1	0	1
1	1	1	0



$$Q(t + 1) = \overline{Q(t)} J + Q(t) \bar{K}$$

- Σχηματικό διάγραμμα



Επιλεγμένη άσκηση: το A-B Flip-Flop

- Το **A-B Flip-Flop** δέχεται ένα σήμα CLK και δύο σύγχρονες εισόδους (το *A* και το *B*).
- Κατά την ανερχόμενη ακμή του CLK ενημερώνει την έξοδο *Q*, σύμφωνα με τις τιμές που έχουν οι είσοδοι *A* και *B*, ως εξής:
 - Όταν οι είσοδοι *A* και *B* έχουν και οι δύο την τιμή 0, η έξοδος *Q* διατηρεί την προηγούμενη τιμή της (*hold*)
 - Όταν η είσοδος *A* έχει την τιμή 0 και η είσοδος *B* έχει την τιμή 1, η έξοδος *Q* εναλλάσσει την τιμή της με το συμπλήρωμα της προηγούμενης τιμής της (*toggle*)
 - Όταν η είσοδος *A* έχει την τιμή 1 και η είσοδος *B* έχει την τιμή 0, η έξοδος *Q* παίρνει την τιμή 0 (*reset*)
 - Όταν οι είσοδοι *A* και *B* έχουν και οι δύο την τιμή 1, η έξοδος *Q* αντιγράφει την τιμή εισόδου *D* (*load*)
- Να σχεδιάσετε το νέο Flip-Flop χρησιμοποιώντας στη συνδυαστική λογική έναν πολυπλέκτη 2 σε 1, μία πύλη XOR και όποια άλλη πύλη απαιτείται

Επιλεγμένη άσκηση: το A-B Flip-Flop

- Από τον **Πίνακα Λειτουργιών** στον **Πίνακα Αλήθειας**

AB	Λειτουργία
00	HOLD
01	TOGGLE
10	RESET
11	LOAD

Επιλεγμένη άσκηση: το A-B Flip-Flop

- Από τον **Πίνακα Λειτουργιών** στον **Πίνακα Αλήθειας**

AB	Λειτουργία
00	HOLD
01	TOGGLE
10	RESET
11	LOAD

A	B	D	Q(t)	Q(t+1)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0

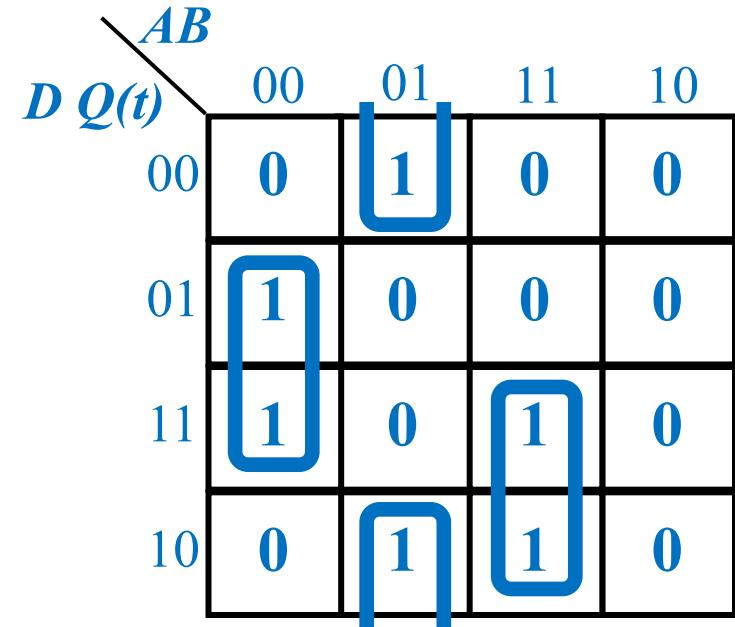
A	B	D	Q(t)	Q(t+1)
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Επιλεγμένη άσκηση: το A-B Flip-Flop

- Από τον **Πίνακα Αλήθειας** στον **Χάρτη Karnaugh** και την **εξίσωση Boole** για την επόμενη κατάσταση $Q(t+1)$

A	B	D	$Q(t)$	$Q(t+1)$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1

A	B	D	$Q(t)$	$Q(t+1)$
1	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	1
1	1	1	1	1

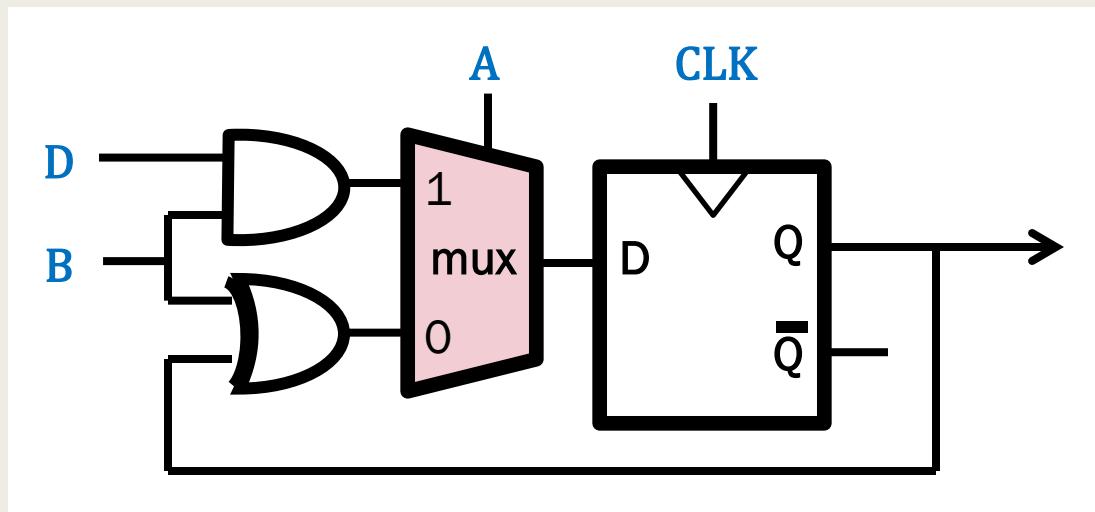


$$\begin{aligned}
 Q(t+1) &= \overline{A} \overline{B} Q(t) + \overline{A} B \overline{Q(t)} + ABD = \overline{A} (\overline{B} Q(t) + B \overline{Q(t)}) + ABD \\
 &= \overline{A} (B \oplus Q(t)) + A(BD)
 \end{aligned}$$

Επιλεγμένη άσκηση: το A-B Flip-Flop

- Από την **εξίσωση Boole** για την επόμενη κατάσταση $Q(t+1)$ στο **σχηματικό διάγραμμα** του νέου Flip-Flop
 - Να σχεδιάσετε το νέο Flip-Flop χρησιμοποιώντας στη συνδυαστική λογική έναν πολυπλέκτη 2 σε 1, μία πύλη XOR και όποια άλλη πύλη απαιτείται

$$\begin{aligned} Q(t+1) &= \overline{A}\overline{B}Q(t) + \overline{A}B\overline{Q(t)} + ABD = \overline{A}(\overline{B}Q(t) + B\overline{Q(t)}) + ABD \\ &= \overline{A}(B \oplus Q(t)) + A(BD) \end{aligned}$$



Επιλεγμένη άσκηση 2: το A-B Flip-Flop

- Το **A-B Flip-Flop** δέχεται ένα σήμα CLK και δύο σύγχρονες εισόδους (το *A* και το *B*).
- Κατά την ανερχόμενη ακμή του CLK ενημερώνει την έξοδο *Q*, σύμφωνα με τις τιμές που έχουν οι είσοδοι *A* και *B*, ως εξής:
 - Όταν οι είσοδοι *A* και *B* έχουν και οι δύο την τιμή 0, η έξοδος *Q* διατηρεί την προηγούμενη τιμή της (*hold*)
 - Όταν η είσοδος *A* έχει την τιμή 0 και η είσοδος *B* έχει την τιμή 1, η έξοδος *Q* εναλλάσσει την τιμή της με το συμπλήρωμα της προηγούμενης τιμής της (*toggle*)
 - Όταν η είσοδος *A* έχει την τιμή 1 και η είσοδος *B* έχει την τιμή 0, η έξοδος *Q* αντιγράφει την τιμή της εισόδου *D* (*load*)
 - Όταν οι είσοδοι *A* και *B* έχουν και οι δύο την τιμή 1, η έξοδος *Q* παίρνει την τιμή 1 (*set*)
- Να σχεδιάσετε το νέο Flip-Flop χρησιμοποιώντας στη συνδυαστική λογική έναν πολυπλέκτη 2 σε 1, μία πύλη XOR και όποια άλλη πύλη απαιτείται

Επιλεγμένη άσκηση 2: το A-B Flip-Flop

- Από τον **Πίνακα Λειτουργιών** στον **Πίνακα Αλήθειας**

AB	Λειτουργία
00	HOLD
01	TOGGLE
10	LOAD
11	SET

A	B	D	Q(t)	Q(t+1)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0

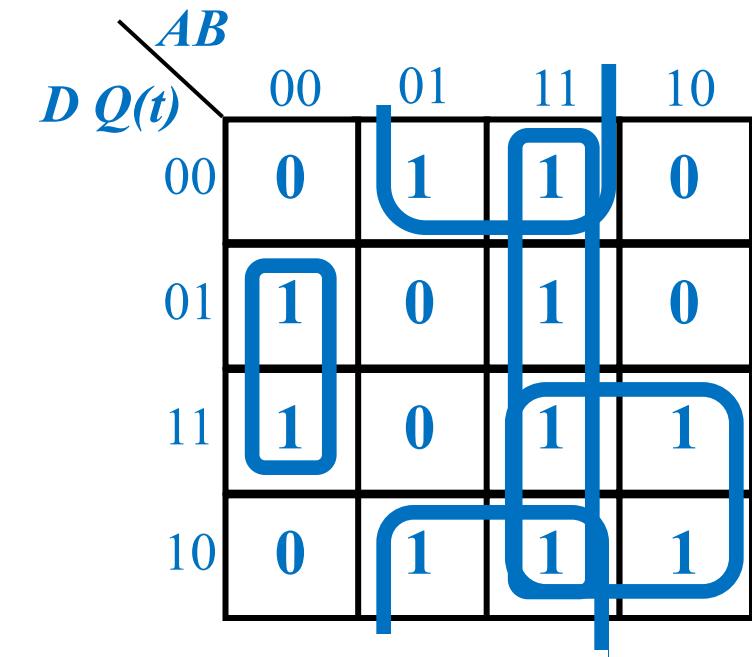
A	B	D	Q(t)	Q(t+1)
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Επιλεγμένη άσκηση 2: το A-B Flip-Flop

- Από τον **Πίνακα Αλήθειας** στον **Χάρτη Karnaugh** και την **εξίσωση Boole** για την επόμενη κατάσταση $Q(t+1)$

A	B	D	$Q(t)$	$Q(t+1)$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0

A	B	D	$Q(t)$	$Q(t+1)$
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

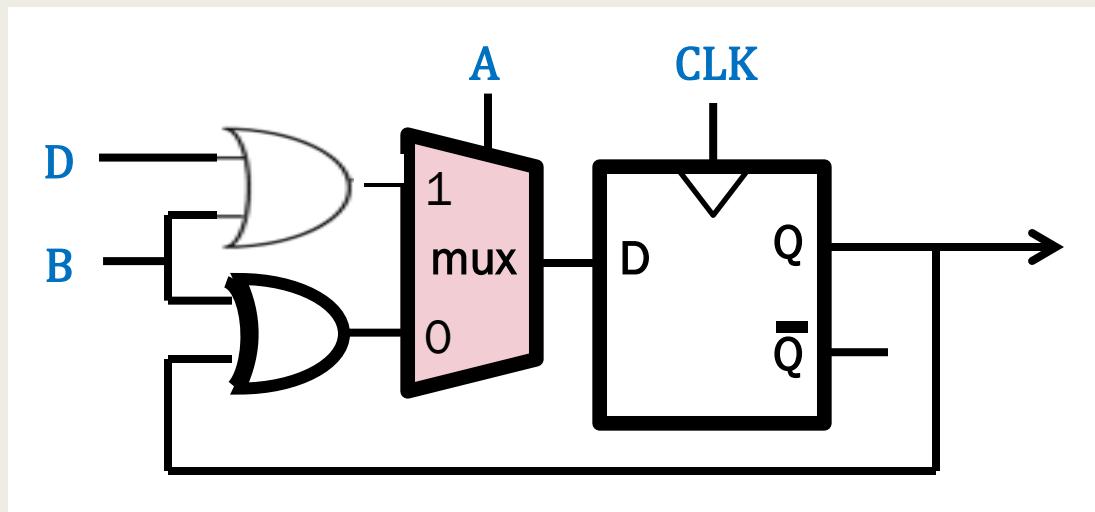


$$\begin{aligned}
 Q(t+1) &= \overline{AB}Q(t) + B\overline{Q(t)} + AB + AD = \overline{AB}Q(t) + (\overline{A} + A)B\overline{Q(t)} + AB + AD = \\
 &= \overline{AB}Q(t) + \overline{AB}\overline{Q(t)} + AB\overline{Q(t)} + AB + AD = \overline{A}(\overline{B}Q(t) + B\overline{Q(t)}) + A(B\overline{Q(t)} + B + D) = \\
 &= \overline{A}(B \oplus Q(t)) + A(B + D)
 \end{aligned}$$

Επιλεγμένη άσκηση 2: το A-B Flip-Flop

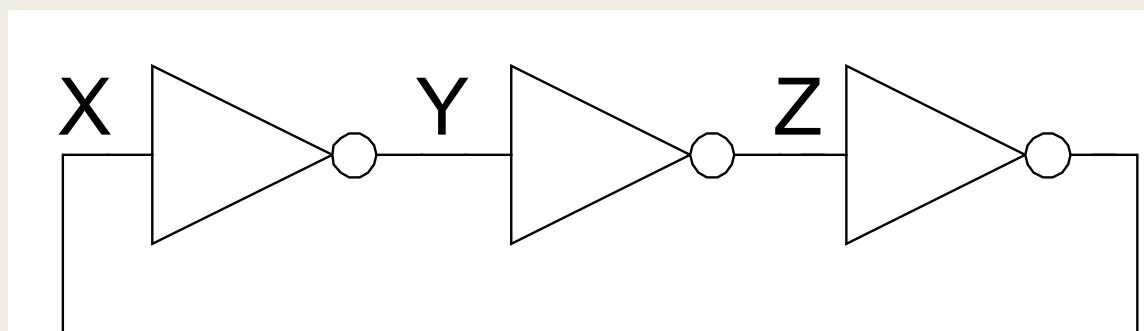
- Από την **εξίσωση Boole** για την επόμενη κατάσταση $Q(t+1)$ στο **σχηματικό διάγραμμα** του νέου Flip-Flop
 - Να σχεδιάσετε το νέο Flip-Flop χρησιμοποιώντας στη συνδυαστική λογική έναν πολυπλέκτη 2 σε 1, μία πύλη XOR και όποια άλλη πύλη απαιτείται

$$Q(t+1) = \bar{A}(B \oplus Q(t)) + A(B + D)$$



Ασταθή ακολουθιακά κυκλώματα*

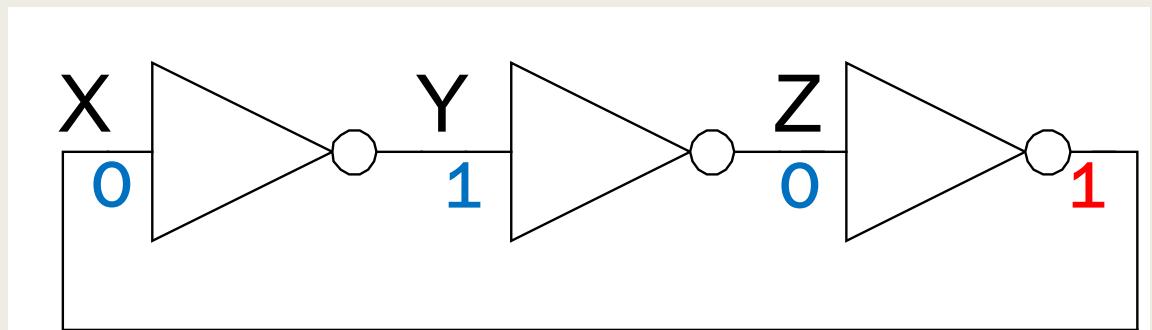
- Τα ασταθή κυκλώματα δεν σταθεροποιούνται σε μία κατάσταση, αλλά ταλαντώνται
- Παράδειγμα: ο ταλαντωτής δακτυλίου (ring oscillator)
 - Κατασκευάζεται με **3 διασυνδεδεμένους αντιστροφείς** σε μία **κυκλική διαδρομή**
 - Αρκεί το πλήθος των αντιστροφών να είναι **περιπτό**
 - Δεν υπάρχει είσοδος που να ελέγχει την κατάσταση του κυκλώματος
 - Θεωρούμε έναν κόμβο ως έξοδο, έστω η έξοδος X



* Παράδειγμα 3.3

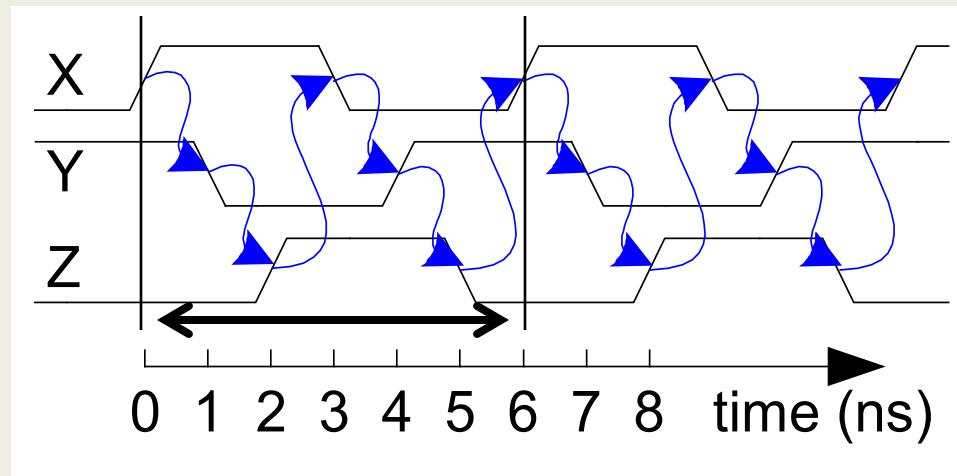
Ασταθή ακολουθιακά κυκλώματα

- Λειτουργία του ταλαντωτή δακτυλίου (ring oscillator)
 - Θεωρούμε ότι ο κόμβος X πηγαίνει ανεξέλεγκτα στην τιμή 0, όταν ενεργοποιούμε το κύκλωμα για πρώτη φορά
 - Ο κόμβος Y πηγαίνει στην τιμή 1 και ο κόμβος Z στην τιμή 0
 - Μετά την ολοκλήρωση της κυκλικής διαδρομής η νέα τιμή του κόμβου X είναι 1
 - που είναι συμπληρωματική της προηγούμενης τιμής ($X=0$)
 - Άρα το κύκλωμα αδυνατεί να σταθεροποιηθεί σε μία κατάσταση, αλλά **ταλαντώνεται**



Ασταθή ακολουθιακά κυκλώματα

- Περίοδος του ταλαντωτή δακτυλίου (ring oscillator)
 - Η περίοδος της ταλάντωσης εξαρτάται από την καθυστέρηση διάδοσης του αντιστροφέα
 - η συγκεκριμένη καθυστέρηση εξαρτάται με τη σειρά της από τον τρόπο κατασκευής του αντιστροφέα, την τάση τροφοδοσίας και τη θερμοκρασία
 - είναι δύσκολο να προβλεφθεί με ακρίβεια η περίοδος του ταλαντωτή δακτυλίου (δεν είναι ταλαντωτής ακριβείας)
 - Εάν η καθυστέρηση διάδοσης ενός αντιστροφέα είναι 1 ns, η περίοδος είναι 6 ns

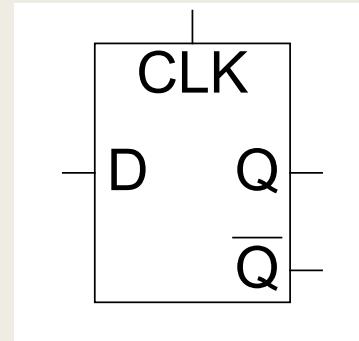


D Latch

- Πίνακας Αλήθειας

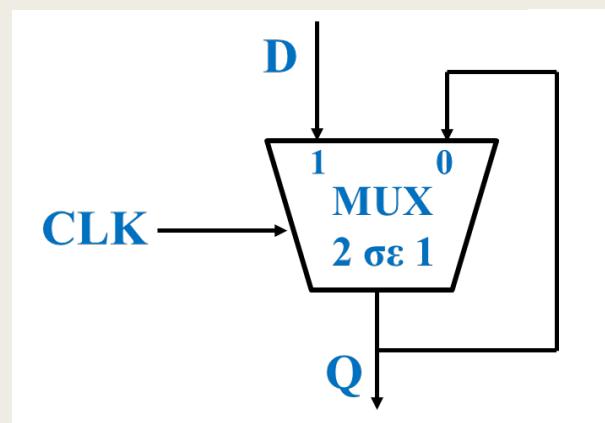
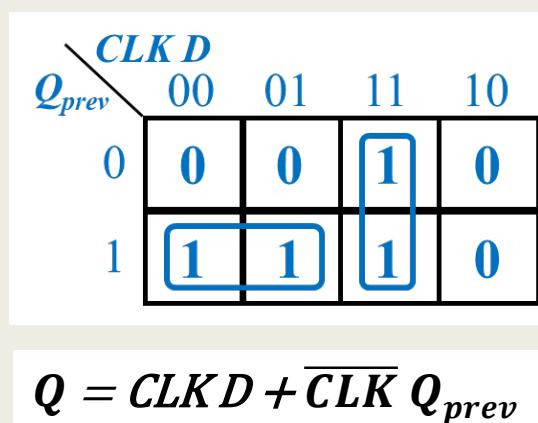
CLK	D	D'	S	R	Q	\bar{Q}
0	X	X'	0	0	Q_{prev}	\bar{Q}_{prev}
1	0	1	0	1	0	1
1	1	0	1	0	1	0

- Σύμβολο



- Υλοποίηση με πολυπλέκτη 2 σε 1

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

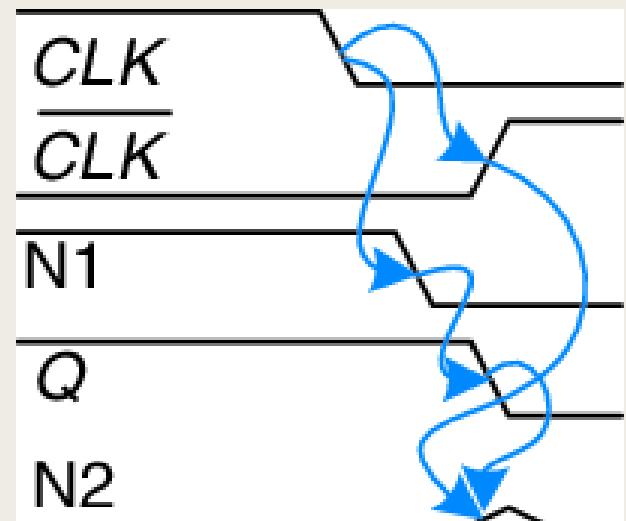
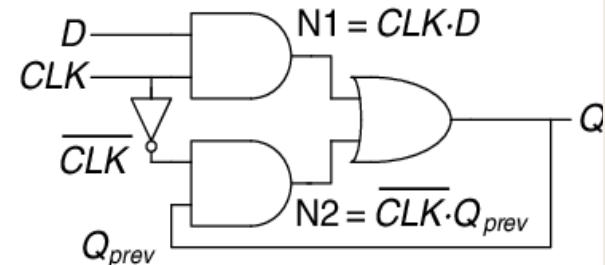


Πιθανόν να εμφανίσει μια συνθήκη συναγωνισμού!

Συνθήκες συναγωνισμού*

- Το D-Latch στην υλοποίηση με πολυπλέκτη 2 σε 1 πιθανόν να εμφανίσει μια **συνθήκη συναγωνισμού** (race condition)
- Έστω ότι το D-Latch αρχικά είναι σταθεροποιημένο στην **κατάσταση 1** ($CLK = D = N1 = Q = 1$)
- Στην κατερχόμενη ακμή του CLK (από το 1 στο 0) το D-Latch πρέπει φυσιολογικά να αποθηκεύσει την προηγούμενη κατάστασή του $Q_{prev} = 1$
- Αν όμως η καθυστέρηση διάδοσης μέσω της πύλης NOT (από **CLK σε \overline{CLK}**) είναι μεγαλύτερη της καθυστέρησης διάδοσης μέσω των πυλών AND και OR (από **CLK σε Q**), τότε το **Q** θα γίνει 0 πριν το **CLK** γίνει 1 και θα αποθηκεύσει λανθασμένα ως προηγούμενη κατάσταση το 0, αντί το 1
- Σε μια τέτοια περίπτωση, η άνοδος του κόμβου N2 δεν θα συμβεί ποτέ, και η έξοδος **Q** θα παραμείνει κολλημένη στο 0

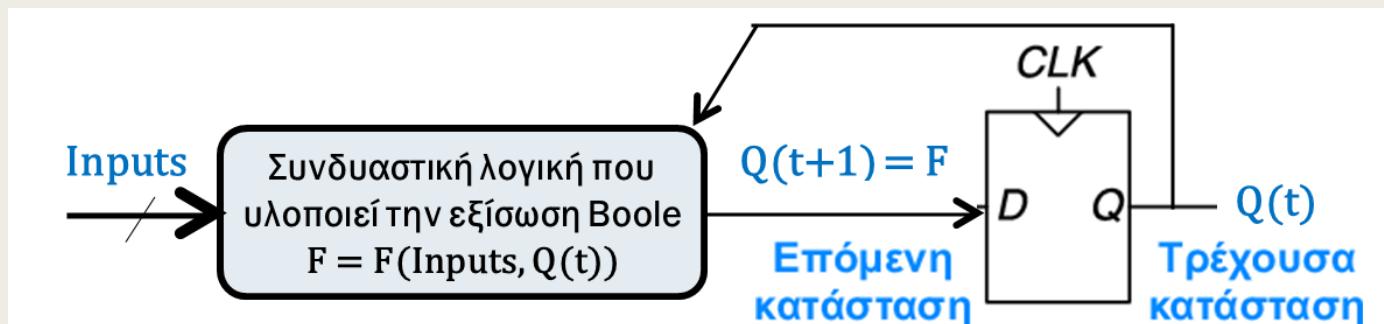
$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$



* Παράδειγμα 3.4

Σύγχρονα ακολουθιακά κυκλώματα

- Η ρίζα του κακού είναι οι **κυκλικές διαδρομές** που δημιουργούν:
 - ανεπιθύμητες *συνθήκες συναγωνισμού* ή *ασταθή συμπεριφορά*
- Οι κυκλικές διαδρομές είναι χαρακτηριστικό των **ασύγχρονων ακολουθιακών κυκλωμάτων**, όπου η έξοδος συνδέεται κατευθείαν στην είσοδο με μία **διαδρομή ανάδρασης**
- Αυτού του είδους τα προβλήματα αντιμετωπίζονται αποτελεσματικά με την **εισαγωγή καταχωρητών** σε ένα ή περισσότερα σημεία της διαδρομής ανάδρασης
 - το κύκλωμα μετασχηματίζεται σε μια συνδεσμολογία συνδυαστικής λογικής και καταχωρητών που ονομάζεται **σύγχρονο ακολουθιακό κύκλωμα**



Προδιαγραφές σύγχρονων ακολουθιακών κυκλωμάτων

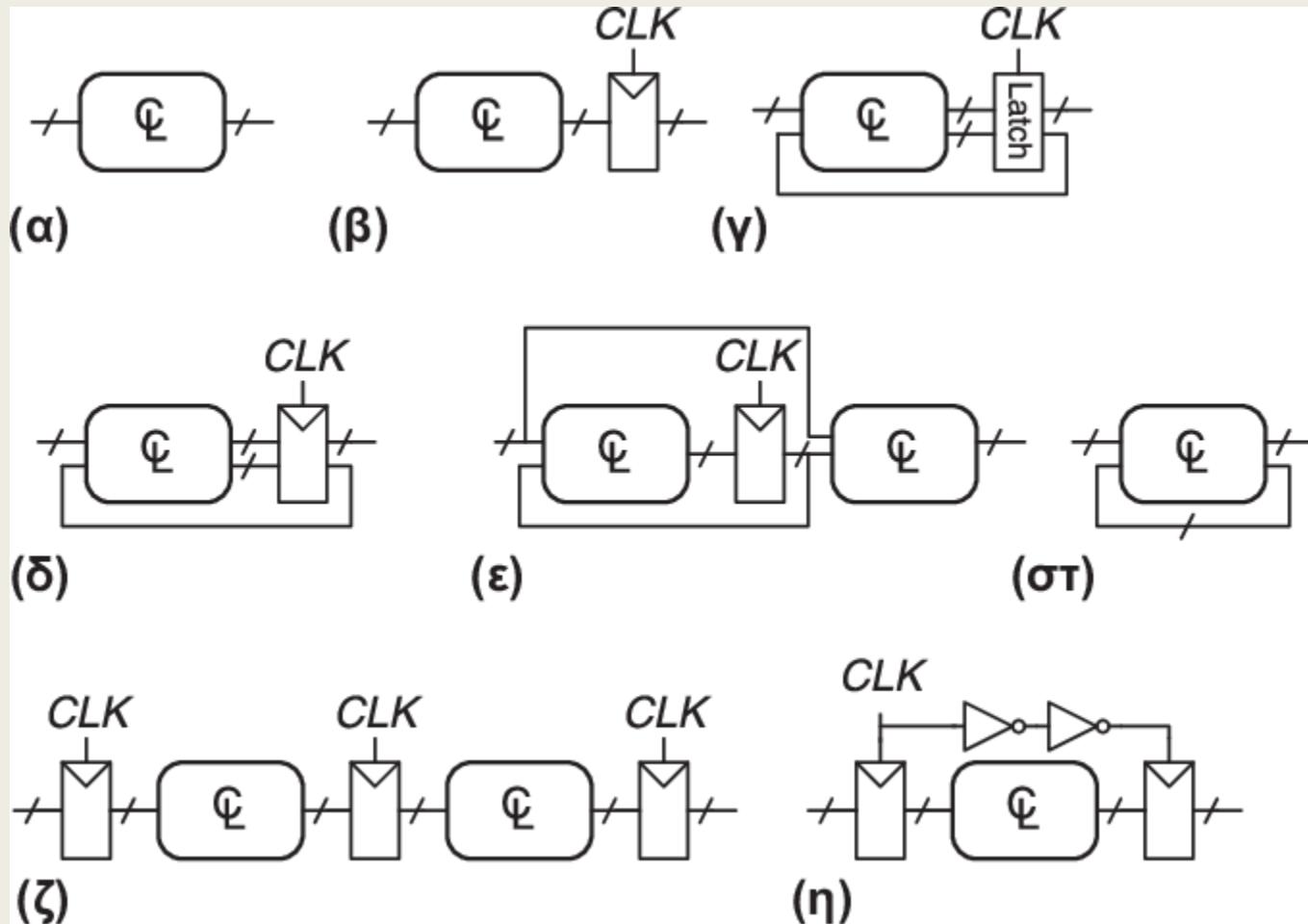
- Η κατάσταση του σύγχρονου ακολουθιακού κυκλώματος μεταβάλλεται στην **ανερχόμενη ακμή του CLK**, αφού πρώτα η συνδυαστική λογική αποκτήσει σταθερές τιμές στις εξόδους της
- Η **λειτουργική προδιαγραφή** περιγράφει λεπτομερώς την **επόμενη κατάσταση $Q(t+1)$** και την **τιμή κάθε εξόδου** για κάθε πιθανό συνδυασμό τιμών της **τρέχουσας κατάστασης $Q(t)$** και των **εισόδων**
- Η **προδιαγραφή χρονισμού για κάθε D Flip-Flop** περιλαμβάνει:
 - τον χρόνο που μεσολαβεί από την ανερχόμενη ακμή του CLK έως τη μεταβολή της εξόδου
 - ένα άνω φράγμα t_{pcq} (**time of propagation from clock to Q**)
 - ένα κάτω φράγμα t_{ccq} (**time of contamination from clock to Q**)
 - τον χρόνο που πρέπει να είναι σταθερή η είσοδος δεδομένων **D πριν και μετά** την ανερχόμενη ακμή του CLK
 - τον χρόνο σταθεροποίησης (πριν) t_{setup} (**set-up time**)
 - τον χρόνο διατήρησης (μετά) t_{hold} (**hold time**)

Προδιαγραφές σύγχρονων ακολουθιακών κυκλωμάτων

- Οι κανόνες της **σύνθεσης σύγχρονων ακολουθιακών κυκλωμάτων** ορίζουν ότι ένα κύκλωμα είναι σύγχρονο ακολουθιακό κύκλωμα αν αποτελείται από αλληλοσυνδεόμενα στοιχεία κυκλωμάτων τέτοια ώστε να ισχύουν τα εξής:
 - κάθε στοιχείο του κυκλώματος είναι είτε καταχωρητής, είτε συνδυαστικό κύκλωμα
 - τουλάχιστον ένα στοιχείο του κυκλώματος είναι καταχωρητής
 - όλοι οι καταχωρητές δέχονται το ίδιο σήμα ρολογιού
 - κάθε κυκλική διαδρομή περιέχει τουλάχιστον έναν καταχωρητή
- Κάθε ακολουθιακό κύκλωμα που δεν είναι σύγχρονο ονομάζεται **ασύγχρονο ακολουθιακό κύκλωμα**

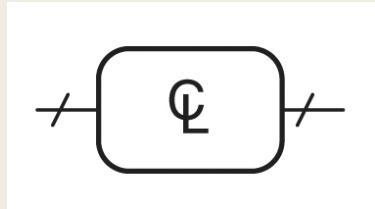
Παράδειγμα 3.5

Ποια από τα παρακάτω κυκλώματα είναι συνδυαστικά κυκλώματα, σύγχρονα ακολουθιακά κυκλώματα ή ασύγχρονα ακολουθιακά κυκλώματα;

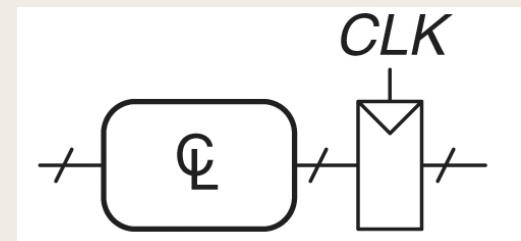


Παράδειγμα 3.5

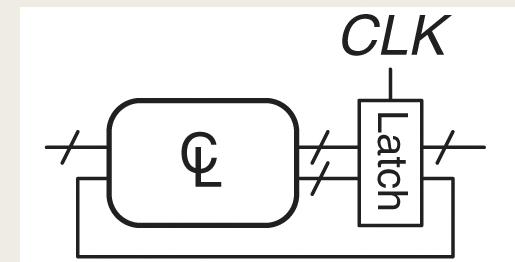
- Το κύκλωμα (α) **είναι συνδυαστικό** δεν περιλαμβάνει ούτε καταχωρητές, ούτε κυκλική διαδρομή



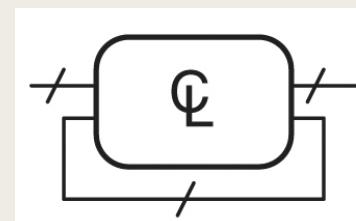
- Το κύκλωμα (β) είναι ένα απλό **σύγχρονο ακολουθιακό κύκλωμα** (συνδυαστική λογική και καταχωρητής)



- Το κύκλωμα (γ) λόγω του Latch είναι **ασύγχρονο ακολουθιακό κύκλωμα**



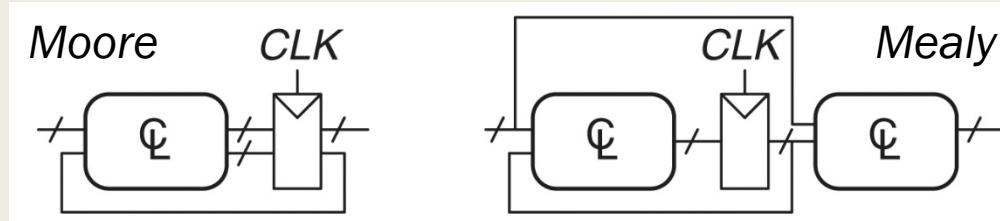
- Το κύκλωμα (στ) λόγω της κυκλικής διαδρομής ανάδρασης είναι **ασύγχρονο ακολουθιακό κύκλωμα**



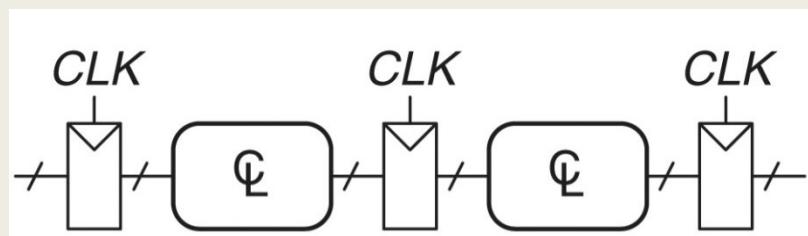
Παράδειγμα 3.5

- Τα κυκλώματα (δ), (ε) και (ζ) παραπέμπουν σε **σύγχρονη ακολουθιακή λογική**

- πρόκειται για τις δύο μορφές **μηχανών πεπερασμένων καταστάσεων**



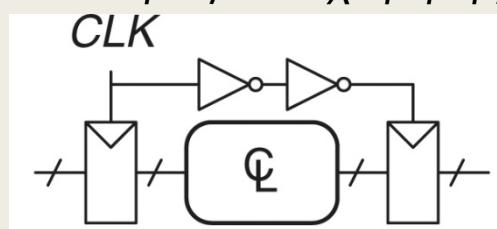
- πρόκειται για συνδεσμολογία **διοχέτευσης (pipelining)**



Σχεδίαση στο επίπεδο
της μεταφοράς καταχωρητών
Register Transfer Level – RTL

- Το κύκλωμα (η) **δεν είναι σύγχρονο ακολουθιακό κύκλωμα** με την αυστηρή έννοια του όρου

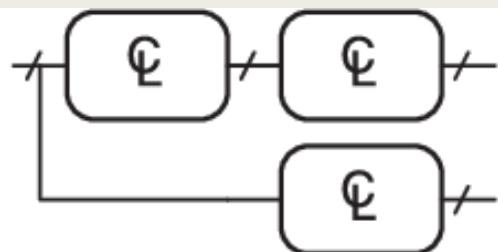
- ο δεύτερος καταχωρητής δέχεται το σήμα με καθυστέρηση δύο πυλών NOT



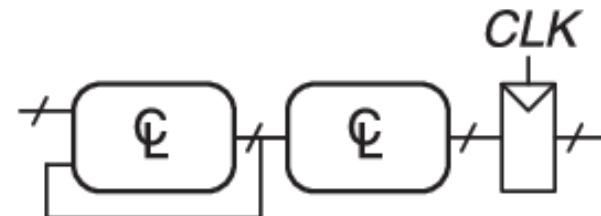
Επιλεγμένες ασκήσεις

■ Ασκηση 3.18

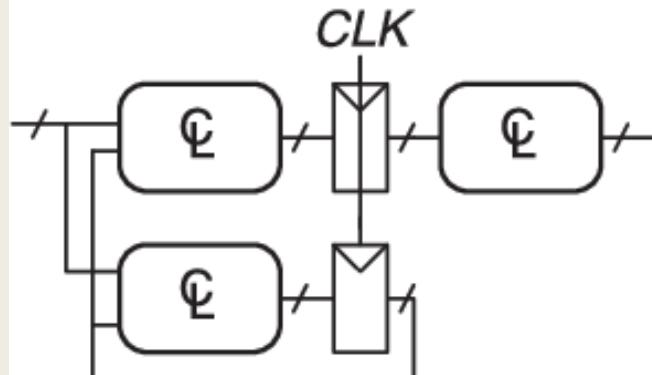
Ποια από τα κυκλώματα της Εικόνας 3.68 είναι σύγχρονα ακολουθιακά κυκλώματα; Αιτιολογήστε την απάντησή σας



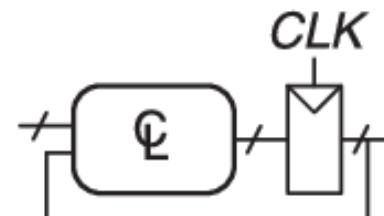
(α)



(γ)



(β)



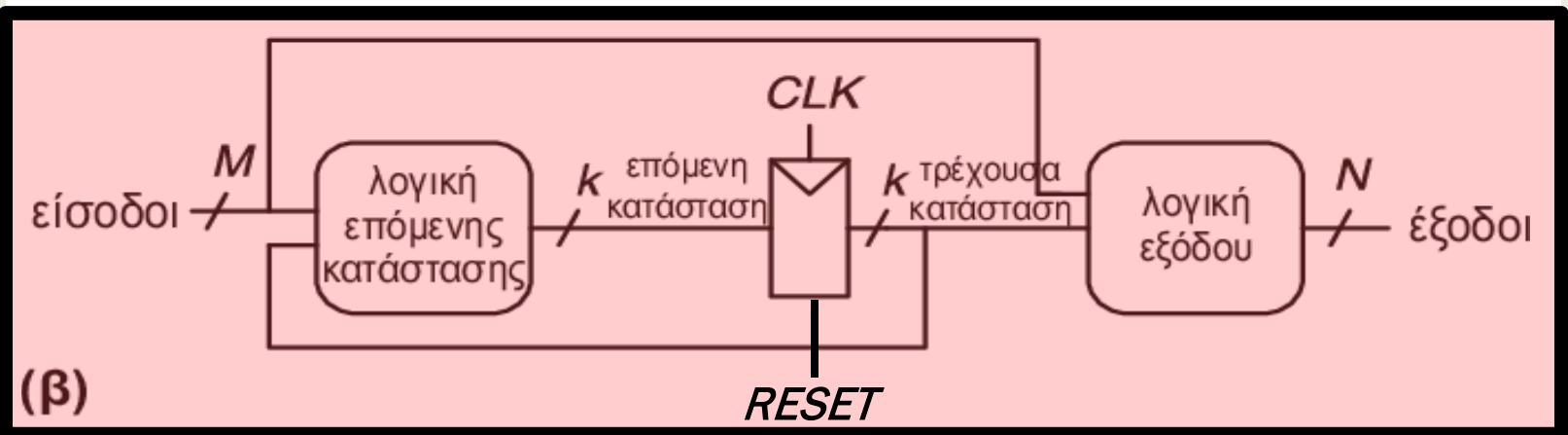
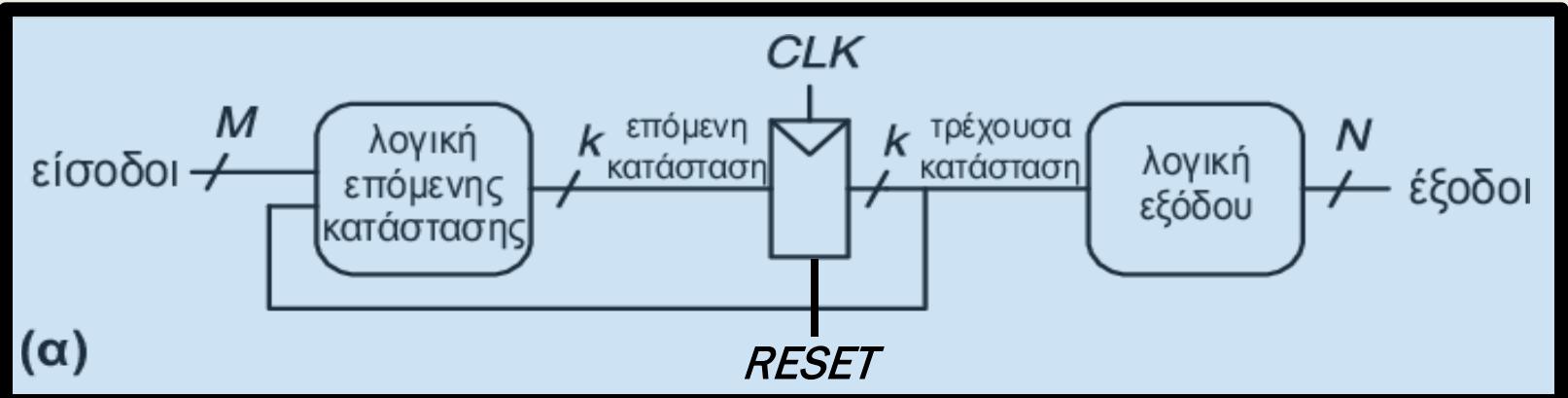
(δ)

Μηχανές πεπερασμένων καταστάσεων

- Μία **μηχανή πεπερασμένων καταστάσεων** (finite state machines, FSM) είναι ένα σύγχρονο ακολουθιακό κύκλωμα που διαθέτει **Μεισόδους**, **Νεξόδους**, ένα σήμα ρολογιού **CLK** και ένα σήμα επαναφοράς **RESET**, και συνίσταται από:
 - έναν **καταχωρητή καταστάσεων** (state register) μεγέθους k bit, που αποθηκεύει από k μέχρι το πολύ 2^k διακριτές καταστάσεις
 - μία **λογική επόμενης κατάστασης** (*next state logic*) που υπολογίζει την επόμενη κατάσταση $Q(t+1)$ ως συνάρτηση της τρέχουσας κατάστασης $Q(t)$ και των M εισόδων
 - μία **λογική εξόδου** (*output logic*) που υπολογίζει τις τιμές των εξόδων ως συνάρτηση της τρέχουσας κατάστασης $Q(t)$ και προαιρετικά των M εισόδων
- Το σήμα επαναφοράς **RESET** στον καταχωρητή καταστάσεων είναι αναγκαίο ώστε να επιβάλλουμε μια **αρχική κατάσταση** στη μηχανή FSM, όταν την ενεργοποιούμε για πρώτη φορά
- Το προαιρετικό σήμα έγκρισης **EN** χρησιμοποιείται για να καθορίζει αν η μηχανή FSM αλλάξει κατάσταση κατά την επόμενη ακμή του CLK

Μηχανές πεπερασμένων καταστάσεων

- Υπάρχουν δύο τύποι μηχανών πεπερασμένης καταστάσης
 - **(α) Μηχανές τύπου Moore** (οι έξοδοι εξαρτώνται μόνο από την παρούσα κατάσταση)
 - **(β) Μηχανές τύπου Mealy** (οι έξοδοι εξαρτώνται από την παρούσα κατάσταση *και* τις εισόδους)



Μηχανές πεπερασμένων καταστάσεων

- Η μηχανή πεπερασμένων καταστάσεων τύπου Mealy είναι πιο γενική από τη μηχανή τύπου Moore
- Οι μηχανές πεπερασμένων καταστάσεων **τύπου Moore**
 - έχουν *περισσότερες καταστάσεις*
 - συνήθως πλεονεκτούν σε ταχύτητα και μέγεθος της *λογικής εξόδου*
- Οι μηχανές πεπερασμένων καταστάσεων **τύπου Mealy**
 - έχουν *λιγότερες καταστάσεις*
 - συνήθως πλεονεκτούν σε ταχύτητα και μέγεθος της *λογικής επόμενης κατάστασης*
- Μία μηχανή πεπερασμένων καταστάσεων μπορεί να έχει εξόδους και των δύο τύπων

Μηχανές πεπερασμένων καταστάσεων

- Η απόδοση και το κόστος της μηχανής εξαρτάται από:
 - Το **πλήθος των καταστάσεων** (περισσότερες στη μηχανή τύπου Moore)
 - Την **πολυπλοκότητα των διακλαδώσεων** ανά κατάσταση (μεγαλύτερη στη μηχανή τύπου Mealy)
 - Το **μέγεθος του καταχωρητή καταστάσεων**
 - Το **πλήθος των ψηφίων που αλλάζουν τιμή** από την τρέχουσα κατάσταση στην επόμενη κατάσταση
 - Άρα, από την κωδικοποίηση των καταστάσεων
 - Δυαδική
 - Gray
 - Μονόθερμη (one-hot)

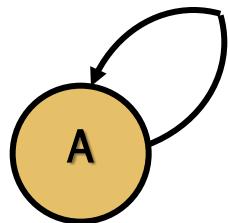
Διάγραμμα μεταβολής κατάστασης

- Το πρώτο στάδιο στη σχεδίαση μίας μηχανής FSM είναι η σχεδίαση του **διαγράμματος μεταβολής κατάστασης** (state transition diagram)
 - Στη σύγχρονη ψηφιακή σχεδίαση το διάγραμμα μεταβολής κατάστασης κωδικοποιείται σε μία γλώσσα περιγραφής υλικού (π.χ. VHDL) και όλα τα υπόλοιπα στάδια της σχεδίασης τα αναλαμβάνει το εργαλείο λογισμικού
- Το διάγραμμα μεταβολής κατάστασης περιγράφει:
 - τις **πιθανές διακριτές καταστάσεις** της μηχανής και πως αυτές μεταβάλλονται σύμφωνα με τις **συνθήκες εισόδου** κατά την ανερχόμενη ακμή του CLK
 - τις αντίστοιχες **εξόδους** τύπου Moore ή Mealy
- Το διάγραμμα μεταβολής κατάστασης απαρτίζεται από:
 - **Κύκλους** που προσδιορίζουν την τρέχουσα κατάσταση $Q(t)$
 - **Βέλη** που προσδιορίζουν τη μετάβαση από την τρέχουσα κατάσταση $Q(t)$ στην επόμενη κατάσταση $Q(t+1)$
 - Όταν υπάρχει μετάβαση με συνθήκη εισόδου, οι τιμές των εισόδων που ικανοποιούν τη συνθήκη γράφονται δίπλα στο βέλος

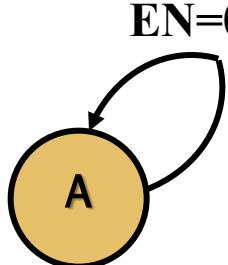
Διάγραμμα μεταβολής κατάστασης

- Παραδείγματα πιθανών **διαγραμμάτων μεταβολής κατάστασης**
 1. Όταν η επόμενη κατάσταση A είναι ίδια με την τρέχουσα κατάσταση A, χωρίς συνθήκη εισόδου
 2. Όταν η επόμενη κατάσταση A είναι ίδια με την τρέχουσα κατάσταση A με συνθήκη εισόδου (π.χ. EN=0)
 3. Όταν η επόμενη κατάσταση B είναι διαφορετική από την τρέχουσα κατάσταση A χωρίς συνθήκη εισόδου
 4. Όταν η επόμενη κατάσταση B είναι διαφορετική από την τρέχουσα κατάσταση A με συνθήκη εισόδου (π.χ. EN=1)

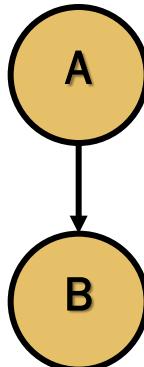
1.



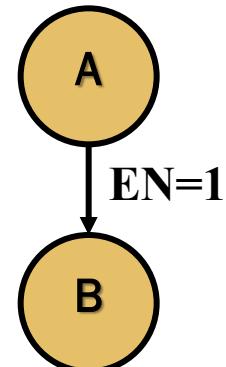
2.



3.

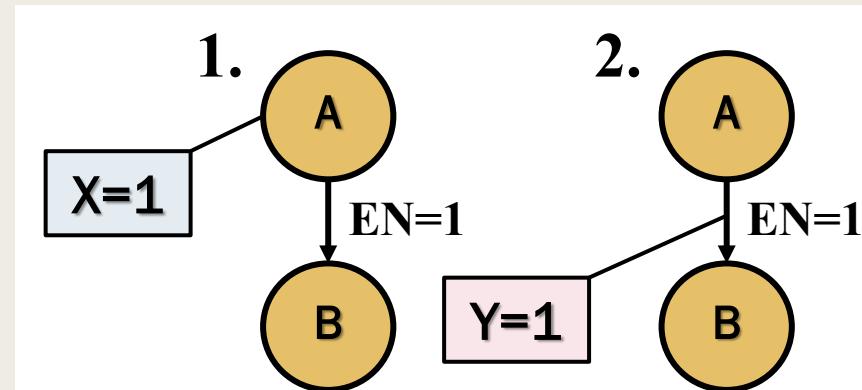


4.



Διάγραμμα μεταβολής κατάστασης

- Επιπλέον, το διάγραμμα μεταβολής κατάστασης απαρτίζεται από:
 - **Πλαίσια** εντός των οποίων γράφονται οι τιμές των εξόδων της μηχανής *FSM*, που εξαρτώνται αποκλειστικά από την τρέχουσα κατάσταση (**έξοδοι τύπου Moore**)
 - Τα πλαίσια αυτά συνδέονται με τον κύκλο της σχετικής τρέχουσας κατάστασης
 - **Πλαίσια** εντός των οποίων γράφονται οι τιμές των εξόδων της μηχανής *FSM*, που εξαρτώνται από την τρέχουσα κατάσταση και την αντίστοιχη συνθήκη εισόδου (**έξοδοι τύπου Mealy**).
 - Τα πλαίσια αυτά συνδέονται με το αντίστοιχο βέλος, δίπλα στις τιμές των εισόδων που ικανοποιούν τη συνθήκη
 - **Παραδείγματα χρήσης πλαισίων:**
 1. Η έξοδος $X=1$ στην κατάσταση A (**τύπου Moore**)
 2. Η έξοδος $Y=1$ στην κατάσταση A, όταν $EN=1$ ($EN=1$) (**τύπου Mealy**)

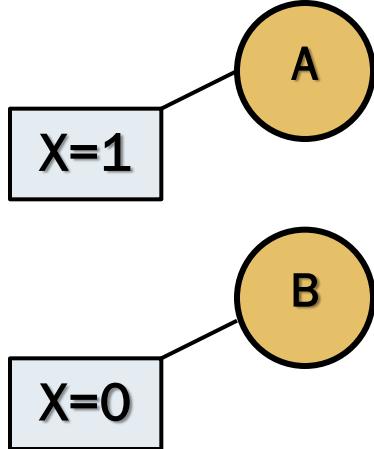


Η χρήση των πλαισίων διευκολύνει πολύ στην κατανόηση ιδίως όταν το πλήθος των σχετικών εξόδων είναι μεγάλο

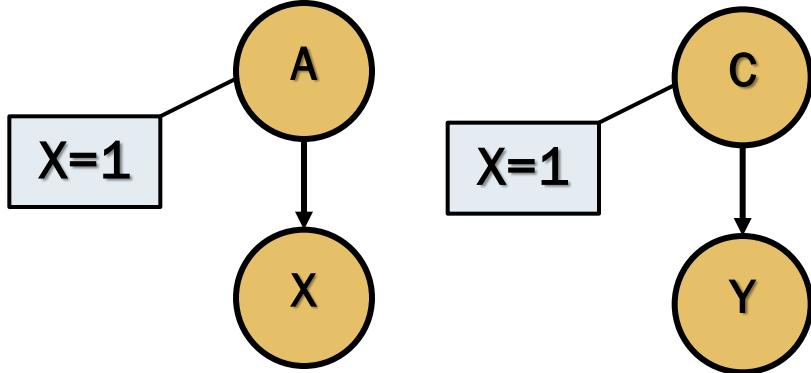
Διαδικασία σχεδίασης μηχανών FSM

- Βήμα 1: Προσδιορίζουμε τις **εισόδους**, τις **εξόδους**, και τις **διακριτές καταστάσεις**
 - Δύο τρέχουσες καταστάσεις χαρακτηρίζονται ως διακριτές μεταξύ τους, εάν :
 - τουλάχιστον μία έξοδος, που εξαρτάται αποκλειστικά από την τρέχουσα κατάσταση, έχει διαφορετική τιμή (παράδειγμα 1), ή/και
 - έχουν τις ίδιες τιμές στις εξόδους, αλλά διαφορετική επόμενη κατάσταση που είναι ανεξάρτητη από τις εισόδους (παράδειγμα 2)
 - Μη διακριτές καταστάσεις ενοποιούνται σε μία κατάσταση, ώστε να προκύψει ελαχιστοποίηση των καταστάσεων

1.



2.

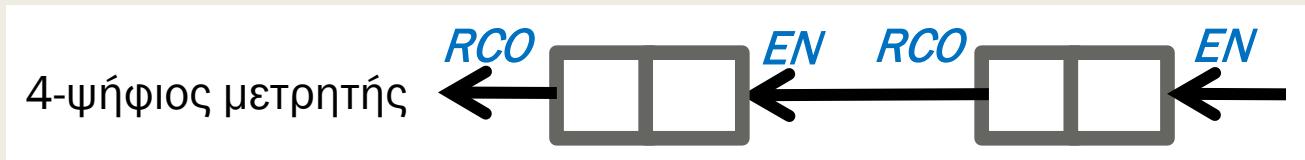


Διαδικασία σχεδίασης μηχανών FSM

- Βήμα 1 (συνέχεια): Προσδιορίζουμε τις **μεταβάσεις** ανάμεσα στις διακριτές καταστάσεις με τις αντίστοιχες **συνθήκες εισόδου**
- Προσδιορίζουμε τις τιμές των **εξόδων** που εξαρτώνται:
 - αποκλειστικά από την **τρέχουσα κατάσταση** (**έξοδοι τύπου Moore**)
 - από την **τρέχουσα κατάσταση** και την αντίστοιχη **συνθήκη εισόδου** (**έξοδοι τύπου Mealy**)
- Βήμα 2: Σχεδιάζουμε το **διάγραμμα μεταβολής κατάστασης**
- Βήμα 3: Σχεδιάζουμε τον **πίνακα μεταβολής κατάστασης**
- Βήμα 4: Επιλέγουμε την **κωδικοποίηση των καταστάσεων** και ενημερώνουμε **τον πίνακα μεταβολής κατάστασης** (μετατρέπεται σε πίνακα αλήθειας)
- Βήμα 5: Απλοποιούμε με K-map και βρίσκουμε τις **εξισώσεις Boole των μεταβλητών των επόμενων καταστάσεων** (συμβολίζουμε με Q^*)
- Βήμα 6: Σχεδιάζουμε **τον πίνακα αληθείας** για τις εξόδους
 - **Týpo Moore**: παρούσες καταστάσεις, έξοδοι
 - **Týpo Mealy**: παρούσες καταστάσεις, είσοδοι, έξοδοι
- Βήμα 7: Απλοποιούμε με K-map και βρίσκουμε τις **εξισώσεις Boole των εξόδων**
- Βήμα 8: Σχεδιάζουμε το **σχηματικό διάγραμμα**

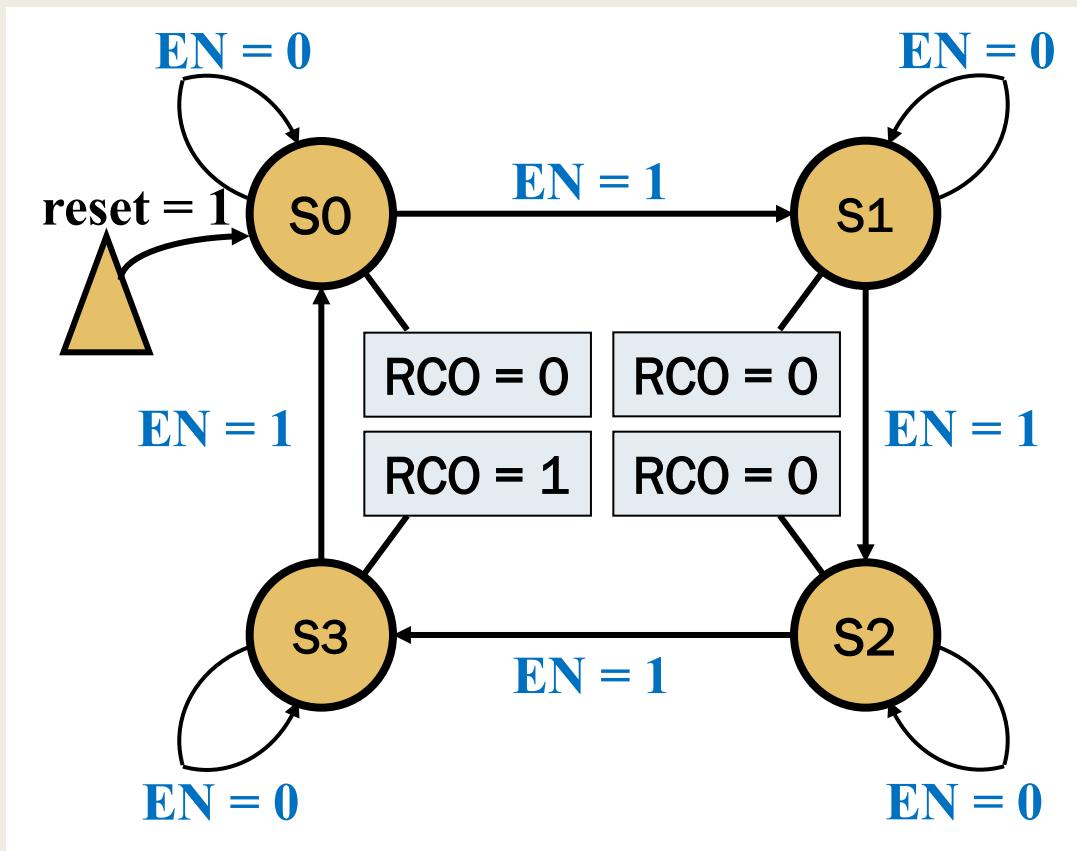
2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 1: Προσδιορίζουμε τις εισόδους, τις εξόδους, και τις διακριτές καταστάσεις
- Ο μετρητής μετρά στο δυαδικό σύστημα από το 0 μέχρι το 3
 - Έχει 4 καταστάσεις S_0, S_1, S_2 και S_3
- Το σήμα **RESET** αρχικοποιεί τον μετρητή στην κατάσταση S_0
 - Η αρχικοποίηση γίνεται σύγχρονα ή ασύγχρονα του **CLK** με κατάλληλη επιλογή των **D Flip-Flop**
- Το σήμα έγκρισης **EN** χρησιμοποιείται για να καθορίζει αν ο μετρητής αλλάζει κατάσταση κατά την επόμενη ακμή του **CLK**
 - **EN = 1**: πηγαίνει στην επόμενη κατάσταση
 - **EN = 0**: παραμένει στην τρέχουσα κατάσταση
- Το σήμα **CLK** επιδρά ταυτόχρονα σε όλα τα **D Flip-Flop**
- Η έξοδος **RCO** (ripple carry output) λαμβάνει την τιμή 1 μόνο όταν ο μετρητής βρίσκεται στην κατάσταση S_3
 - Χρησιμοποιείται σε συνδυασμό με το σήμα **EN** για την κατασκευή ενός μεγάλου μετρητή από πολλούς μικρούς



2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 2: Σχεδιάζουμε το διάγραμμα μεταβολής κατάστασης

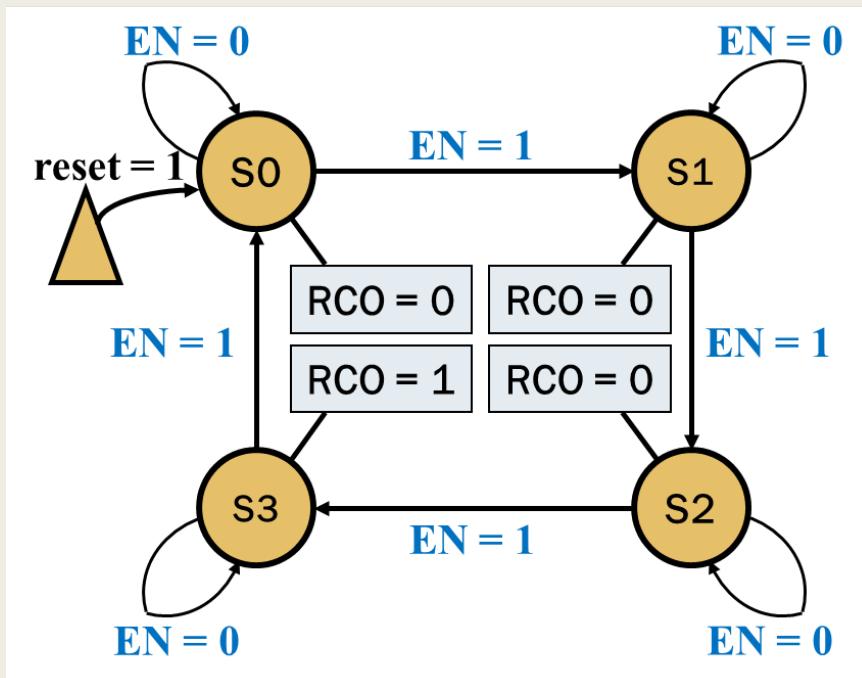


Πίνακας μεταβολής κατάστασης

- Μπορούμε να ορίζουμε το διάγραμμα μεταβολής κατάστασης και με τη μορφή ενός **πίνακα μεταβολής κατάστασης** ο οποίος δείχνει, για κάθε τρέχουσα κατάσταση $Q(t)$ και είσοδο, ποια πρέπει να είναι η επόμενη κατάσταση $Q(t+1)$
 - *Στον πίνακα γίνεται χρήση του συμβόλου X για τις «αδιάφορες» τιμές οποτεδήποτε η επόμενη κατάσταση δεν εξαρτάται από μια συγκεκριμένη είσοδο.*
 - *Η είσοδος **RESET** έχει παραλειφθεί από τις εισόδους του πίνακα*
 - χρησιμοποιούμε ένα D Flip-Flop με δυνατότητα σύγχρονης ή ασύγχρονης επαναφοράς στην αρχική κατάσταση SO

2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 3: Σχεδιάζουμε τον πίνακα μεταβολής κατάστασης



Current State	Inputs	Next State
Q(t)	EN	Q(t+1)
S0	0	S0
S0	1	S1
S1	0	S1
S1	1	S2
S2	0	S2
S2	1	S3
S3	0	S3
S3	1	S0

2-ψήφιος σύγχρονος δυαδικός μετρητής

- **Βήμα 4: Κωδικοποιούμε τις καταστάσεις και ενημερώνουμε τον πίνακα μεταβολής κατάστασης** (μετατρέπεται σε πίνακα αληθείας για τις μεταβλητές επόμενης κατάστασης)
 - Στους δυαδικούς μετρητές, όπου καταστάσεις και έξοδοι ταυτίζονται χρησιμοποιείται η **δυαδική κωδικοποίηση**

Κατάσταση	Δυαδική Κωδικοποίηση
S0	00
S1	01
S2	10
S3	11

Current State	Inputs	Next State
Q1 Q0	EN	Q1* Q0*
0 0	0	0 0
0 0	1	0 1
0 1	0	0 1
0 1	1	1 0
1 0	0	1 0
1 0	1	1 1
1 1	0	1 1
1 1	1	0 0

2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 5: Ελαχιστοποιούμε με K-map και βρίσκουμε τις εξισώσεις Boole των μεταβλητών των επόμενων καταστάσεων
 - συμβολίζουμε με $Q1^*$ και $Q0^*$ ($Q^* = Q(t+1)$)

Current State	Inputs	Next State
Q1 Q0	EN	Q1* Q0*
0 0	0	0 0
0 0	1	0 1
0 1	0	0 1
0 1	1	1 0
1 0	0	1 0
1 0	1	1 1
1 1	0	1 1
1 1	1	0 0

EN	$Q1$	$Q0$	00	01	11	10
0	0	0	1	1		
1	0	1	0		1	

$$Q1^* = \overline{EN}Q1 + Q1\overline{Q0} + EN\overline{Q1}Q0$$

EN	$Q1$	$Q0$	00	01	11	10
0	0	1	1	1		0
1	1	0	0	0		1

$$Q0^* = \overline{EN}Q0 + EN\overline{Q0}$$

2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 5 (συνέχεια): Περαιτέρω απλοποίηση υλικού με τη χρήση των πυλών XOR

Current State	Inputs	Next State
Q1 Q0	EN	Q1* Q0*
0 0	0	0 0
0 0	1	0 1
0 1	0	0 1
0 1	1	1 0
1 0	0	1 0
1 0	1	1 1
1 1	0	1 1
1 1	1	0 0

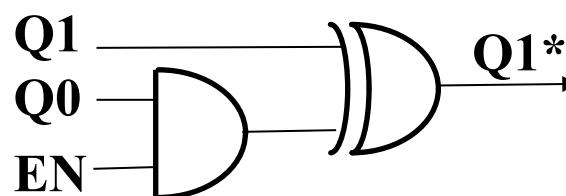
$$Q1^* = \overline{EN}Q1 + Q1\overline{Q0} + EN\overline{Q1}Q0$$

$$Q1^* = \overline{EN} Q1 + Q1 \overline{Q0} + EN \overline{Q1} Q0$$

$$Q1^* = (\overline{EN} + \overline{Q0}) Q1 + EN \overline{Q1} Q0$$

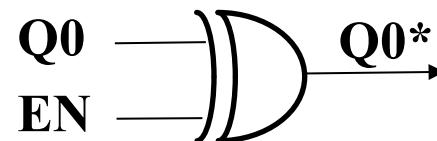
$$Q1^* = (EN \ Q0) \ Q1 + (EN \ Q0) \ \overline{Q1}$$

$$Q1^* = (EN \ Q0) \oplus Q1$$



$$Q0^* = \overline{EN}Q0 + EN\overline{Q0}$$

$$Q0^* = EN \oplus Q0$$

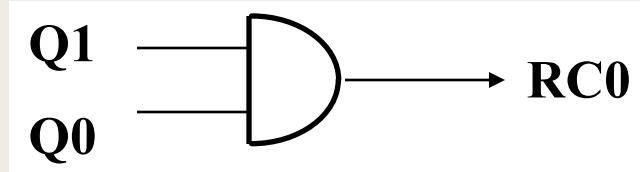


2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 6: Σχεδιάζουμε τον πίνακα αληθείας για τις εξόδους
 - *Tύπου Moore*: παρούσες καταστάσεις, έξοδοι
- Βήμα 7: Βρίσκουμε τις εξισώσεις Boole των εξόδων

Current State	Outputs
Q1 Q0	RC0
0 0	0
0 1	0
1 0	0
1 1	1

$$RC0 = Q1 \cdot Q0$$

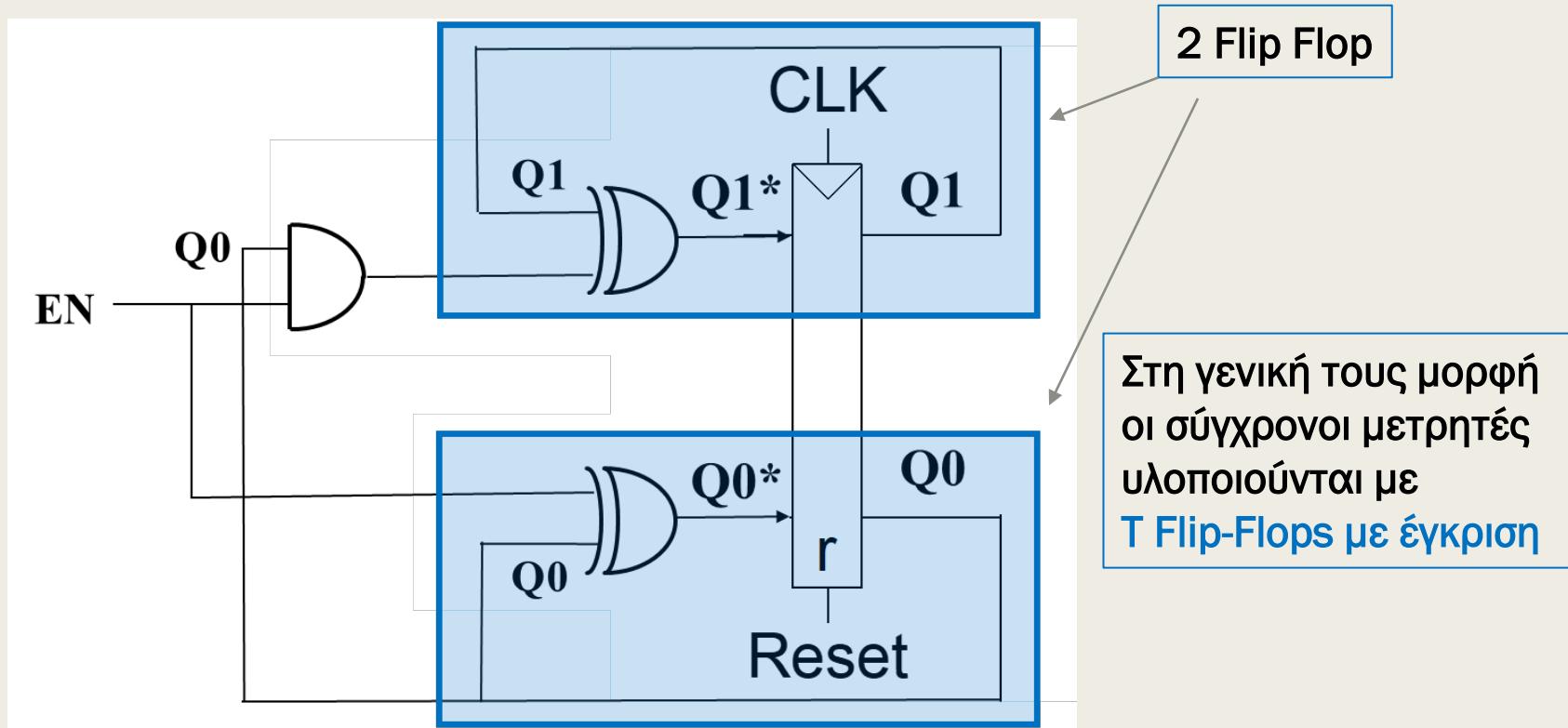


2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 8: Σχεδιάζουμε το σχηματικό διάγραμμα
 - Καταχωρητής καταστάσεων και λογική επόμενης κατάστασης

$$Q1^* = (\text{EN } Q0) \oplus Q1$$

$$Q0^* = \text{EN} \oplus Q0$$



Εμφανίζονται **T Flip-Flop με έγκριση**

T Flip-Flop με έγκριση (enabled)

- Σχεδίαση ενός T Flip-Flop με έγκριση
 - Η είσοδος *ENABLE* (*EN*) είναι ενεργή στο *HIGH* (active *HIGH*)

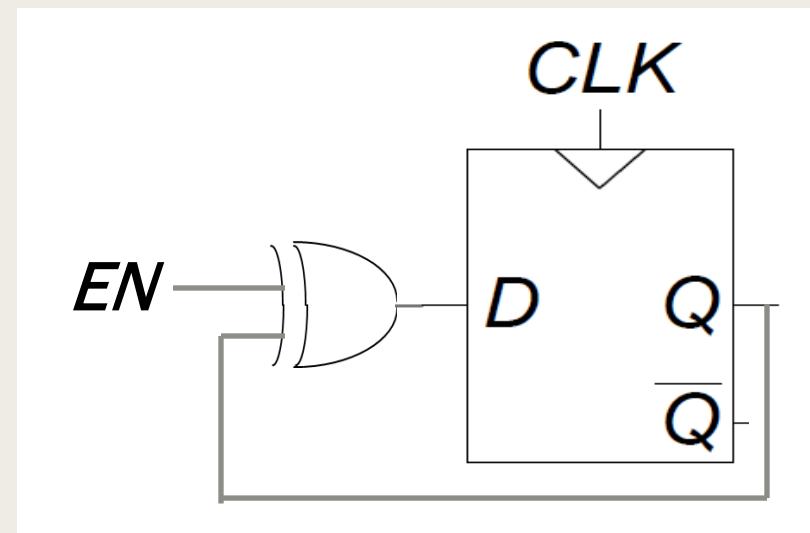
- Πίνακας Αλήθειας

EN	Q(t)	Q(t + 1)	
0	0	0	HOLD
0	1	1	
<hr/>			
1	0	1	TOGGLE
1	1	0	

- Εξίσωση Boole

$$Q(t + 1) = \overline{EN} Q(t) + EN \overline{Q(t)} = EN \oplus Q(t)$$

- Σχηματικό διάγραμμα



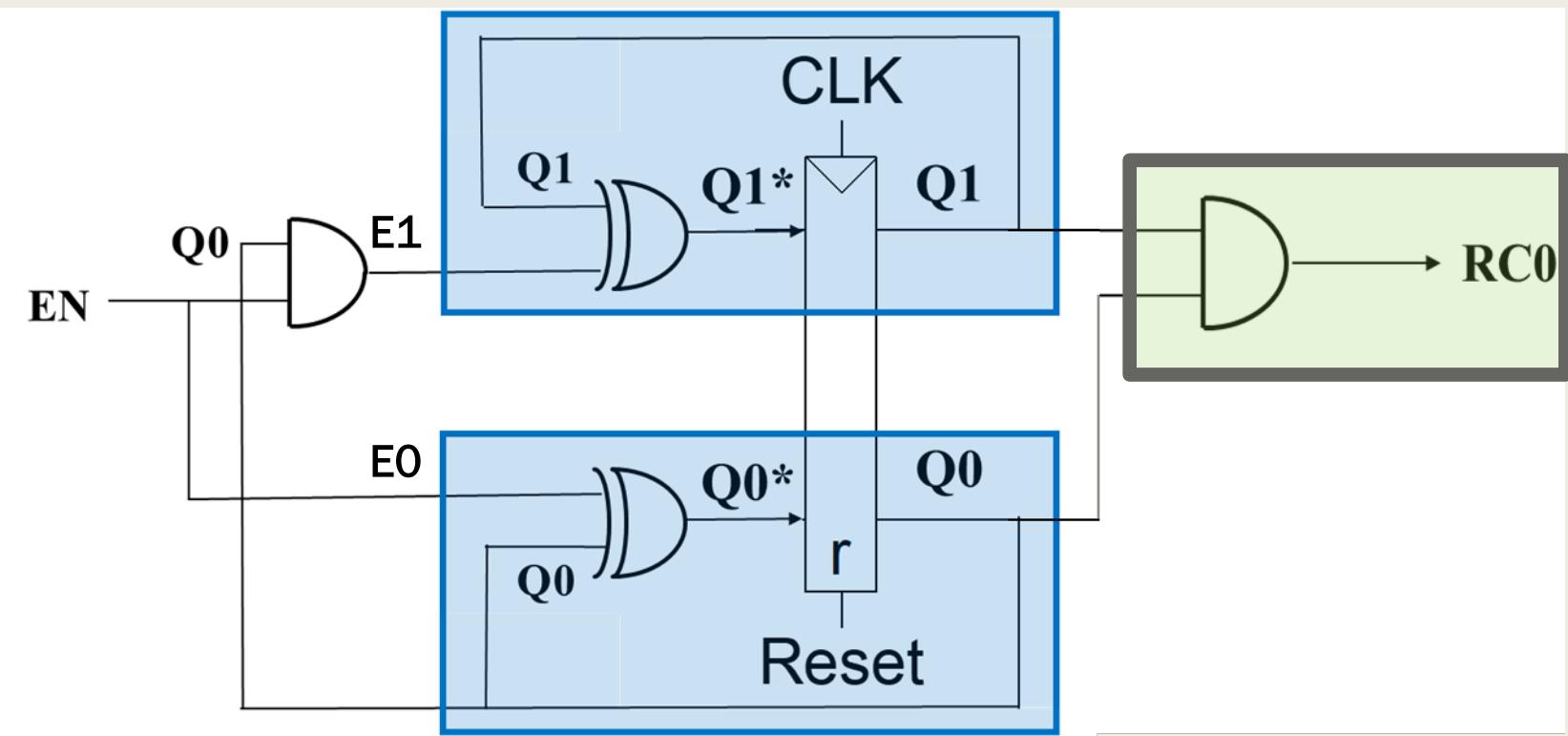
2-ψήφιος σύγχρονος δυαδικός μετρητής

- Βήμα 8 (συνέχεια): Σχεδιάζουμε το σχηματικό διάγραμμα
 - Συμπληρώνουμε και τη λογική εξόδου και τα σήματα έγκρισης $E1$ και $E0$ των T Flip-Flops με έγκριση

$$RC0 = Q1 \bar{Q}0$$

$$E0 = EN$$

$$E1 = EN \bar{Q}0$$



Κωδικοποίηση των καταστάσεων

- Στην πράξη συνήθως χρησιμοποιούνται οι ακόλουθες κωδικοποιήσεις:
 - **Binary (δυαδική)**
 - για δυαδικούς μετρητές, όπου καταστάσεις και έξοδοι ταυτίζονται
 - **Μοναδικού σημαντικού (one-hot) – ένα ξεχωριστό bit ανά κατάσταση**
 - για υλοποιήσεις σε *FPGA* και πλήθος καταστάσεων από 10 μέχρι 30
 - ελαχιστοποιεί τις εξισώσεις *Boole*, αλλά αυξάνει το μέγεθος του καταχωρητή
 - **Gray ή τροποποιημένη Gray**
 - Η πιο διαδεδομένη γιατί συνδυάζει το μικρότερο δυνατό μέγεθος του καταχωρητή καταστάσεων με αρχική τιμή στο όλα-0 και το ελάχιστο πλήθος των ψηφίων που αλλάζουν τιμή από κατάσταση σε κατάσταση
 - Μόνο ένα ψηφίο στις περισσότερες περιπτώσεις

6 καταστάσεις	One-Hot	Binary	τροπ. Gray
A	000001	000	000
B	000010	001	001
C	000100	010	011
D	001000	011	010
E	010000	100	110
F	100000	101	100 (111)

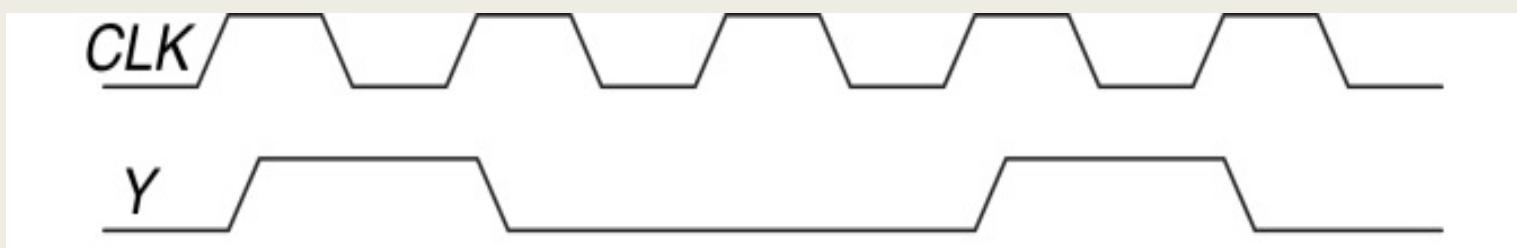
Κωδικοποίηση των κατασκευών

- Gray (Κατασκευή ακολουθίας για N bit)
- Για $N=0 \Rightarrow$ κώδικας {‘0’}
- Για $N=1 \Rightarrow$ κώδικας {‘0’, ‘1’}
- Για $N>1$, τοποθετούμε ‘0’ μπροστά από κάθε κωδική λέξη του κώδικα Gray των $N-1$ bit, και ακολούθως ‘1’ μπροστά από κάθε κωδική λέξη σε αντίστροφη σειρά του κώδικα Gray των $N-1$ bit

N=0	N=1	N=2	N=3
0	0	00	000
	1	01	001
		11	011
		10	010
			110
			111
			101
			100

Μετρητής διαίρεσης διά του 3*

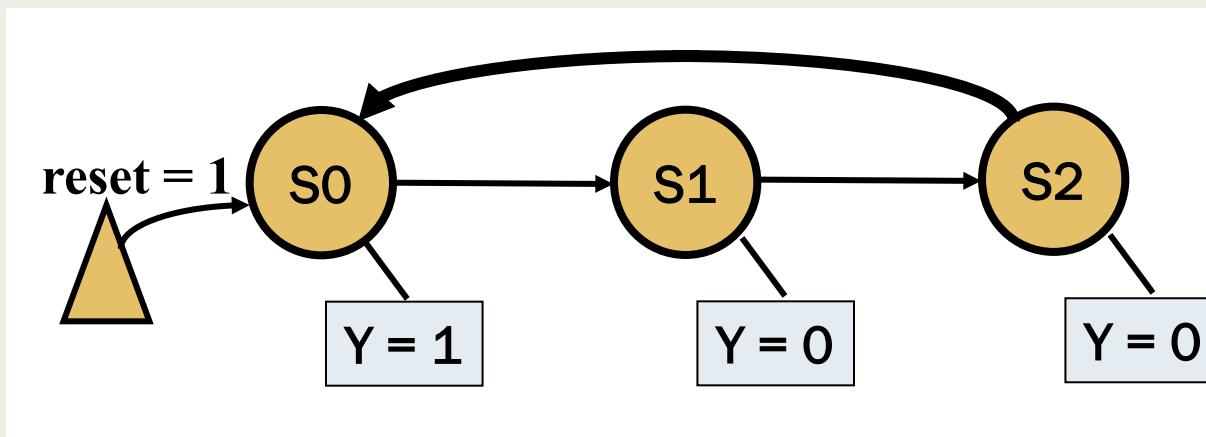
- Ένας μετρητής διαίρεσης διά του N (divide-by-N counter) διαθέτει μία έξοδο Y και καμία είσοδο
- Η έξοδος Y έχει την τιμή HIGH για έναν από κάθε N κύκλους του CLK
 - Η έξοδος Y διαιρεί τη συχνότητα του ρολογιού δια N
- Δίδεται το διάγραμμα χρονισμού για έναν μετρητή διαίρεσης διά του 3
- Μελετήστε την επίδραση που έχει στην πολυπλοκότητα της σχεδίασης η επιλογή της κωδικοποίησης των καταστάσεων
 - (α) δυαδική κωδικοποίηση και
 - (β) κωδικοποίηση μοναδικού σημαντικού (one-hot)



*Παράδειγμα 3.6

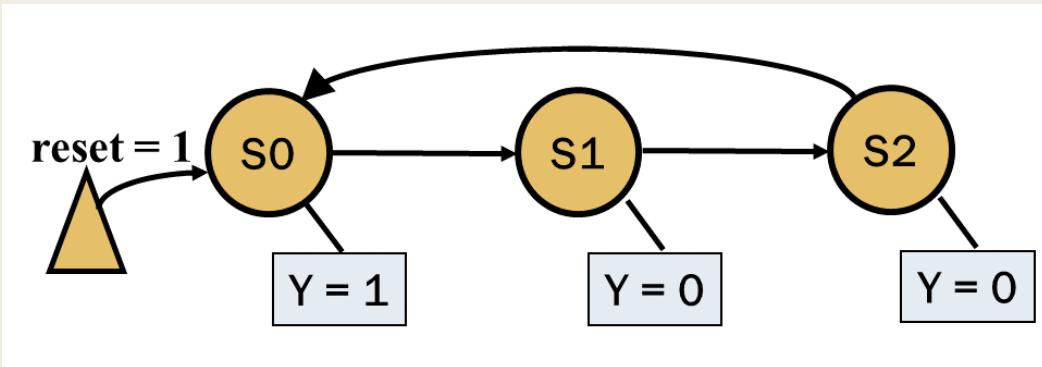
Μετρητής διαίρεσης διά του 3*

- **Βήμα 2: Σχεδιάζουμε το διάγραμμα μεταβολής κατάστασης**
 - Ο μετρητής έχει **3 καταστάσεις S_0 , S_1 και S_2**
 - Σχηματίζουν έναν κύκλο
 - Το σήμα **RESET** αρχικοποιεί τον μετρητή στην κατάσταση S_0
 - Η αρχικοποίηση γίνεται σύγχρονα ή ασύγχρονα του CLK με κατάλληλη επιλογή των D Flip-Flop



Μετρητής διαίρεσης διά του 3*

- Βήμα 3: Σχεδιάζουμε τον πίνακα μεταβολής κατάστασης



Current State	Next State
Q(t)	Q(t+1)
S0	S1
S1	S2
S2	S0

- Βήμα 4: Κωδικοποιούμε τις καταστάσεις

Κατάσταση	Δυαδική Κωδικοποίηση	One-hot Κωδικοποίηση
S0	00	001
S1	01	010
S2	10	100

Μετρητής διαίρεσης διά του 3*

- **Βήματα 4 & 6: Ενημερώνουμε τον πίνακα μεταβολής κατάστασης και ενσωματώνουμε τον πίνακα αληθείας για την έξοδο Y**

Current State	Next State
Q(t)	Q(t+1)
S0	S1
S1	S2
S2	S0

Κατάσταση	Δυαδική Κωδικοποίηση	One-hot Κωδικοποίηση
S0	00	001
S1	01	010
S2	10	100

Δυαδική Κωδικοποίηση		
Current State	Next State	Output
Q1 Q0	Q1* Q0*	Y
0 0	0 1	1
0 1	1 0	0
1 0	0 0	0

One-hot Κωδικοποίηση		
Current State	Next State	Output
Q2 Q1 Q0	Q2* Q1* Q0*	Y
0 0 1	0 1 0	1
0 1 0	1 0 0	0
1 0 0	0 0 1	0

Μετρητής διαίρεσης διά του 3*

- **Βήματα 5 & 7:** Βρίσκουμε τις εξισώσεις Boole των μεταβλητών των επόμενων καταστάσεων (συμβολίζουμε με Q^*) και της εξόδου

Δυαδική Κωδικοποίηση		
Current State	Next State	Output
Q1 Q0	Q1* Q0*	Y
0 0	0 1	1
0 1	1 0	0
1 0	0 0	0

One-hot Κωδικοποίηση		
Current State	Next State	Output
Q2 Q1 Q0	Q2* Q1* Q0*	Y
0 0 1	0 1 0	1
0 1 0	1 0 0	0
1 0 0	0 0 1	0

$$Q1^* = \overline{Q1} Q0$$

$$Q0^* = \overline{Q1} \overline{Q0}$$

$$Y = \overline{Q1} \overline{Q0}$$

$$Q2^* = Q1$$

$$Q1^* = Q0$$

$$Q0^* = Q2$$

$$Y = Q0$$

Ελαχιστοποιεί τις εξισώσεις Boole, αλλά αυξάνει το μέγεθος του καταχωρητή

Μετρητής διαίρεσης διά του 3*

■ Βήμα 8: Σχεδιάζουμε το σχηματικό διάγραμμα

Δυαδική
κωδικοποίηση
καταστάσεων

$$Q1^* = \overline{Q1} Q0$$

$$Q0^* = \overline{Q1} \overline{Q0}$$

$$Y = \overline{Q1} \overline{Q0}$$

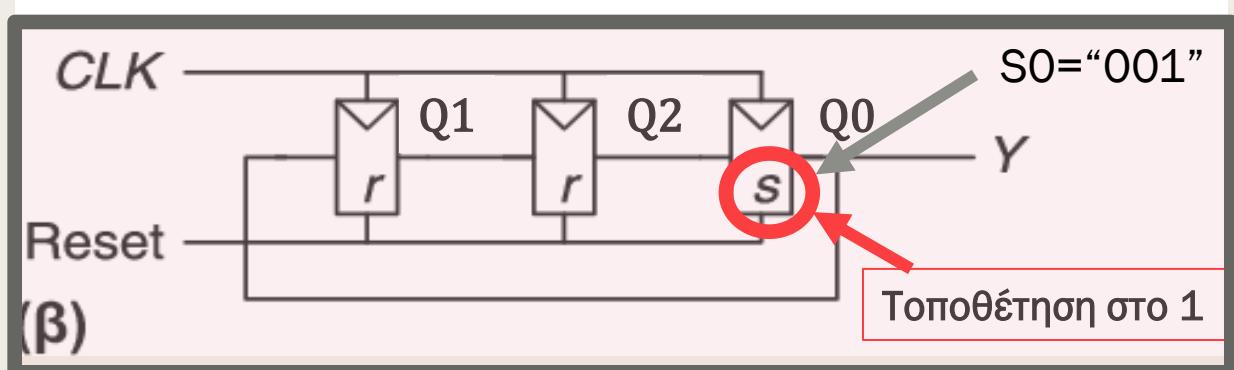
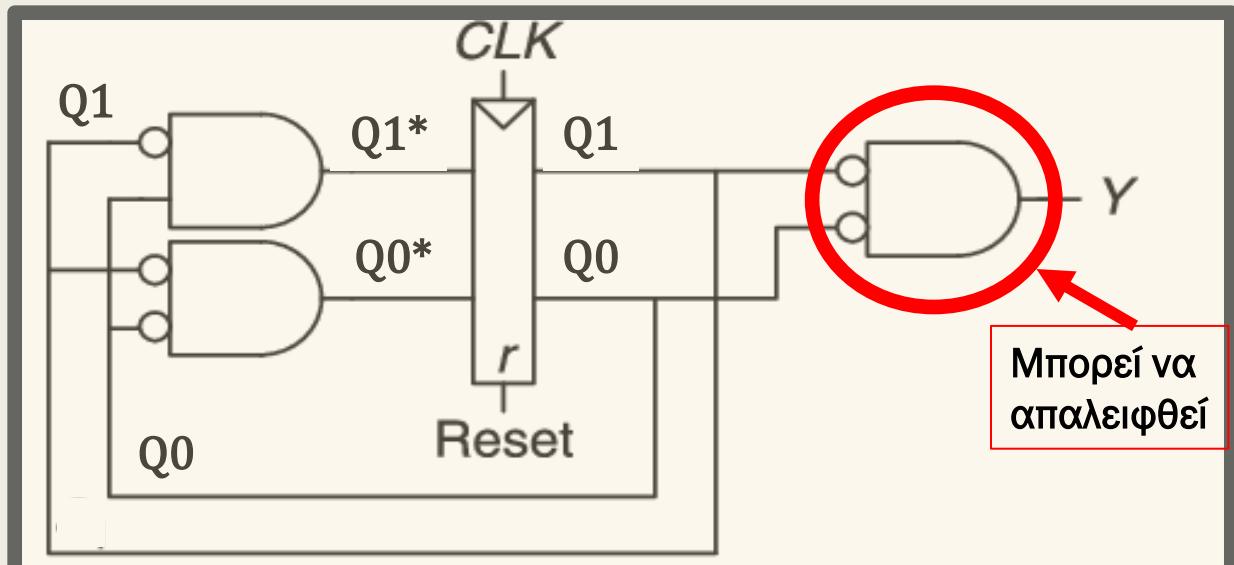
One-hot
κωδικοποίηση
καταστάσεων

$$Q2^* = Q1$$

$$Q1^* = Q0$$

$$Q0^* = Q2$$

$$Y = Q0$$



Επιλεγμένες ασκήσεις

■ Άσκηση 3.27

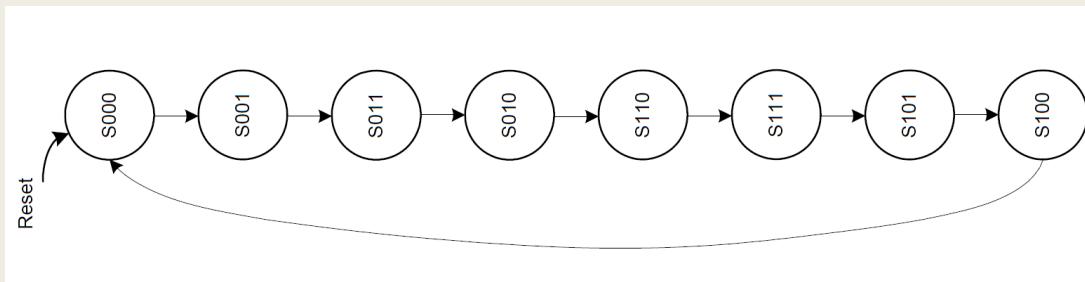
Σχεδιάστε έναν μετρητή FSM υπολοίπου διαίρεσης ως προς το 8 (modulo 8 counter), των 3 bit, ο οποίος χρησιμοποιεί κώδικα Gray

- Ο μετρητής έχει τρεις εξόδους και καμία είσοδο
- Ο μετρητής αρχικοποιείται με το σήμα *RESET* στην κατάσταση 000.
- Σε κάθε ανερχόμενη ακμή του *CLK*, η έξοδος προχωράει στον επόμενο αριθμό του κώδικα Gray
- Αφού φτάσει στον αριθμό 7 (κατάσταση 100 στον κώδικα Gray), πρέπει να ξεκινάει από την αρχή με τον αριθμό 0 (κατάσταση 000 στον κώδικα Gray).
- Να χρησιμοποιήσετε κωδικοποίηση Gray, ώστε να ταυτίζονται οι έξοδοι με τις καταστάσεις

Αριθμός	Κώδικας Gray		
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Μετρητής Gray των 3 bit

- **Βήμα 1: Προσδιορίζουμε τις εισόδους, τις εξόδους, και τις καταστάσεις**
 - Ο μετρητής δεν διαθέτει άλλη είσοδο πέραν του σήματος Reset
 - Οι καταστάσεις ταυτίζονται με τις εξόδους (τύπου Moore)
 - Έχει 8 καταστάσεις με κωδικοποίηση Gray
- **Βήμα 2: Σχεδιάζουμε το διάγραμμα μεταβολής κατάστασης**



Current State	Next State
S000	S001
S001	S011
S011	S010
S010	S110
S110	S111
S111	S101
S101	S100
S100	S000

- **Βήμα 3: Σχεδιάζουμε τον πίνακα μεταβολής κατάστασης**

Μετρητής Gray των 3 bit

- Βήμα 4: Κωδικοποιούμε τις καταστάσεις και ενημερώνουμε τον πίνακα μεταβολής κατάστασης
 - Διαμορφώνουμε κατάλληλα τις γραμμές του Πίνακα Αλήθειας

Cur. State	Next State
S000	S001
S001	S011
S011	S010
S010	S110
S110	S111
S111	S101
S101	S100
S100	S000

Cur. State Q2 Q1 Q0	Next State Q2* Q1* Q0*
0 0 0	0 0 1
0 0 1	0 1 1
0 1 1	0 1 0
0 1 0	1 1 0
1 1 0	1 1 1
1 1 1	1 0 1
1 0 1	1 0 0
1 0 0	0 0 0

Cur. State Q2 Q1 Q0	Next State Q2* Q1* Q0*
0 0 0	0 0 1
0 0 1	0 1 1
0 1 0	1 1 0
0 1 1	0 1 0
1 0 0	0 0 0
1 0 1	1 0 0
1 1 0	1 1 1
1 1 1	1 0 1

Αναδιάταξη του προηγούμενου πίνακα ως binary code

Μετρητής Gray των 3 bit

- Βήμα 5: Ελαχιστοποιούμε με K-map και βρίσκουμε τις εξισώσεις Boolean των μεταβλητών των επόμενων καταστάσεων (εξόδων)

Cur. State Q2 Q1 Q0	Next State Q2* Q1* Q0*
0 0 0	0 0 1
0 0 1	0 1 1
0 1 0	1 1 0
0 1 1	0 1 0
1 0 0	0 0 0
1 0 1	1 0 0
1 1 0	1 1 1
1 1 1	1 0 1

Q_0

Q_2	Q_1	00	01	11	10
0	0	1	1		0
1	0	0		1	1

$$Q_2^* = Q_1 \overline{Q_0} + Q_2 Q_0$$

Q_0

Q_2	Q_1	00	01	11	10
0	0	1	1		0
1	1	1	0	0	0

$$Q_1^* = Q_1 \overline{Q_0} + \overline{Q_2} Q_0$$

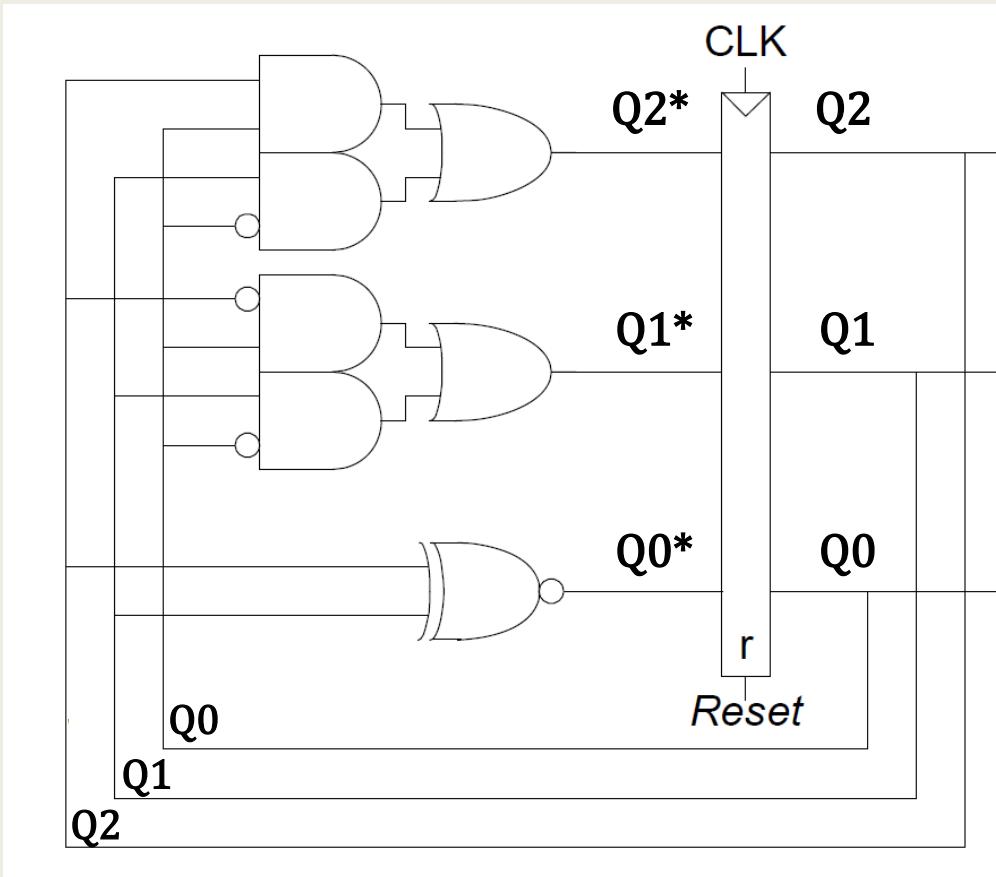
Q_0

Q_2	Q_1	00	01	11	10
0	1	0		1	0
1	1	0	1	1	0

$$Q_0^* = \overline{Q_2} \overline{Q_1} + Q_2 Q_1$$

Μετρητής Gray των 3 bit

- Βήμα 8: Σχεδιάζουμε το σχηματικό διάγραμμα



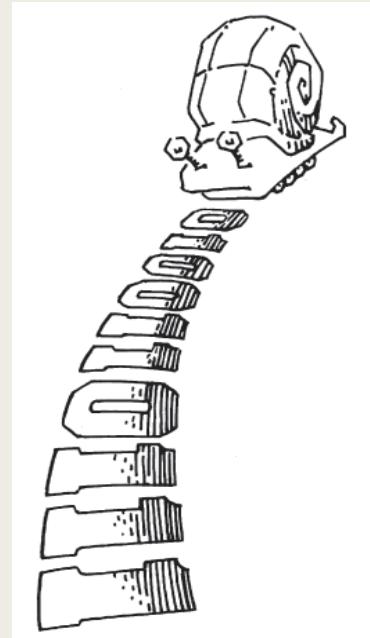
$$Q2^* = Q1 \overline{Q_0} + Q2 Q0$$

$$Q1^* = Q1 \overline{Q_0} + \overline{Q_2} Q0$$

$$Q0^* = \overline{Q_2} \overline{Q_1} + Q2 Q1$$

Μηχανές FSM: Moore vs Mealy*

- Έστω ότι έχουμε ένα σαλιγκάρι-ρομπότ ως κατοικίδιο, το οποίο διαθέτει μια μηχανή FSM για εγκέφαλο
 - Το σαλιγκάρι έρπει κατά μήκος μιας χάρτινης ταινίας που περιέχει μια ακολουθία από 0 και 1
 - Σε κάθε κύκλο του ρολογιού, το σαλιγκάρι έρπει έως το επόμενο bit της ακολουθίας
 - Το σαλιγκάρι χαμογελάει όταν τα δύο τελευταία **διαδοχικά bit** πάνω από τα οποία έχει περάσει είναι **01**
- Σχεδιάστε τη μηχανή FSM έτσι ώστε να υπολογίζει πότε το σαλιγκάρι πρέπει να χαμογελάει
 - Η μηχανή FSM του σαλιγκαριού-ρομπότ είναι ένας **ανιχνευτής ακολουθίας 2 διαδοχικών bit** που μεταδίδονται σειριακά στην είσοδο X
 - Το σαλιγκάρι να χαμογελάει όταν η έξοδος Y γίνεται 1
- Συγκρίνετε τις σχεδιάσεις των μηχανών FSM **Τύπων Moore και Mealy**
- Σχεδιάστε ένα **διάγραμμα χρονισμού** για κάθε τύπο της μηχανής FSM
 - στο οποίο θα φαίνονται η είσοδος, οι καταστάσεις, και η έξοδος καθώς το σαλιγκάρι έρπει κατά μήκος της ακολουθίας **0100110111**



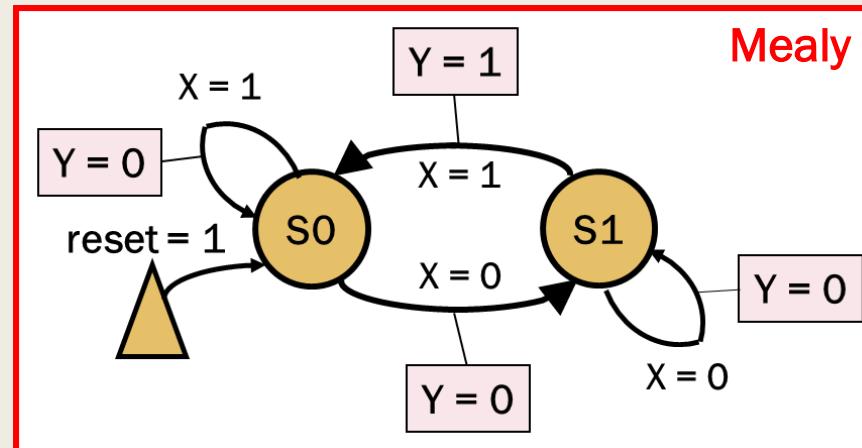
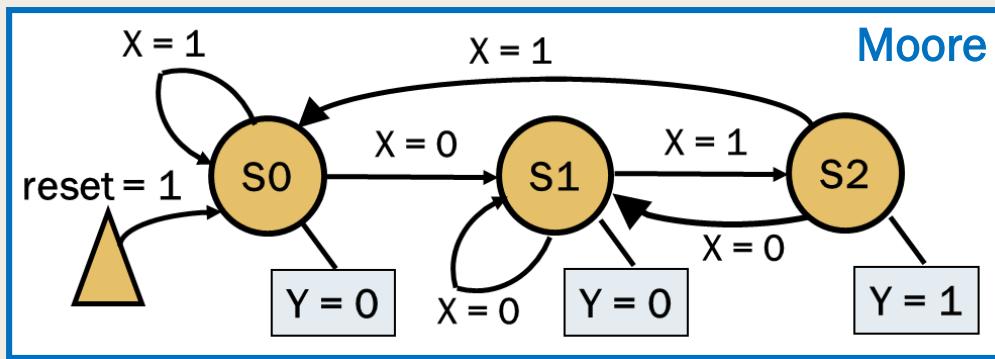
Ανιχνευτής ακολουθίας 2 διαδοχικών bit

- **Βήμα 1: Προσδιορίζουμε τις εισόδους, τις εξόδους, και τις καταστάσεις**
 - Ο ανιχνευτής έχει μία σειριακή είσοδο X και μία έξοδο Y
 - $Y = 1$, όταν τα δύο τελευταία διαδοχικά bit της εισόδου X είναι 01
 - *Καταστάσεις μηχανής Túpo Moore:*
 - S_0 = αρχική κατάσταση, δεν έχει ανιχνευθεί κανένα ψηφίο, $Y = 0$
 - S_1 = έχει ανιχνευθεί στην είσοδο X ένα bit 0, $Y = 0$
 - S_2 = έχουν ανιχνευθεί στην είσοδο X δύο διαδοχικά bit 01, $Y = 1$
 - *Καταστάσεις μηχανής Túpo Mealy:*
 - S_0 = αρχική κατάσταση δεν έχει ανιχνευθεί κανένα ψηφίο
 - εάν $X = 1$, τότε $Y = 0$ και παραμένει στην S_0
 - εάν $X = 0$, τότε $Y = 0$ και πηγαίνει στην S_1 (ανίχνευση 0)
 - S_1 = έχει ανιχνευθεί στην είσοδο X ένα bit 0
 - εάν $X = 1$, τότε $Y = 1$ και πηγαίνει στην S_0 (ανίχνευση 01)
 - εάν $X = 0$, τότε $Y = 0$ και παραμένει στην S_1 (ανίχνευση 0)
 - Δεν χρειάζεται η κατάσταση S_2 για τον προσδιορισμό του $Y = 1$
 - Η έξοδος $Y = 1$ προσδιορίζεται από την είσοδο $X = 1$, όταν βρίσκεται στην κατάσταση S_1

Ανιχνευτής ακολουθίας 2 διαδοχικών bit

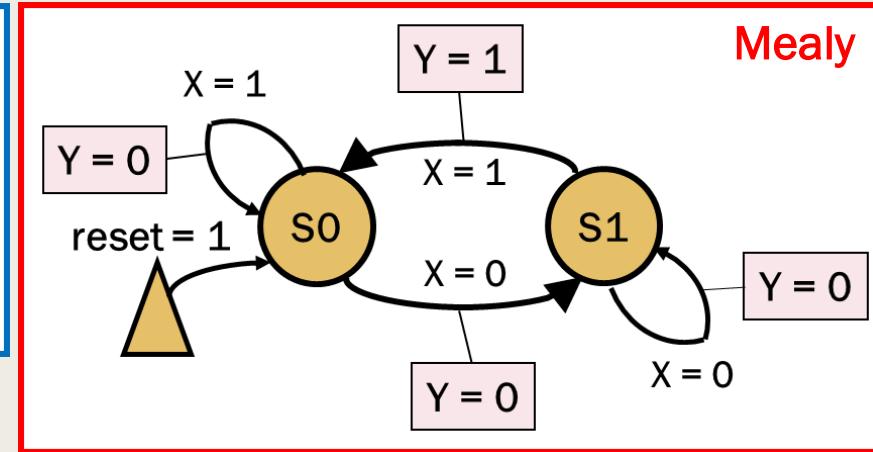
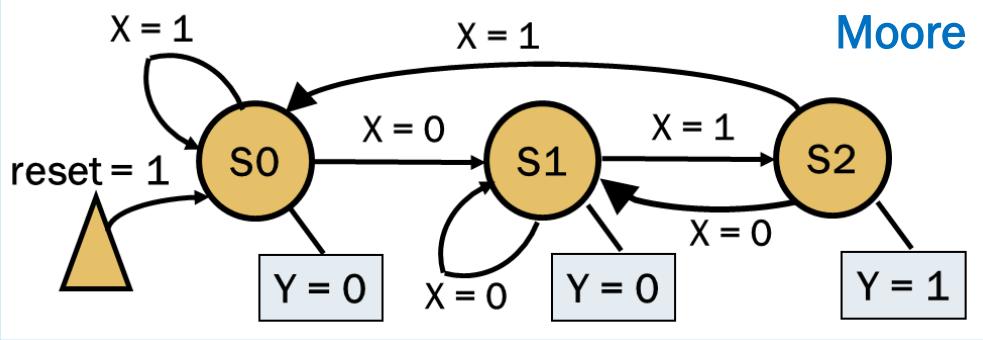
■ Βήμα 2: Σχεδιάζουμε τα διαγράμματα μεταβολής κατάστασης

- Ο ανιχνευτής τύπου Moore έχει **3 καταστάσεις S_0 , S_1 και S_2**
 - Ο ανιχνευτής τύπου Mealy έχει **2 καταστάσεις S_0 και S_1**
 - Το σήμα **RESET** αρχικοποιεί τον ανιχνευτή στην κατάσταση **S_0**
- Η αρχικοποίηση του ανιχνευτή γίνεται σύγχρονα ή ασύγχρονα του CLK με κατάλληλη επιλογή των D Flip-Flop



Ανιχνευτής ακολουθίας 2 διαδοχικών bit

- Βήμα 3: Σχεδιάζουμε τους πίνακες μεταβολής κατάστασης

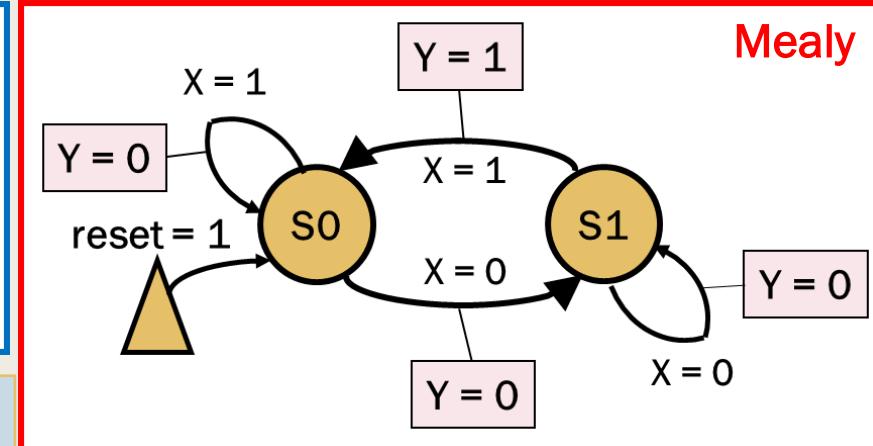
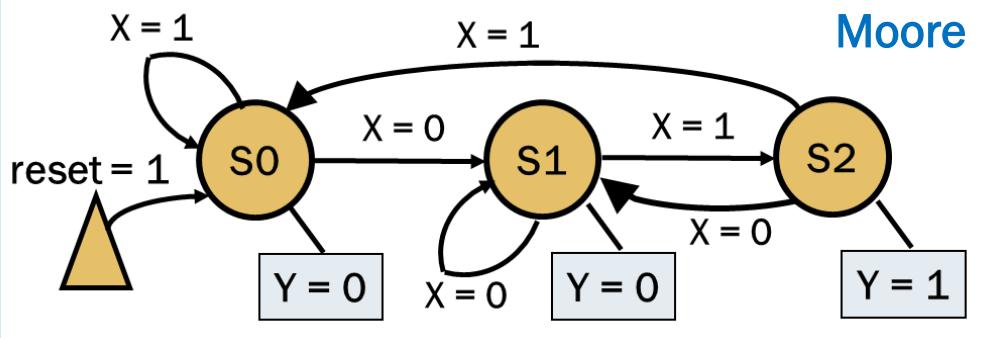


Current State	In put	Next State
$Q(t)$	X	$Q(t+1)$
S0	0	S1
S0	1	S0
S1	0	S1
S1	1	S2
S2	0	S1
S2	1	S0

Current State	In put	Next State
$Q(t)$	X	$Q(t+1)$
S0	0	S1
S0	1	S0
S1	0	S1
S1	1	S0

Ανιχνευτής ακολουθίας 2 διαδοχικών bit

- Βήμα 4: Κωδικοποιούμε τις καταστάσεις και ενημερώνουμε τους πίνακες μεταβολής κατάστασης



Current State	In put	Next State	Κατά σταση	Κωδικο ποίηση
Q1 Q0	X	Q1* Q0*	S0	00
0 0	0	0 1	S1	01
0 0	1	0 0	S2	10
0 1	0	0 1		
0 1	1	1 0		
1 0	0	0 1		
1 0	1	0 0		

Current State	In put	Next State	Κατ.	Κωδ.
Q	X	Q*	S0	0
0	0	1	S1	1
0	1	0		
1	0	1		
1	1	0		

Ανιχνευτής ακολουθίας 2 διαδοχικών bit

- Βήμα 5: Ελαχιστοποιούμε με K-μαρ και βρίσκουμε τις εξισώσεις Boolean των μεταβλητών των επόμενων καταστάσεων

Moore

Current State	In put	Next State
Q1 Q0	X	Q1* Q0*
0 0	0	0 1
0 0	1	0 0
0 1	0	0 1
0 1	1	1 0
1 0	0	0 1
1 0	1	0 0
1 1	0	XX
1 1	1	XX

$$Q1^* = Q0 X$$

$$Q0^* = \bar{X}$$

Mealy

Current State	In put	Next State
Q	X	Q*
0	0	1
0	1	0
1	0	1
1	1	0

$$Q^* = \bar{X}$$

Ανιχνευτής ακολουθίας 2 διαδοχικών bit

- **Βήμα 6: Σχεδιάζουμε τους πίνακες αληθείας για τις εξόδους**
 - *Tύπου Moore*: παρούσες καταστάσεις, έξοδοι
 - *Tύπου Mealy*: παρούσες καταστάσεις, είσοδοι, έξοδοι
- **Βήμα 7: Βρίσκουμε τις εξισώσεις Boole των εξόδων**

Moore

Current State	Output
Q1 Q0	Y
0 0	0
0 1	0
1 0	1
1 1	X

Mealy

Current State	Input	Output
Q	X	Y
0	0	0
0	1	0
1	0	0
1	1	1

$$Q1^* = Q0 X$$

$$Q0^* = \bar{X}$$

$$Y = Q1$$

$$Y = Q X$$

$$Q^* = \bar{X}$$

Ανιχνευτής ακολουθίας 2 διαδοχικών bit

- Βήμα 8: Σχεδιάζουμε το σχηματικό διάγραμμα

Moore

$$Q1^* = Q0 \ X$$

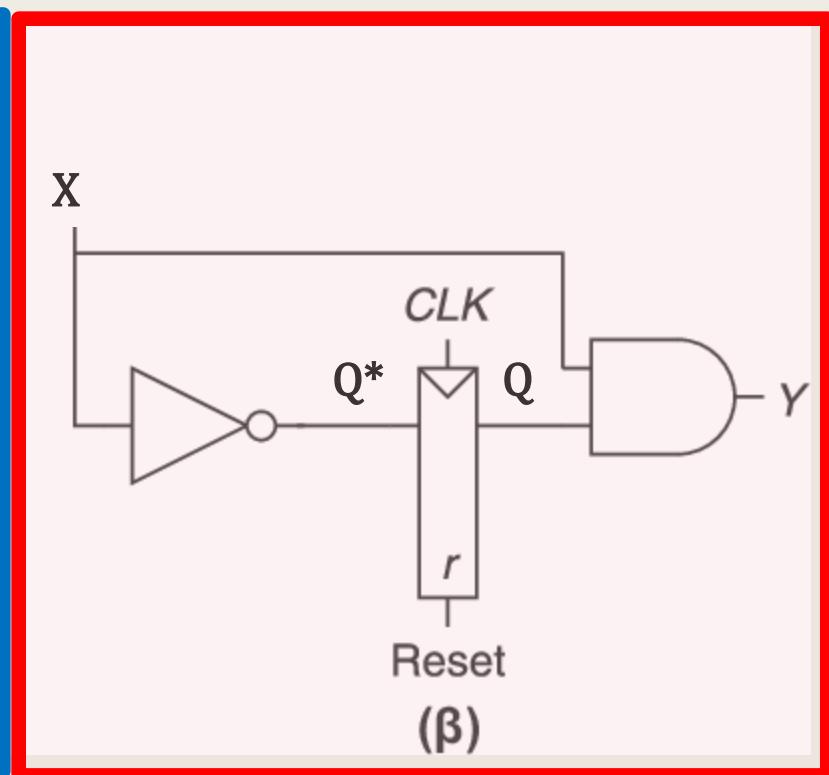
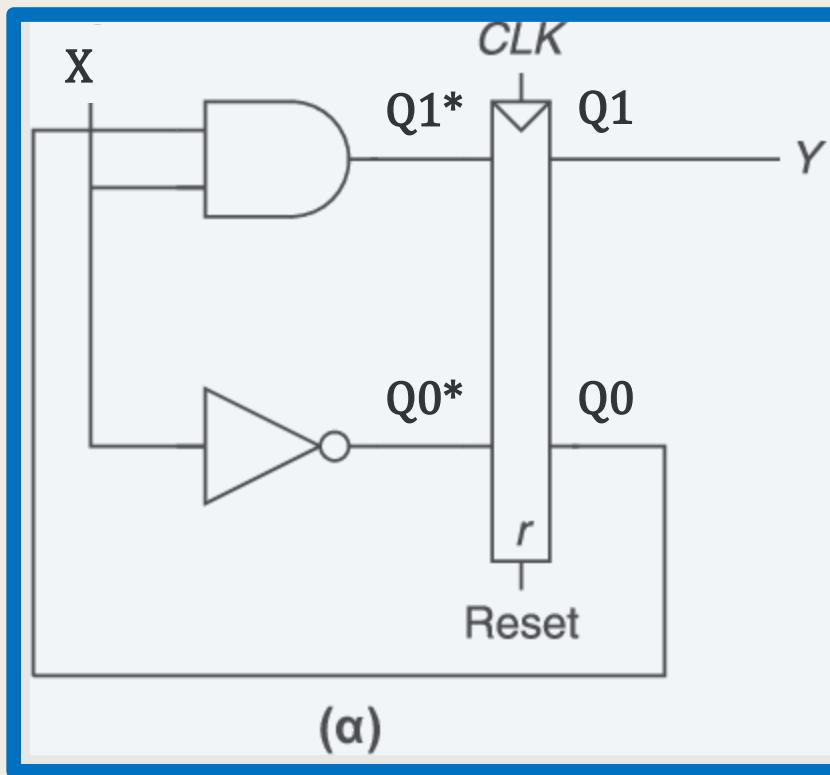
$$Q0^* = \bar{X}$$

$$Y = Q1$$

Mealy

$$Y = Q \ X$$

$$Q^* = \bar{X}$$



Ανιχνευτής ακολουθίας 2 διαδοχικών bit

■ Διάγραμμα χρονισμού

Moore

$$Q1^* = Q0 X$$

$$Q0^* = \bar{X}$$

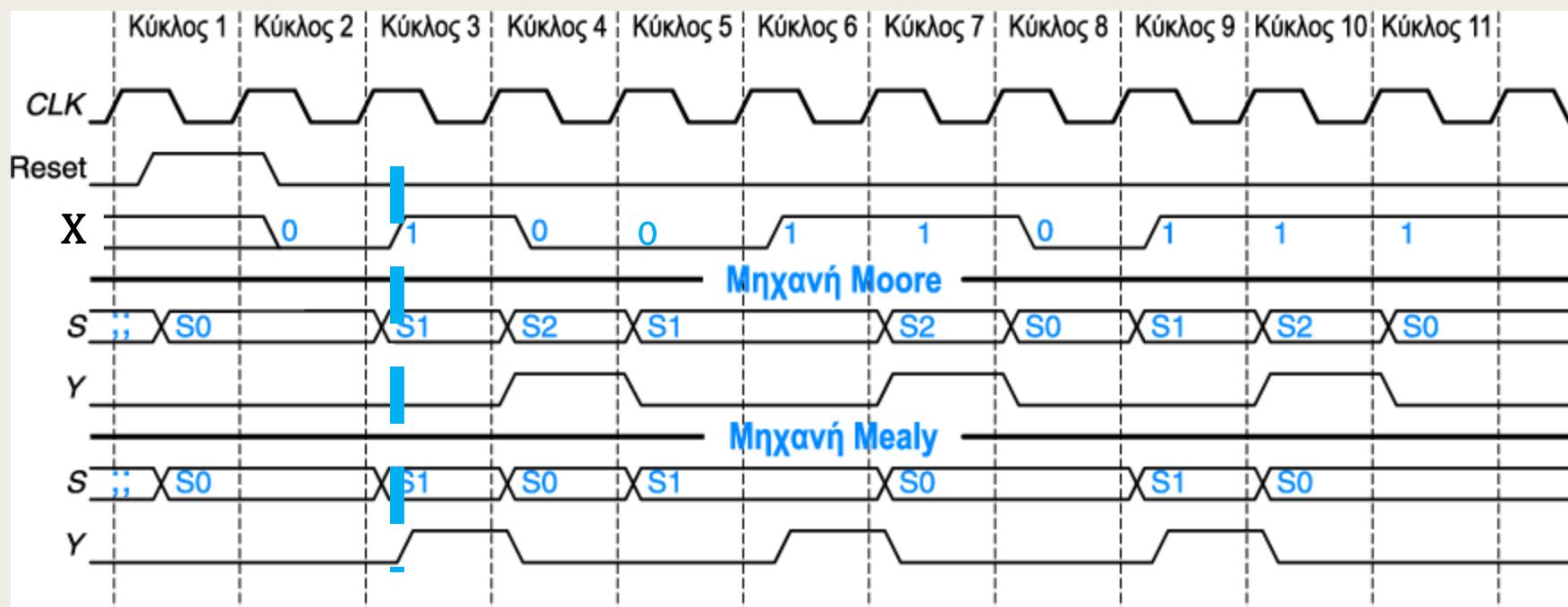
$$Y = Q1$$

Mealy

$$Y = Q X$$

$$Q^* = \bar{X}$$

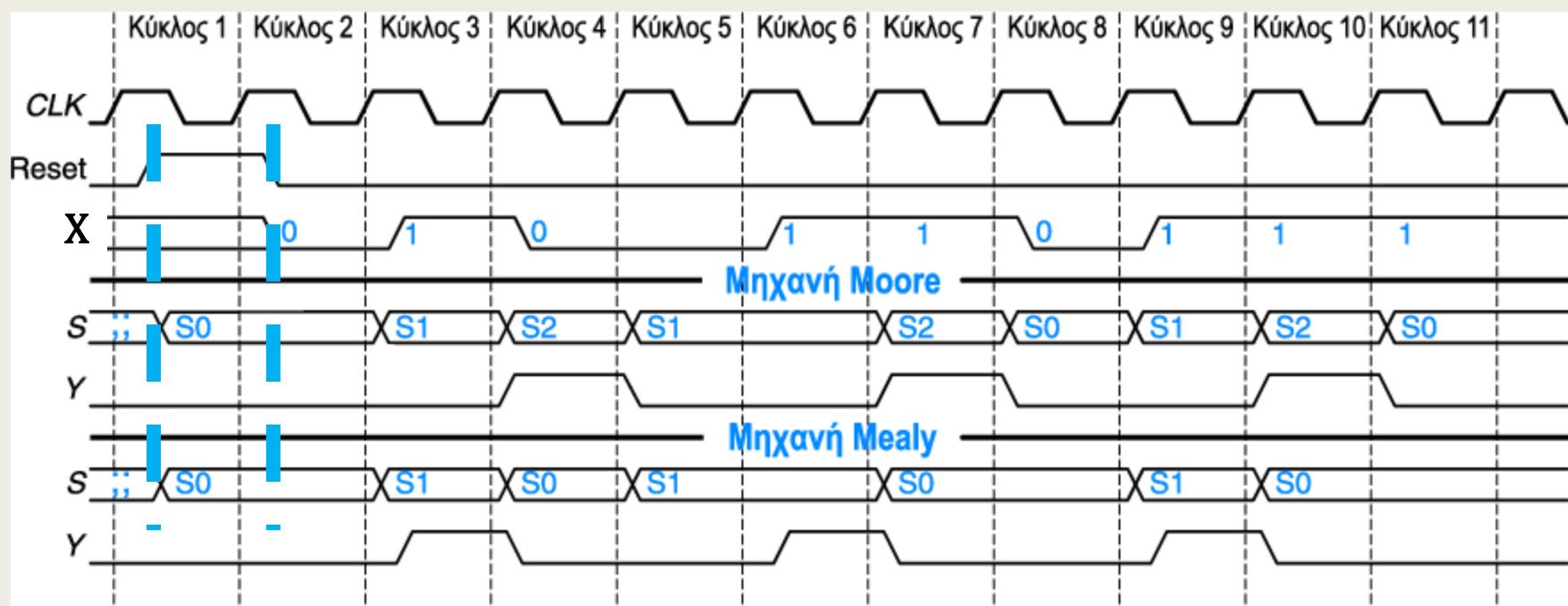
Αν η έξοδος Y της μηχανής τύπου Mealy καθυστερούσε μέσω ενός D Flip-Flop, τότε θα ταίριαζε με την έξοδο Y της μηχανής τύπου Moore



Ανιχνευτής ακολουθίας 2 διαδοχικών bit

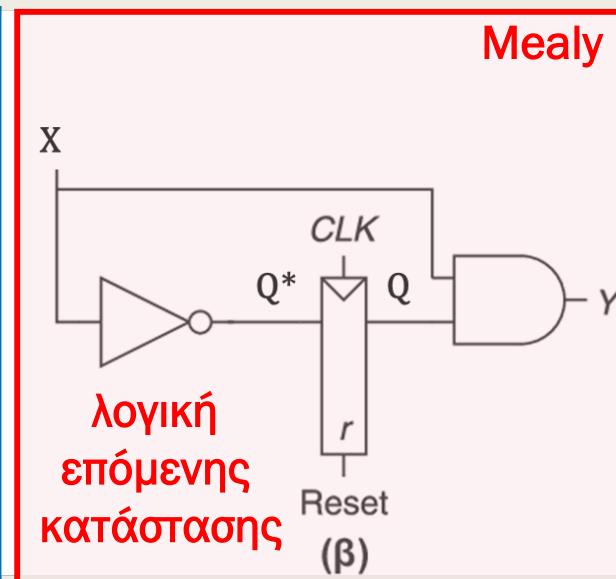
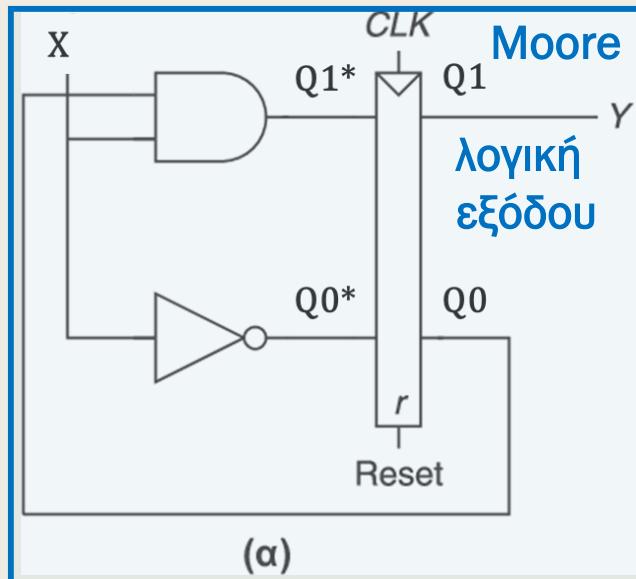
■ Διάγραμμα χρονισμού

- Όταν ενεργοποιηθεί για πρώτη φορά η μηχανή *FSM*, η κατάστασή της είναι **απροσδιόριστη** (;;)
- Με την ενεργοποίηση του **ασύγχρονου σήματος Reset** η μηχανή τίθεται στην αρχική κατάσταση *S0* (πριν την ανερχόμενη ακμή του *CLK*)
- Με την απενεργοποίηση του ασύγχρονου σήματος *Reset* η είσοδος *X* παίρνει την πρώτη τιμή της (*X=0* στον κύκλο 2)
- Οι δύο μηχανές ακολουθούν **διαφορετική ακολουθία καταστάσεων**
- Η άνοδος της εξόδου *Y* της μηχανής *Mealy* συμβαίνει **έναν κύκλο νωρίτερα**, επειδή η έξοδος αποκρίνεται στην είσοδο αντί να περιμένει για τη μεταβολή της κατάστασης



Μηχανές FSM: Moore vs Mealy*

- Η μηχανή πεπερασμένων καταστάσεων τύπου Mealy είναι πιο γενική από τη μηχανή τύπου Moore
 - Η **έξοδος εμφανίζεται πιο γρήγορα** (σε λιγότερους κύκλους)
- Οι μηχανές πεπερασμένων καταστάσεων **Τύπου Moore**
 - έχουν **περισσότερες καταστάσεις**
 - συνήθως πλεονεκτούν σε ταχύτητα και μέγεθος της **λογικής εξόδου**
- Οι μηχανές πεπερασμένων καταστάσεων **Τύπου Mealy**
 - έχουν **λιγότερες καταστάσεις**
 - συνήθως πλεονεκτούν σε ταχύτητα και μέγεθος της **λογικής επόμενης κατάστασης**



Ανιχνευτής ακολουθίας 4 διαδοχικών bit

- Σχεδιάστε το διάγραμμα μεταβολής κατάστασης για τον ανιχνευτή της ακολουθίας **0101**
 - Στηρίζεται σε μηχανή τύπου *Moore*

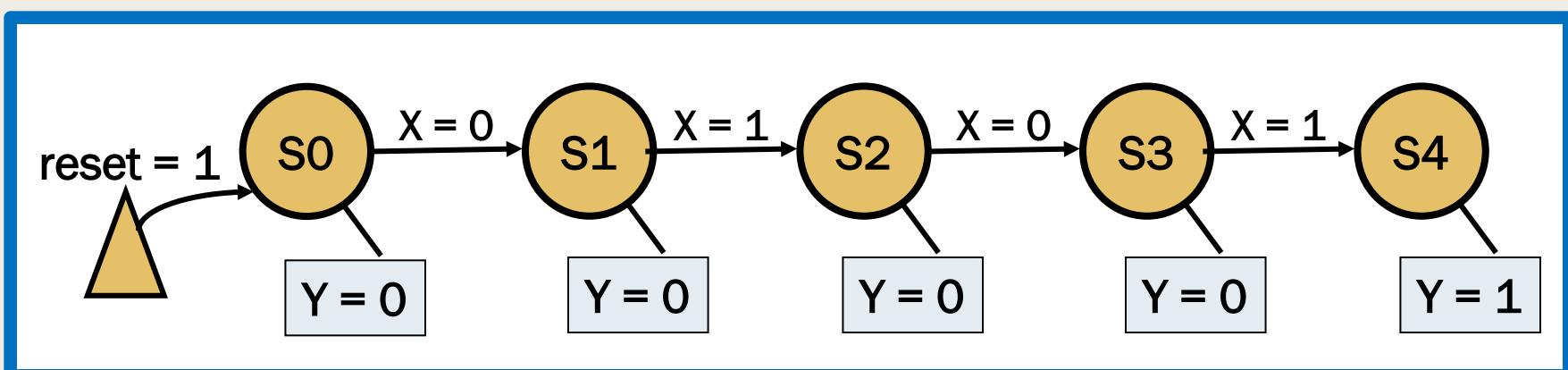
Ανιχνευτής ακολουθίας 4 διαδοχικών bit

■ Βήμα 1: Προσδιορίζουμε τις εισόδους, τις εξόδους, και τις καταστάσεις

- Ο ανιχνευτής έχει μία σειριακή είσοδο X και μία έξοδο Y
 - $Y = 1$, όταν τα 4 τελευταία διαδοχικά bit της εισόδου X είναι **0101**
- Καταστάσεις μηχανής **Τύπου Moore**:
 - S_0 = αρχική κατάσταση, δεν έχει ανιχνευθεί κανένα ψηφίο, $Y = 0$
 - S_1 = έχει ανιχνευθεί στην είσοδο X ένα bit 0, $Y = 0$
 - S_2 = έχουν ανιχνευθεί στην είσοδο X δύο διαδοχικά bit 01, $Y = 0$
 - S_3 = έχουν ανιχνευθεί στην είσοδο X τρία διαδοχικά bit 010, $Y = 0$
 - S_4 = έχουν ανιχνευθεί στην είσοδο X τέσσερα διαδοχικά bit 0101, $Y = 1$

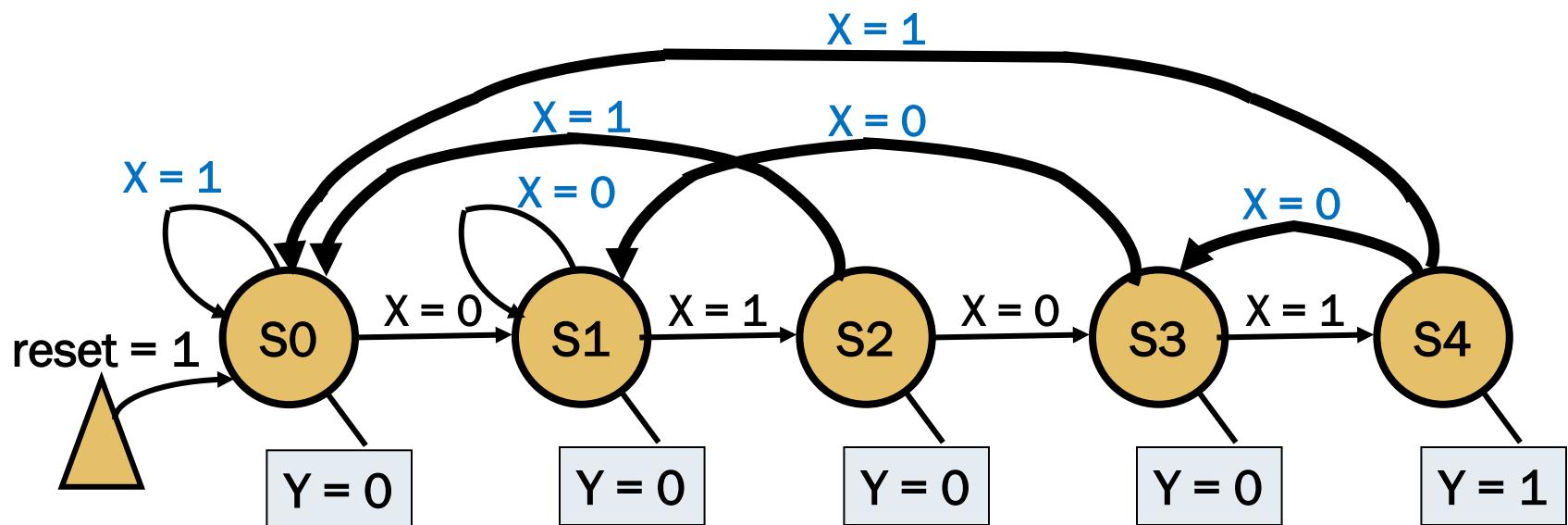
Ανιχνευτής ακολουθίας 4 διαδοχικών bit

- **Βήμα 2: Σχεδιάζουμε το διάγραμμα μεταβολής κατάστασης**
 - Ο ανιχνευτής τύπου Moore έχει **5 καταστάσεις S_0, S_1, S_2, S_3 και S_4**
 - Το σήμα **RESET** αρχικοποιεί τον ανιχνευτή στην κατάσταση **S_0**
 - Η αρχικοποίηση του ανιχνευτή γίνεται σύγχρονα ή ασύγχρονα του CLK με κατάλληλη επιλογή των D Flip-Flop
 - Η σχεδίαση του διαγράμματος γίνεται σε δύο φάσεις:
 - Στην πρώτη φάση σχεδιάζουμε τις τρέχουσες καταστάσεις (κύκλους) και τα βέλη με τη συνθήκη εισόδου ($X = 0$ ή $X = 1$) που προσδιορίζει τη μετάβαση από την τρέχουσα κατάσταση στην επόμενη κατάσταση, σύμφωνα με την ακολουθία των 4 bit (0101)



Ανιχνευτής ακολουθίας 4 διαδοχικών bit

- **Βήμα 2: Σχεδιάζουμε το διάγραμμα μεταβολής κατάστασης**
 - Ο ανιχνευτής τύπου Moore έχει **5 καταστάσεις S_0, S_1, S_2, S_3 και S_4**
 - Η σχεδίαση του διαγράμματος γίνεται σε δύο φάσεις:
 - Στη δεύτερη φάση σχεδιάζουμε τα βέλη με την **συμπληρωματική συνθήκη εισόδου** ($X = 1$ ή $X = 0$, αντίστοιχα)



Επιλεγμένες ασκήσεις

■ **Άσκηση 3.25**

Το σαλιγκάρι-ρομπότ έχει μία κόρη που διαθέτει για εγκέφαλο μια μηχανή FSM που στηρίζεται σε μηχανή τύπου Moore

- Το σαλιγκάρι-κόρη χαμογελάει όταν έρπει πάνω από την ακολουθία **1101** ή την ακολουθία **1110**.
 - Σχεδιάστε το διάγραμμα μεταβολής κατάστασης γι' αυτό το χαρούμενο σαλιγκάρι

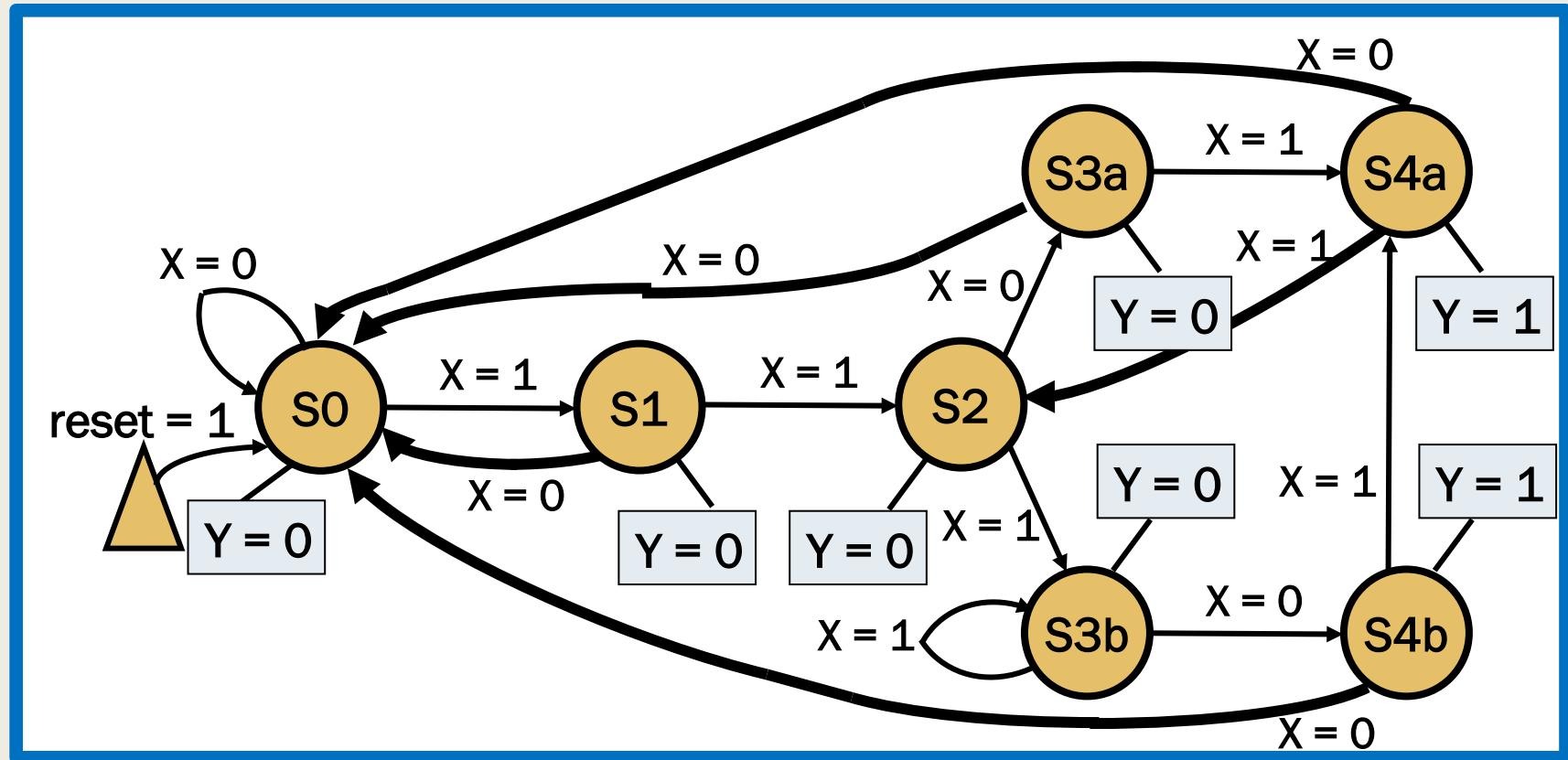
Άσκηση 3.25: Λύση (FSM τύπου Moore)

■ Βήμα 1: Προσδιορίζουμε τις εισόδους, τις εξόδους, και τις καταστάσεις

- Ο ανιχνευτής έχει μία σειριακή είσοδο X και μία έξοδο Y
 - $Y = 1$, όταν τα 4 τελευταία διαδοχικά bit της εισόδου X είναι **1101** ή **1110**
- Καταστάσεις μηχανής **τύπου Moore**:
 - S_0 = αρχική κατάσταση, δεν έχει ανιχνευθεί κανένα ψηφίο, $Y = 0$
 - S_1 = έχει ανιχνευθεί στην είσοδο X ένα bit 1, $Y = 0$
 - S_2 = έχουν ανιχνευθεί στην είσοδο X δύο διαδοχικά bit 11, $Y = 0$
 - S_{3a} = έχουν ανιχνευθεί στην είσοδο X τρία διαδοχικά bit 110, $Y = 0$
 - S_{3b} = έχουν ανιχνευθεί στην είσοδο X τρία διαδοχικά bit 111, $Y = 0$
 - S_{4a} = έχουν ανιχνευθεί στην είσοδο X τέσσερα διαδοχικά bit 1101, $Y = 1$
 - S_{4b} = έχουν ανιχνευθεί στην είσοδο X τέσσερα διαδοχικά bit 1110, $Y = 1$

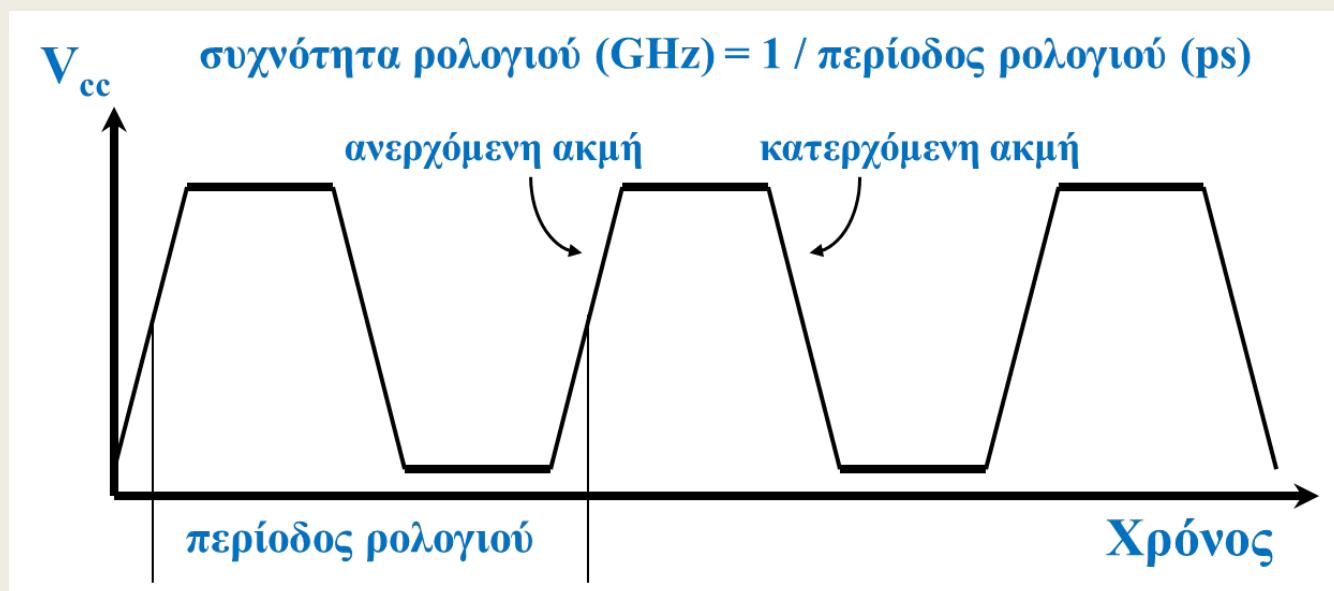
Άσκηση 3.25: Λύση (FSM τύπου Moore)

- **Βήμα 2: Σχεδιάζουμε το διάγραμμα μεταβολής κατάστασης**
 - Ο ανιχνευτής τύπου Moore έχει **7 καταστάσεις $S_0, S_1, S_2, S_{3a}, S_{3b}, S_{4a}$ και S_{4b}**
 - Το σήμα **RESET** αρχικοποιεί τον ανιχνευτή στην κατάσταση **S_0**
 - Η αρχικοποίηση του ανιχνευτή γίνεται σύγχρονα ή ασύγχρονα του CLK με κατάλληλη επιλογή των D Flip-Flop



Χρονισμός ακολουθιακής λογικής

- Το σήμα του ρολογιού CLK είναι ένας τετραγωνικός παλμός
 - Ένα D Flip-Flop αντιγράφει την είσοδο D στην έξοδο Q κατά την ανερχόμενη ακμή του CLK
 - Η συγκεκριμένη διαδικασία ονομάζεται δειγματοληψία (sampling) του D με βάση την ανερχόμενη ακμή του CLK
 - Για να επιτευχθεί η δειγματοληψία η είσοδος D πρέπει να είναι σταθερή πριν και μετά την ανερχόμενη ακμή του CLK (δυναμική πειθαρχία)
 - Άλλιώς μπορεί να εμφανισθεί μετασταθερότητα

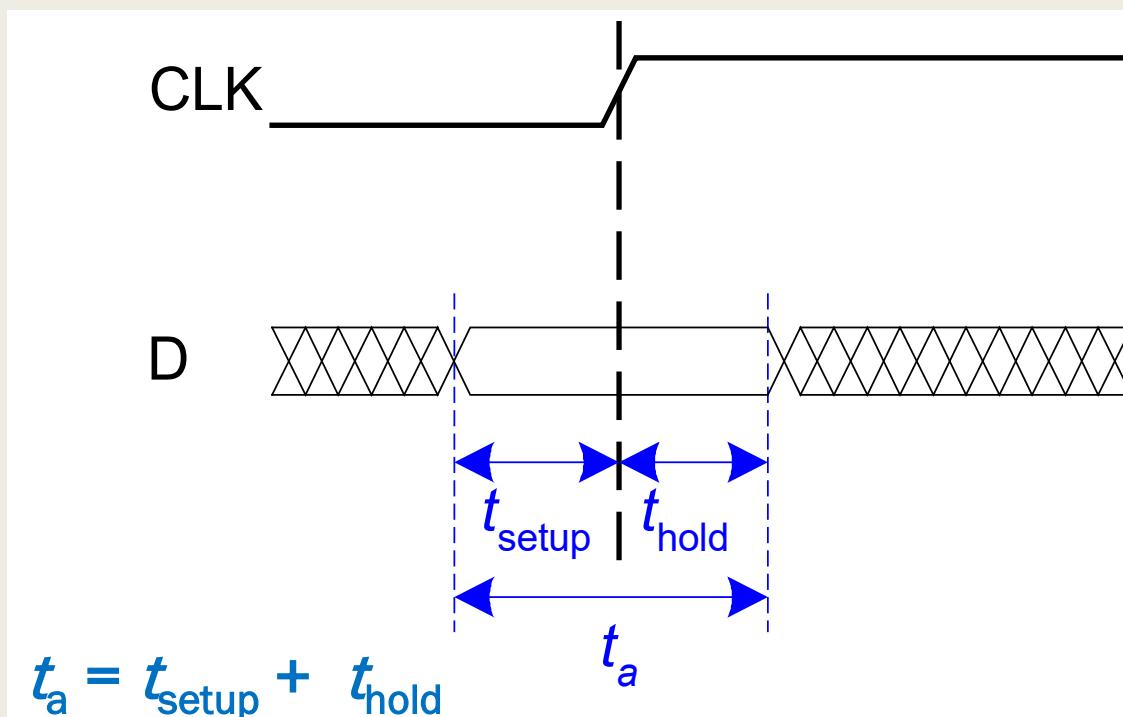


Χρονικό περιορισμοί στη δειγματοληψία

■ Χρονικοί περιορισμοί στην είσοδο D του D Flip-Flop

που πρέπει να ικανοποιούνται κατά τη δειγματοληψία:

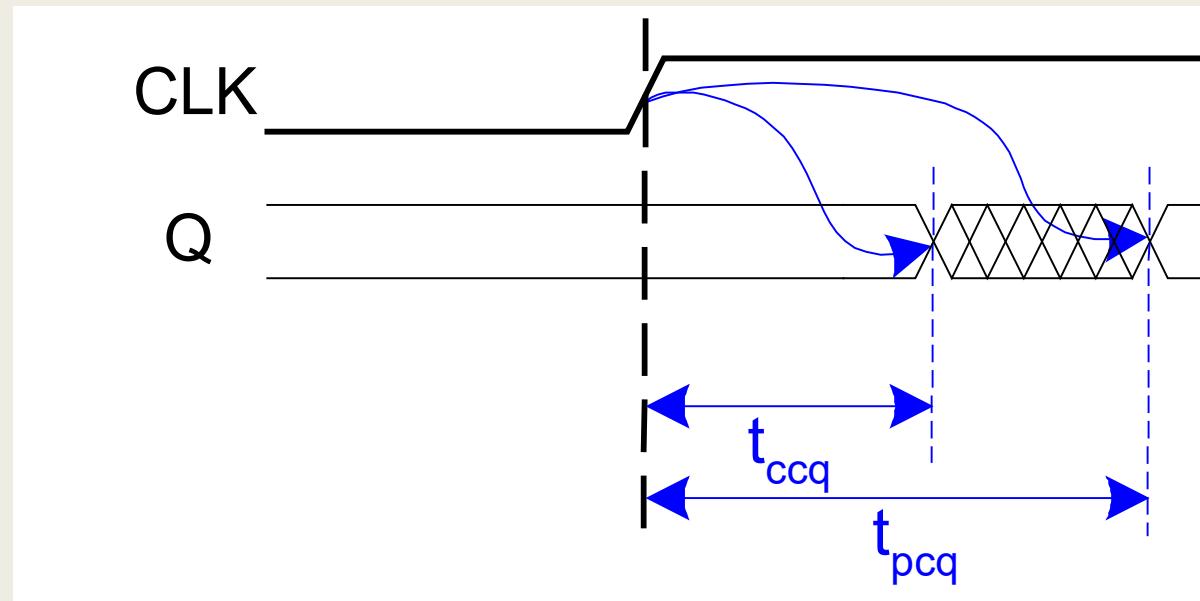
- **Χρόνος σταθεροποίησης (set-up time, t_{setup})** = ο χρόνος που η είσοδος D πρέπει να παραμένει σταθερή **πριν** από την ανερχόμενη ακμή του CLK
- **Χρόνος διατήρησης (hold time, t_{hold})** = ο χρόνος που η είσοδος D πρέπει να παραμένει σταθερή **μετά** από την ανερχόμενη ακμή του CLK
- **Χρόνος ανοίγματος (aperture time, t_a)** = ο χρόνος που η είσοδος D πρέπει να παραμένει σταθερή **πριν και μετά** από την ανερχόμενη ακμή του CLK



Χρονικοί περιορισμοί στη δειγματοληψία

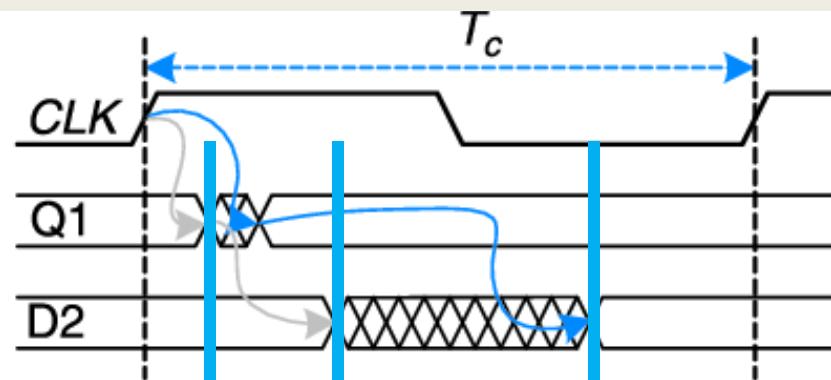
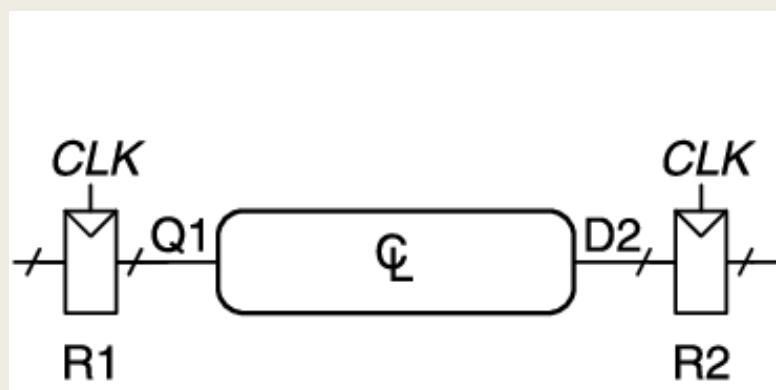
- **Χρονικοί περιορισμοί στην έξοδο Q του D Flip-Flop**
που πρέπει να ικανοποιούνται κατά τη δειγματοληψία:

- **Καθυστέρηση διάδοσης (propagation clock to Q delay, t_{pcq})** =
ο μέγιστος χρόνος εντός του οποίου η έξοδος Q πρέπει να έχει
οπωσδήποτε σταθεροποιηθεί μετά από την ανερχόμενη ακμή του CLK
- **Καθυστέρηση μόλυνσης (contamination clock to Q delay, t_{ccq})** =
ο ελάχιστος χρόνος που πρέπει να παρέλθει ώστε η έξοδος Q
να αρχίσει να μεταβάλλεται μετά από την ανερχόμενη ακμή του CLK



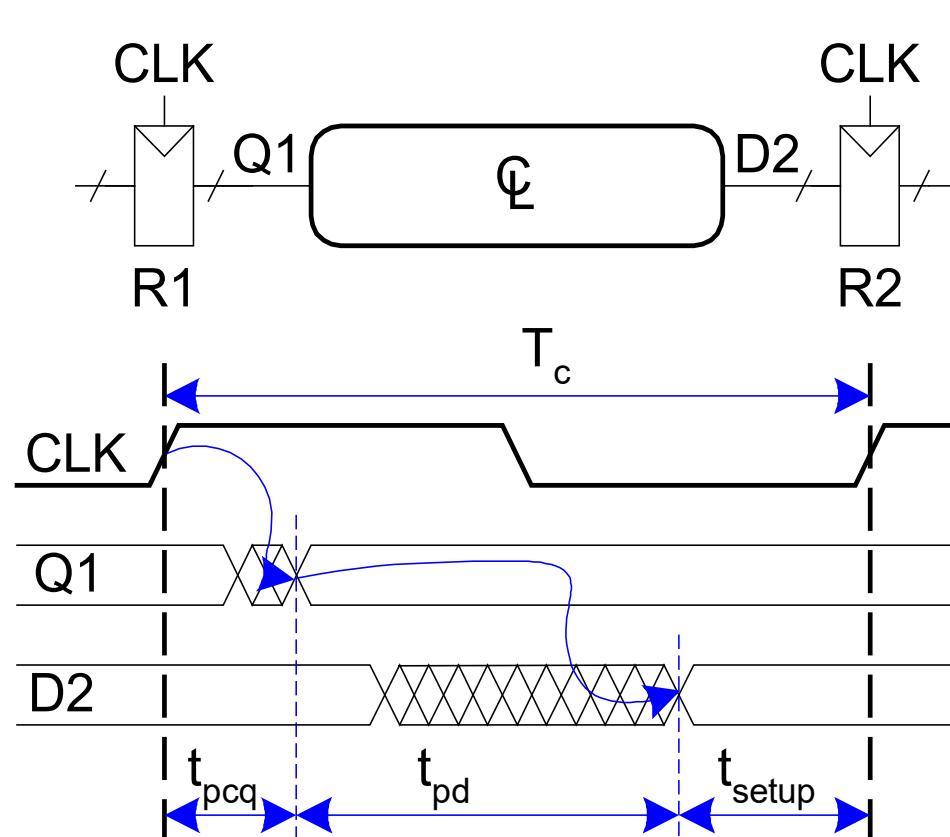
Σχεδίαση στο επίπεδο της μεταφοράς καταχωρητών

- Οι **μέγιστες** και οι **ελάχιστες** καθυστερήσεις ανάμεσα στους καταχωρητές εξαρτώνται από τις καθυστερήσεις στους καταχωρητές και την συνδυαστική λογική (καθυστέρηση διάδοσης t_{pd} και μόλυνσης t_{cd})
- Ένα σήμα κατά τη διάδοσή του μέσα από τη συνδυαστική λογική μπορεί να εμφανίσει μεταβατικό παλμό και να ταλαντώνεται για κάποιο φραγμένο χρονικό διάστημα
- Σύμφωνα με τη αρχή της δυναμικής πειθαρχίας το σήμα θα πρέπει να έχει σταθεροποιηθεί πριν την επόμενη ανερχόμενη ακμή του CLK
 - Υπολογίζεται η **ελάχιστη περίοδος του CLK (T_c)** που εξασφαλίζει την κανονική λειτουργία του συστήματος
 - Αντίστοιχα, ορίζεται η **μέγιστη συχνότητα του CLK ($F_c = 1/T_c$)**



Ελάχιστη περίοδος του CLK T_c

- Υπολογίζεται από τη **μέγιστη καθυστέρηση διάδοσης** μεταξύ των καταχωρητών R1 και του R2 διαμέσου της συνδυαστικής λογικής
 - *Η είσοδος του R2 πρέπει να έχει σταθεροποιηθεί τουλάχιστον κατά τον χρόνο σταθεροποίησης t_{setup} πριν την ανερχόμενη ακμή του CLK*
 - *Εξαρτάται από την καθυστέρηση διάδοσης t_{pd} της συνδυαστικής λογικής*
 - πλήθος διαδοχικών πυλών που μπορούν να υπάρχουν στην κρίσιμη διαδρομή



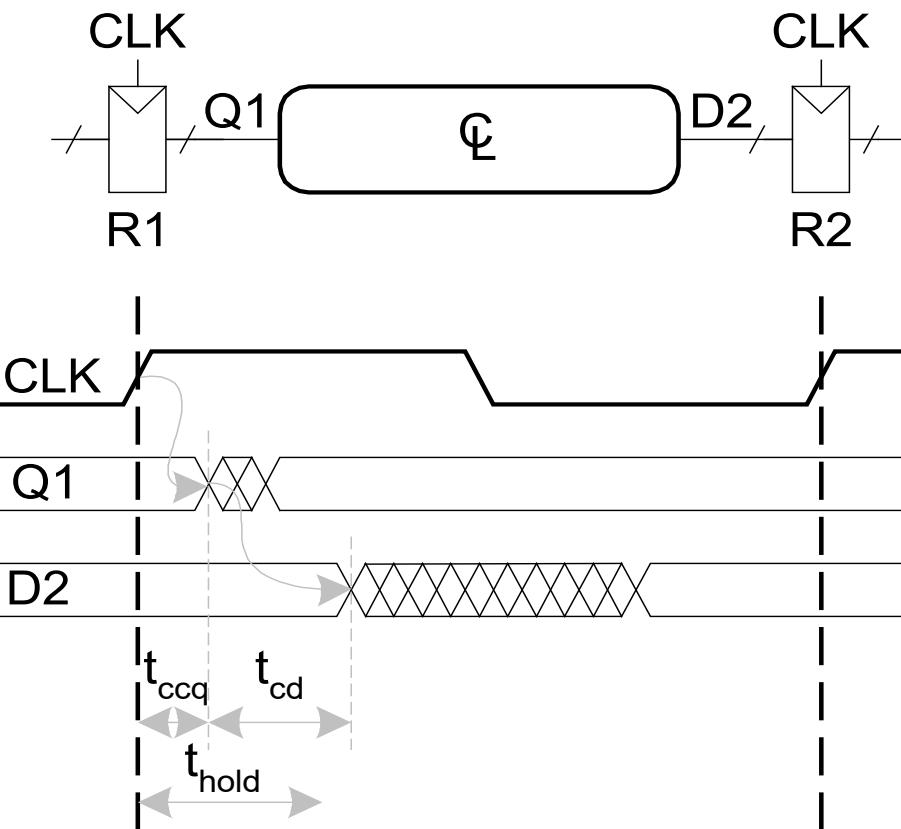
$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$
$$t_{pd} \leq T_c - (t_{pcq} + t_{setup})$$

t_{pcq} : καθυστέρηση διάδοσης του καταχωρητή
 t_{pd} : καθυστέρηση διάδοσης της συνδυαστικής λογικής
 t_{setup} : χρόνος σταθεροποίησης

Επιβάρυνση δημιουργίας ακολουθίας (sequencing overhead): $t_{pcq} + t_{setup}$

Ελάχιστη καθυστέρηση μόλυνσης t_{cd}

- Εξαρτάται από την **ελάχιστη καθυστέρηση διάδοσης** μεταξύ των καταχωρητών R1 και του R2 διαμέσου της συνδυαστικής λογικής
 - Η είσοδος του R2 πρέπει να παραμένει σταθερή του λάχιστον κατά τον **χρόνο διατήρησης** t_{hold} μετά την ανερχόμενη ακμή του CLK
 - Εξαρτάται από την **καθυστέρηση μόλυνσης** t_{cd} της συνδυαστικής λογικής
 - πλήθος διαδοχικών πυλών που μπορούν να υπάρχουν στη σύντομη διαδρομή



$$t_{hold} < t_{ccq} + t_{cd}$$
$$t_{cd} > t_{hold} - t_{ccq}$$

t_{ccq} : καθυστέρηση μόλυνσης του καταχωρητή
 t_{cd} : καθυστέρηση μόλυνσης της συνδυαστικής λογικής
 t_{hold} : χρόνος διατήρησης

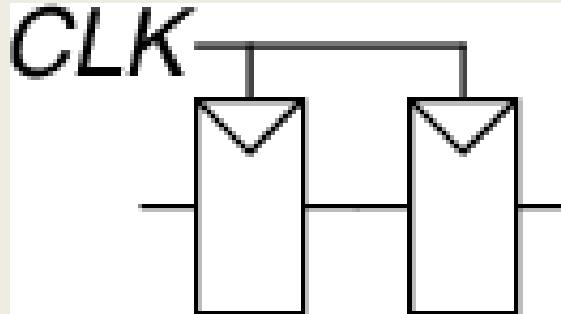
Εξετάζουμε τα κυκλώματα για πιθανή παραβίαση του χρόνου διατήρησης

Τοποθέτηση D Flip-Flop στη σειρά

- Για να τοποθετηθούν δύο D Flip-Flops στη σειρά θα πρέπει να ισχύουν οι συνθήκες:

$$T_c \geq t_{pcq} + t_{\text{setup}}$$

$$t_{\text{hold}} < t_{ccq}$$

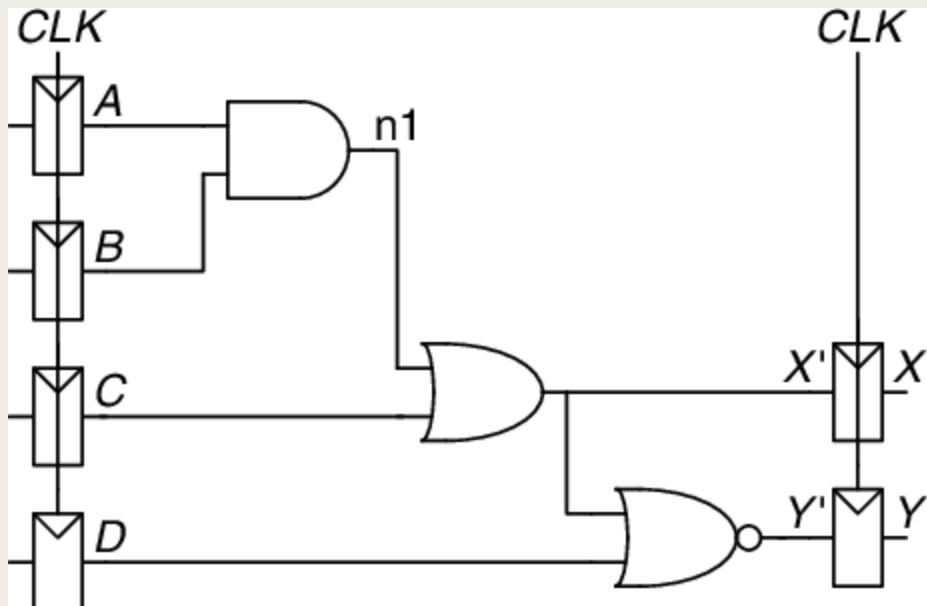


$$T_c \geq t_{pcq} + \cancel{t_{pd}} + t_{\text{setup}}$$
$$t_{\text{hold}} < t_{ccq} + \cancel{t_{cl}}$$

- Συνήθως, στην πράξη $t_{\text{hold}} = 0$
- Οι **περιορισμοί λόγω του χρόνου διατήρησης** είναι σημαντικοί
 - Αν παραβιάζονται, η μόνη λύση είναι να αυξηθεί η καθυστέρηση μόλυνσης για τη διέλευση μέσα από τα κυκλώματα λογικής, κάτι που απαιτεί την **εκ νέου σχεδίαση του κυκλώματος**
 - Σε αντίθεση με τους περιορισμούς λόγω του χρόνου σταθεροποίησης, δεν μπορούμε να διορθώσουμε τους περιορισμούς λόγω του χρόνου διατήρησης ρυθμίζοντας απλώς την περίοδο του ρολογιού.

Ανάλυση Χρονισμού*

- Δεδομένου του σχηματικού διαγράμματος ενός κυκλώματος και των χαρακτηριστικών χρονισμού συγκεκριμένης τεχνολογίας υπολογίζουμε:
 - τη μέγιστη συχνότητα λειτουργίας ($F_c = 1/T_c$)



Συνδυαστική λογική:

$$t_{pd} = 3 \times 40 \text{ ps} = 120 \text{ ps}$$

$$t_{cd} = 1 \times 25 \text{ ps} = 25 \text{ ps}$$

Χαρακτηριστικά χρονισμού

Καταχωρητή:

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 80 \text{ ps}$$

$$t_{\text{setup}} = 50 \text{ ps}$$

$$t_{\text{hold}} = 60 \text{ ps}$$

Πύλης

$$t_{pd} = 40 \text{ ps}$$

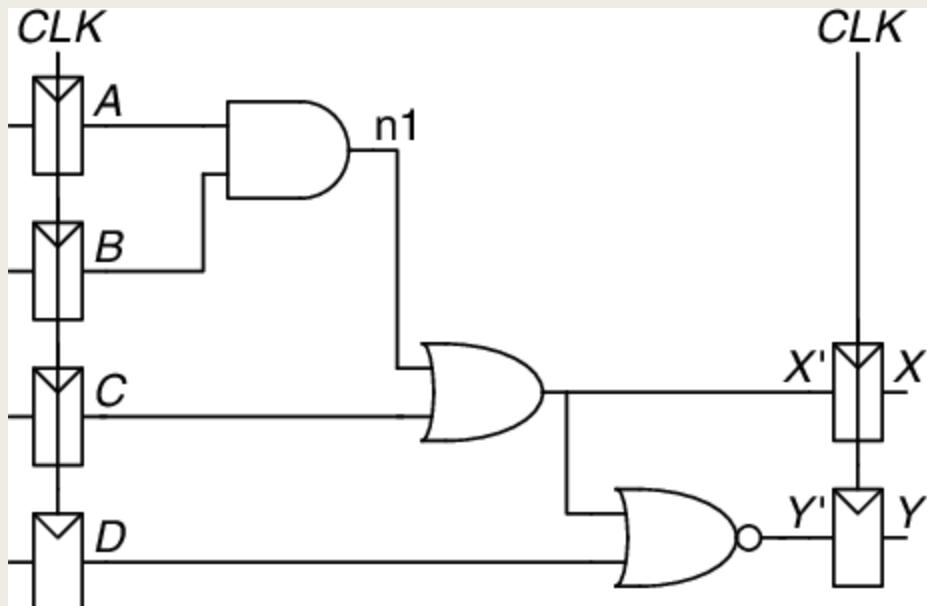
$$t_{cd} = 25 \text{ ps}$$

$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} = (80 + 120 + 50) \text{ ps} = 250 \text{ ps}$$
$$F_c = 4 \text{ GHz}$$

*Παράδειγμα 3.10

Ανάλυση Χρονισμού*

- Δεδομένου του σχηματικού διαγράμματος ενός κυκλώματος και των χαρακτηριστικών χρονισμού συγκεκριμένης τεχνολογίας υπολογίζουμε:
 - την *πιθανή παραβίαση του χρόνου διατήρησης*



Συνδυαστική λογική:

$$t_{pd} = 3 \times 40 \text{ ps} = 120 \text{ ps}$$

$$t_{cd} = 1 \times 25 \text{ ps} = 25 \text{ ps}$$

Χαρακτηριστικά χρονισμού

Καταχωρητή:

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 80 \text{ ps}$$

$$t_{\text{setup}} = 50 \text{ ps}$$

$$t_{\text{hold}} = 60 \text{ ps}$$

Πύλης

$$t_{pd} = 40 \text{ ps}$$

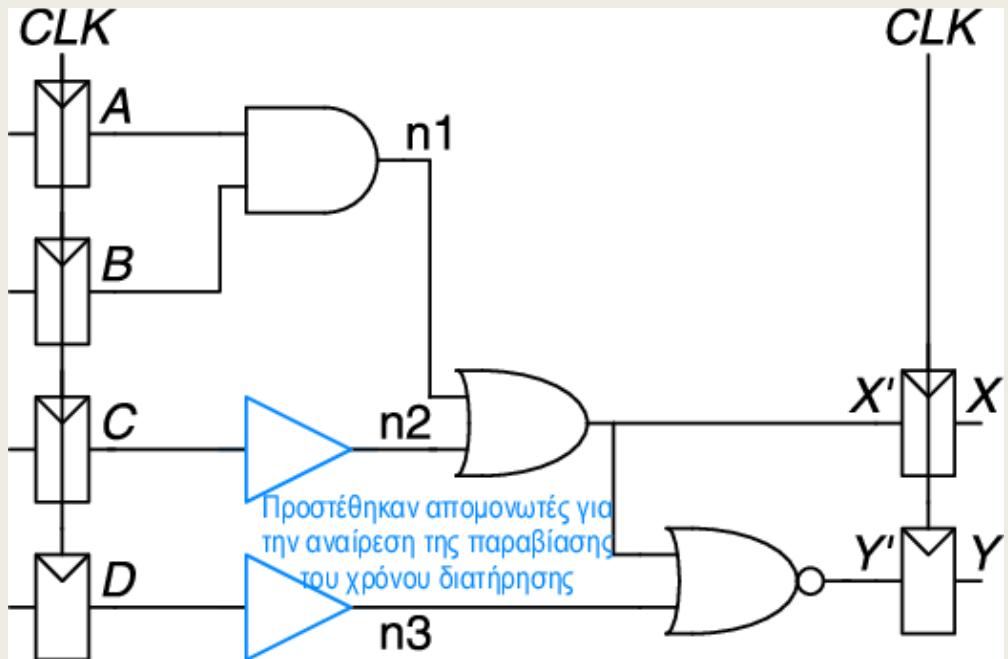
$$t_{cd} = 25 \text{ ps}$$

$$t_{\text{hold}} < t_{ccq} + t_{cd} \Rightarrow 60 \text{ ps} < (30 + 25) \text{ ps} \quad (\text{Δεν ισχύει})$$

*Παράδειγμα 3.10

Διόρθωση παραβιάσεων του χρόνου διατήρησης*

- Θα πρέπει να επανασχεδιάσουμε το κύκλωμα με την προσθήκη απομονωτών στις σύντομες διαδρομές, ώστε να μην παραβιάζεται ο χρόνος διατήρησης



Χαρακτηριστικά χρονισμού

Καταχωρητή:

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 80 \text{ ps}$$

$$t_{\text{setup}} = 50 \text{ ps}$$

$$t_{\text{hold}} = 60 \text{ ps}$$

Πύλης

$$t_{pd} = 40 \text{ ps}$$

$$t_{cd} = 25 \text{ ps}$$

Συνδυαστική λογική:

$$t_{pd} = 3 \times 40 \text{ ps} = 120 \text{ ps}$$

$$t_{cd} = 2 \times 25 \text{ ps} = 50 \text{ ps}$$

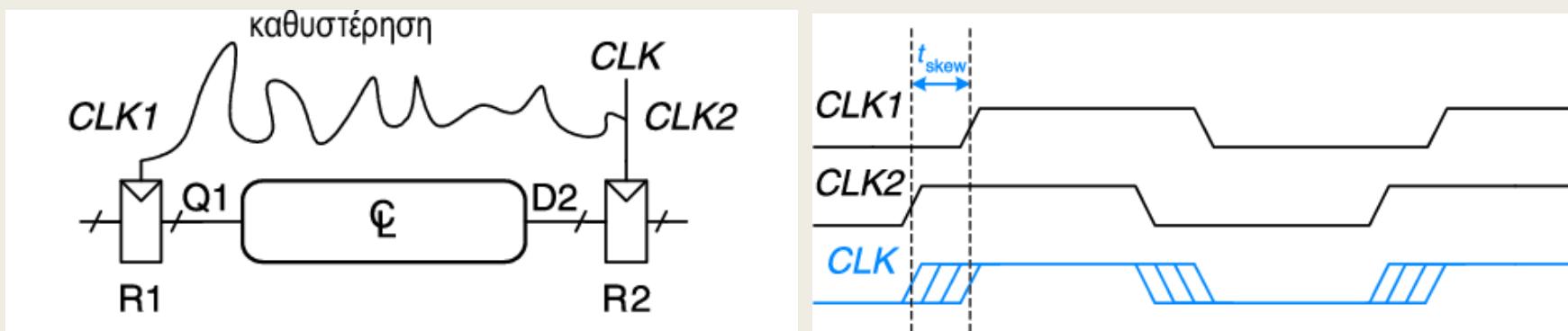
$$t_{\text{hold}} < t_{ccq} + t_{cd} \Rightarrow 60 \text{ ps} < (30 + 50) \text{ ps (OK!)}$$

Προσοχή! Κάποιοι μικροεπεξεργαστές υψηλών επιδόσεων, χρησιμοποιούν pulsed latches με μικρή καθυστέρηση από το ρολόι έως την έξοδο Q , αλλά μεγάλο χρόνο διατήρησης

*Παράδειγμα 3.11

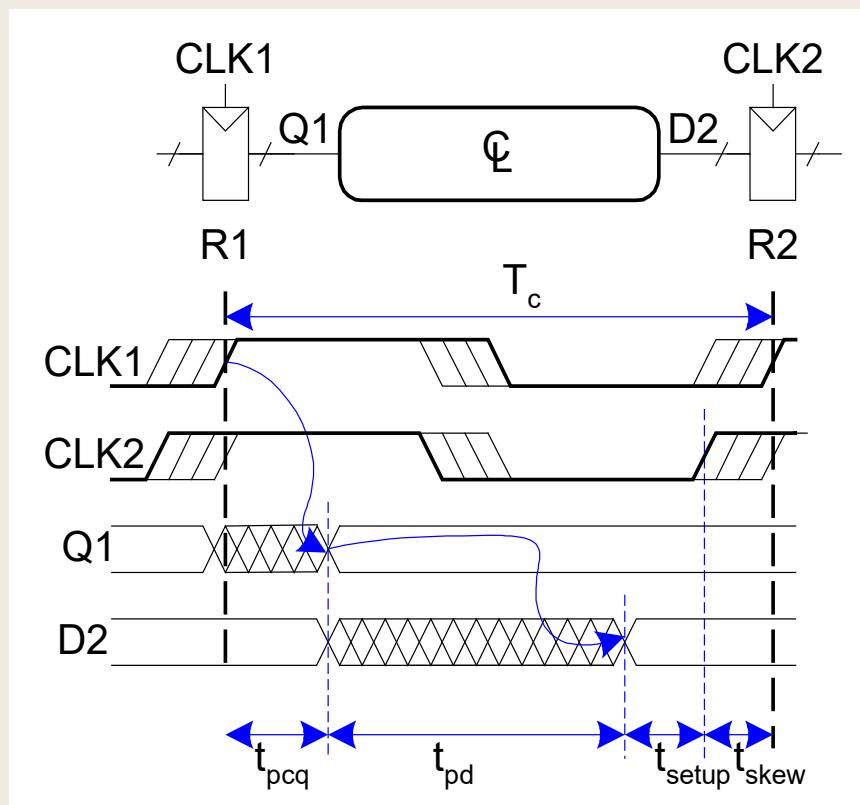
Απόκλιση ρολογιού t_{skew}

- Το σήμα του ρολογιού CLK δεν φθάνει συνήθως στους καταχωρητές ακριβώς την ίδια χρονική στιγμή εξ αιτίας διαφορετικών καθυστερήσεων διάδοσης που οφείλονται:
 - Στην ύπαρξη διαφορετικού μήκους διαδρομών από την πηγή του σήματος CLK μέχρι την είσοδο του ρολογιού στον εκάστοτε καταχωρητή
 - Στην επίδραση του θορύβου
 - Στην προσθήκη πύλης σε κάποια από τις διαδρομές του ρολογιού
- Απόκλιση ρολογιού ονομάζεται η αυξομείωση στην εμφάνιση των ανερχόμενων ακμών του σήματος CLK
 - Στην πράξη λαμβάνουμε υπόψη τη **χειρότερη περίπτωση του χρόνου απόκλισης ρολογιού (clock skew) t_{skew}**



Νέα ελάχιστη περίοδος του CLK T_c

- Στην χειρότερη περίπτωση το CLK2 έρχεται νωρίτερα από το CLK1 κατά t_{skew}
- Ο χρόνος σταθεροποίησης t_{setup} επιβαρύνεται με τον χρόνο απόκλισης t_{skew}



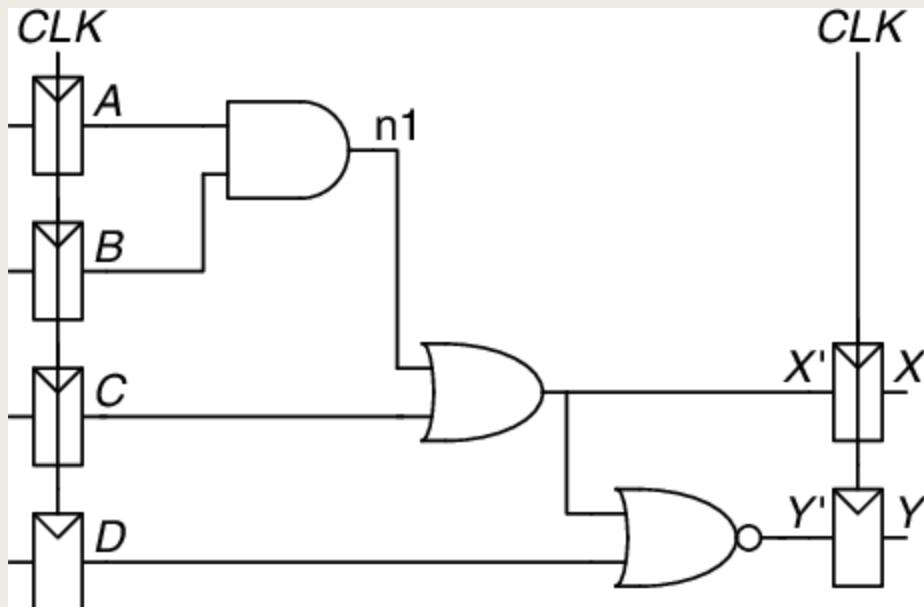
$$T_c \geq t_{pcq} + t_{pd} + (t_{setup} + t_{skew})$$
$$t_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew})$$

t_{pcq} : καθυστέρηση διάδοσης του καταχωρητή
 t_{pd} : καθυστέρηση διάδοσης της συνδυαστικής λογικής
 t_{setup} : χρόνος σταθεροποίησης
 t_{skew} : χρόνος απόκλισης

Επιβάρυνση δημιουργίας ακολουθίας (sequencing overhead): $t_{pcq} + t_{setup} + t_{skew}$

Ανάλυση Χρονισμού με t_{skew} *

- Δεδομένου του σχηματικού διαγράμματος ενός κυκλώματος και των χαρακτηριστικών χρονισμού συγκεκριμένης τεχνολογίας υπολογίζουμε:
 - τη μέγιστη συχνότητα λειτουργίας ($F_c = 1/T_c$) λαμβάνοντας υπόψη και τη χειρότερη περίπτωση του χρόνου απόκλισης



Συνδυαστική λογική:

$$t_{pd} = 3 \times 40 \text{ ps} = 120 \text{ ps}$$

$$t_{cd} = 1 \times 25 \text{ ps} = 25 \text{ ps}$$

Χαρακτηριστικά χρονισμού

Καταχωρητή:

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 80 \text{ ps}$$

$$t_{\text{setup}} = 50 \text{ ps} \quad t_{\text{skew}} = 50 \text{ ps}$$

$$t_{\text{hold}} = 60 \text{ ps}$$

Πύλης

$$t_{pd} = 40 \text{ ps}$$

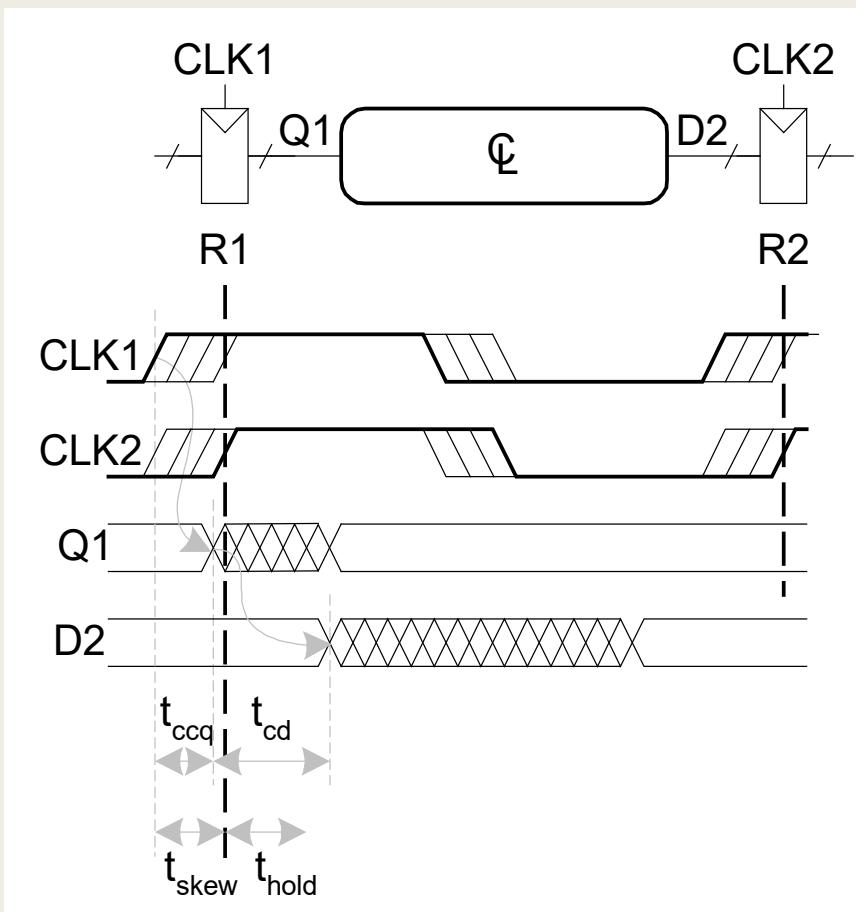
$$t_{cd} = 25 \text{ ps}$$

$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} + t_{\text{skew}} = (80 + 120 + 50 + 50) \text{ ps} = 300 \text{ ps}$$
$$F_c = 3.33 \text{ GHz} \text{ αντί } 4 \text{ GHz}$$

*Παράδειγμα 3.12

Νέα ελάχιστη καθυστέρηση μόλυνσης t_{cd}

- Στην χειρότερη περίπτωση το CLK1 έρχεται νωρίτερα από το CLK2 κατά t_{skew}
- Ο χρόνος διατήρησης t_{hold} επιβαρύνεται με τον χρόνο απόκλισης t_{skew}



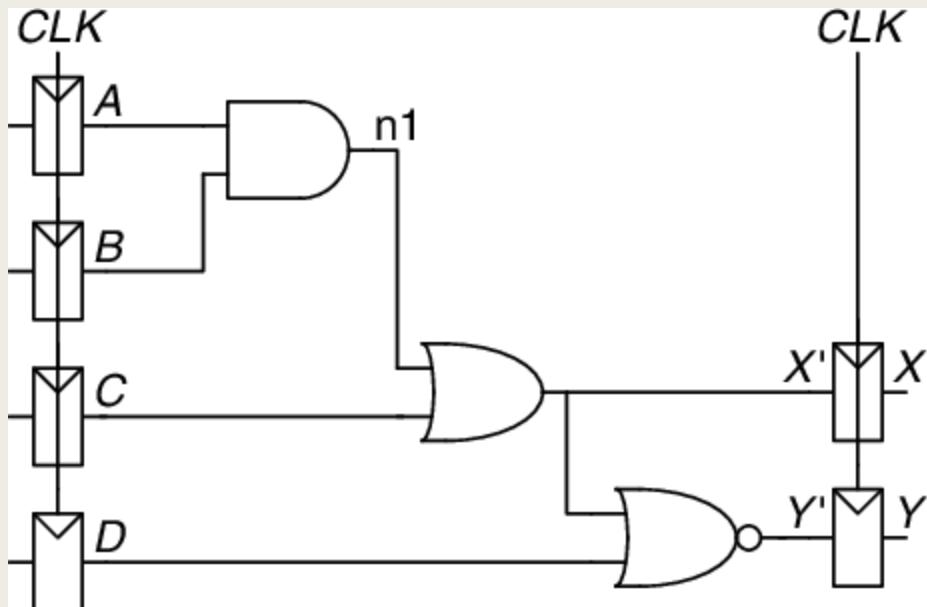
$$t_{hold} + t_{skew} < t_{ccq} + t_{cd}$$
$$t_{cd} > (t_{hold} + t_{skew}) - t_{ccq}$$

t_{ccq} : καθυστέρηση μόλυνσης του καταχωρητή
 t_{cd} : καθυστέρηση μόλυνσης της συνδυαστικής λογικής
 t_{hold} : χρόνος διατήρησης
 t_{skew} : χρόνος απόκλισης

Εξετάζουμε τα κυκλώματα για πιθανή παραβίαση του χρόνου διατήρησης με την επιβάρυνση του χρόνου απόκλισης

Ανάλυση Χρονισμού με t_{skew} *

- Δεδομένου του σχηματικού διαγράμματος ενός κυκλώματος και των χαρακτηριστικών χρονισμού συγκεκριμένης τεχνολογίας υπολογίζουμε:
 - την **πιθανή παραβίαση του χρόνου διατήρησης** λαμβάνοντας υπόψη και τη χειρότερη περίπτωση του **χρόνου απόκλισης**



Συνδυαστική λογική:

$$t_{pd} = 3 \times 40 \text{ ps} = 120 \text{ ps}$$

$$t_{cd} = 1 \times 25 \text{ ps} = 25 \text{ ps}$$

Χαρακτηριστικά χρονισμού

Καταχωρητή:

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 80 \text{ ps}$$

$$t_{\text{setup}} = 50 \text{ ps} \quad t_{\text{skew}} = 50 \text{ ps}$$

$$t_{\text{hold}} = 60 \text{ ps}$$

Πύλης

$$t_{pd} = 40 \text{ ps}$$

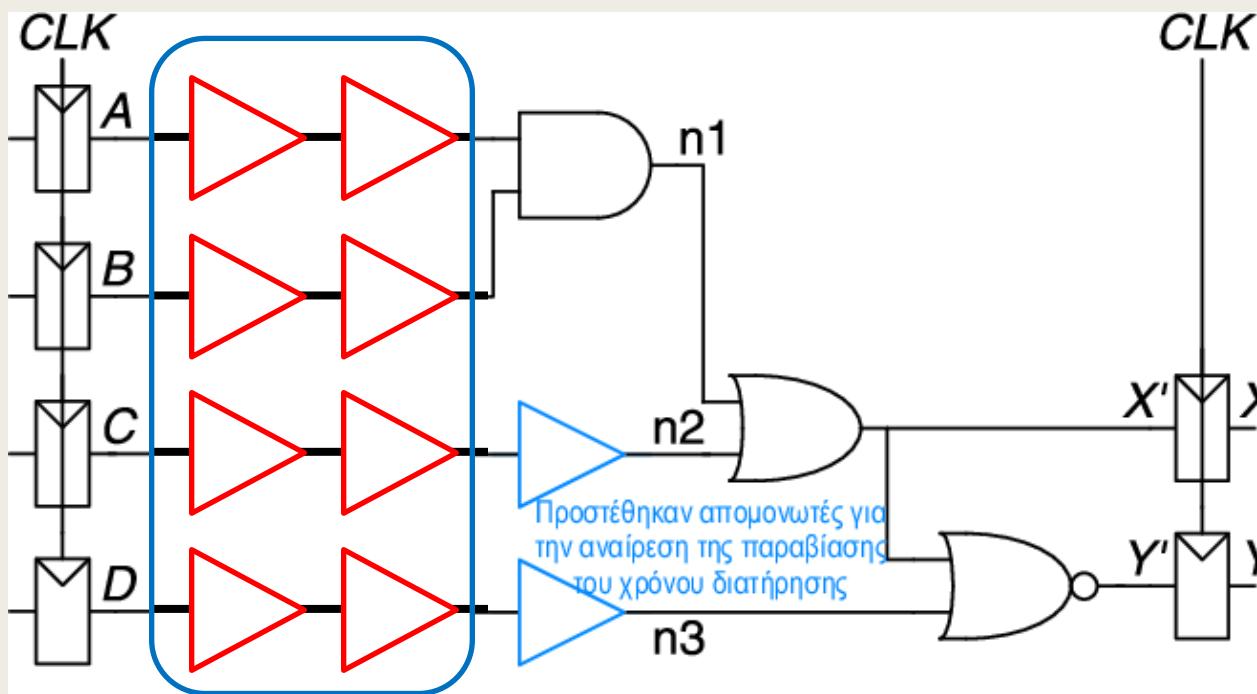
$$t_{cd} = 25 \text{ ps}$$

$$t_{\text{hold}} + t_{\text{skew}} < t_{ccq} + t_{cd} \Rightarrow (60+50) \text{ ps} < (30+25) \text{ ps}$$

(Δεν ισχύει και γίνεται χειρότερο, 110 ps αντί 60 ps)

Διόρθωση παραβιάσεων του χρόνου διατήρησης*

- Θα πρέπει να επανασχεδιάσουμε το κύκλωμα με την προσθήκη απομονωτών στις σύντομες διαδρομές, ώστε να μην παραβιάζεται ο χρόνος διατήρησης λαμβάνοντας υπόψη και τη χειρότερη περίπτωση του **χρόνου απόκλισης**
 - Απαιτούνται περισσότεροι απομονωτές (επιπλέον 2 σε κάθε διαδρομή), αλλά αυξάνει σημαντικά η ελάχιστη περίοδος του CLK T_c (από 300 ps σε 380 ps)



Χαρ/κά χρονισμού

Καταχωρητή:

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 80 \text{ ps}$$

$$t_{\text{setup}} = 50 \text{ ps}$$

$$t_{\text{hold}} = 60 \text{ ps}$$

$$t_{\text{skew}} = 50 \text{ ps}$$

Πύλης

$$t_{pd} = 40 \text{ ps}$$

$$t_{cd} = 25 \text{ ps}$$

Συνδυαστική λογική:

$$t_{pd} = 5 \times 40 \text{ ps} = 200 \text{ ps}$$

$$t_{cd} = 4 \times 25 \text{ ps} = 100 \text{ ps}$$

$$t_{\text{hold}} + t_{\text{skew}} < t_{ccq} + t_{cd} \Rightarrow (60 + 50) \text{ ps} < (30 + 100) \text{ ps}$$

(OK!)

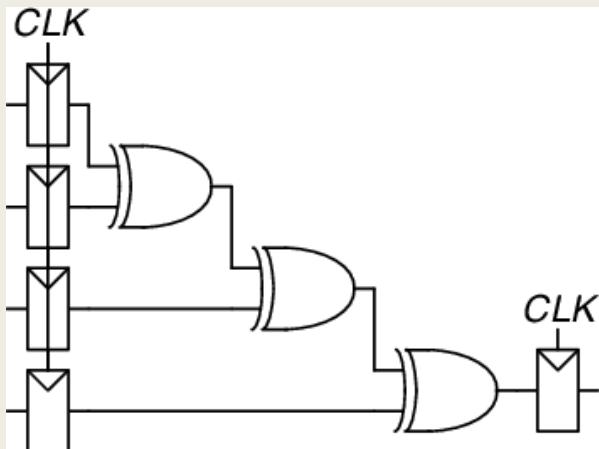
Επιλεγμένες ασκήσεις

Άσκηση 3.33

Ο Γιάννης Μπιτάκης έχει σχεδιάσει το κύκλωμα της Εικόνας 3.74 ώστε να υπολογίζει την τιμή μιας καταχωρισμένης (registered) συνάρτησης XOR με 4 εισόδους.

- (a) Ποια είναι η μέγιστη συχνότητα λειτουργίας του κυκλώματος με ρολόι χωρίς απόκλιση;
- (b) Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα αν πρέπει να λειτουργεί στα 2 GHz;
- (c) Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα προτού αντιμετωπίσει μία πιθανή παραβίαση του χρόνου διατήρησης;

Στον πίνακα δίδονται τα χαρακτηριστικά χρονισμού των στοιχείων του κυκλώματος



Συνδυαστική λογική:

$$t_{pd} = 3 \times 100 \text{ ps} = 300 \text{ ps}$$

$$t_{cd} = 1 \times 55 \text{ ps} = 55 \text{ ps}$$

Χαρ/κά χρονισμού

Καταχωρητή:

$$t_{ccq} = 50 \text{ ps}$$

$$t_{pcq} = 70 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 20 \text{ ps}$$

Πύλης XOR

$$t_{pd} = 100 \text{ ps}$$

$$t_{cd} = 55 \text{ ps}$$

(a) $T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} = (70+300+60) \text{ ps} = 430 \text{ ps}$

$$f = 1 / 430 \text{ ps} = 2,33 \text{ GHz}$$

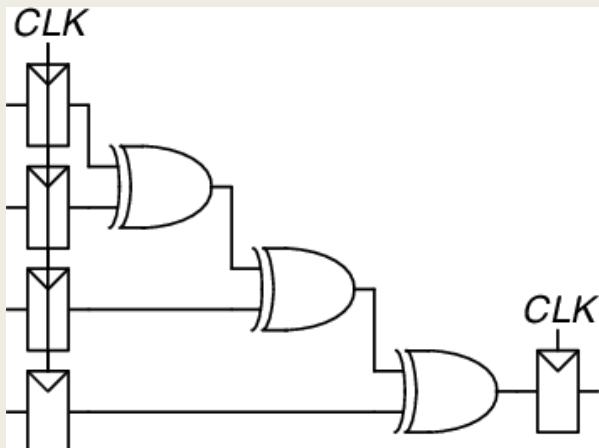
Επιλεγμένες ασκήσεις

Άσκηση 3.33

Ο Γιάννης Μπιτάκης έχει σχεδιάσει το κύκλωμα της Εικόνας 3.74 ώστε να υπολογίζει την τιμή μιας καταχωρισμένης (registered) συνάρτησης XOR με 4 εισόδους.

- Ποια είναι η μέγιστη συχνότητα λειτουργίας του κυκλώματος με ρολόι χωρίς απόκλιση;
- Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα αν πρέπει να λειτουργεί στα 2 GHz;
- Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα προτού αντιμετωπίσει μία πιθανή παραβίαση του χρόνου διατήρησης;

Στον πίνακα δίδονται τα χαρακτηριστικά χρονισμού των στοιχείων του κυκλώματος



Συνδυαστική λογική:

$$t_{pd} = 3 \times 100 \text{ ps} = 300 \text{ ps}$$

$$t_{cd} = 1 \times 55 \text{ ps} = 55 \text{ ps}$$

Χαρ/κά χρονισμού

Καταχωρητή:

$$t_{ccq} = 50 \text{ ps}$$

$$t_{pcq} = 70 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 20 \text{ ps}$$

Πύλης XOR

$$t_{pd} = 100 \text{ ps}$$

$$t_{cd} = 55 \text{ ps}$$

(b) $T_c = 1 / f = 1 / 2 \text{ GHz} = 500 \text{ ps}$

$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} + t_{\text{skew}} \Rightarrow t_{\text{skew}} \leq T_c - (t_{pcq} + t_{pd} + t_{\text{setup}})$$

$$t_{\text{skew}} \leq [500 - (70 + 300 + 60)] \text{ ps} = (500 - 430) \text{ ps} = 70 \text{ ps}$$

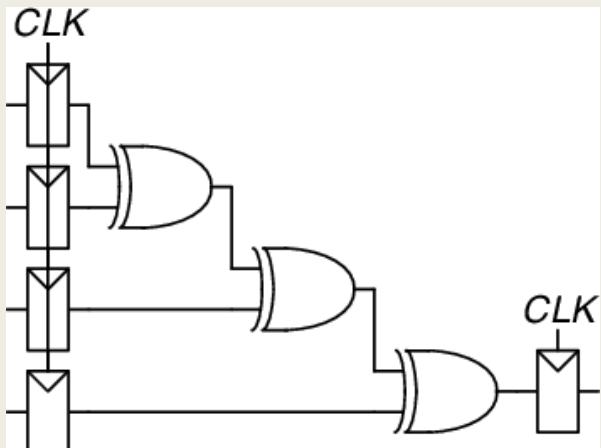
Επιλεγμένες ασκήσεις

Άσκηση 3.33

Ο Γιάννης Μπιτάκης έχει σχεδιάσει το κύκλωμα της Εικόνας 3.74 ώστε να υπολογίζει την τιμή μιας καταχωρισμένης (registered) συνάρτησης XOR με 4 εισόδους.

- Ποια είναι η μέγιστη συχνότητα λειτουργίας του κυκλώματος με ρολόι χωρίς απόκλιση;
- Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα αν πρέπει να λειτουργεί στα 2 GHz;
- Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα προτού αντιμετωπίσει μία πιθανή παραβίαση του χρόνου διατήρησης;

Στον πίνακα δίδονται τα χαρακτηριστικά χρονισμού των στοιχείων του κυκλώματος



Συνδυαστική λογική:

$$t_{pd} = 3 \times 100 \text{ ps} = 300 \text{ ps}$$

$$t_{cd} = 1 \times 55 \text{ ps} = 55 \text{ ps}$$

Χαρ/κά χρονισμού

Καταχωρητή:

$$t_{ccq} = 50 \text{ ps}$$

$$t_{pcq} = 70 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 20 \text{ ps}$$

Πύλης XOR

$$t_{pd} = 100 \text{ ps}$$

$$t_{cd} = 55 \text{ ps}$$

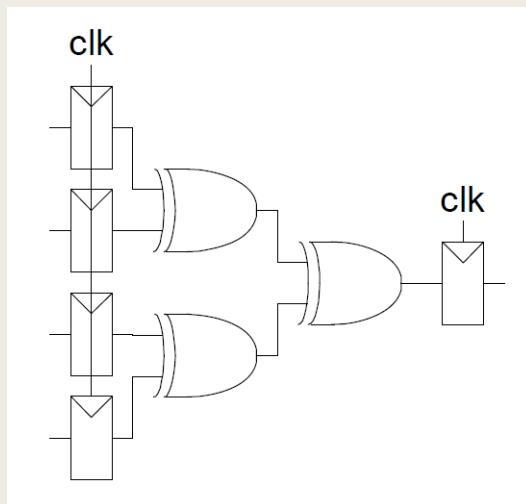
(c) $t_{\text{hold}} + t_{\text{skew}} < t_{ccq} + t_{cd} \Rightarrow t_{\text{skew}} < t_{ccq} + t_{cd} - t_{\text{hold}} \Rightarrow$
 $t_{\text{skew}} < (50 + 55 - 20) \text{ ps} = 85 \text{ ps}$

Επιλεγμένες ασκήσεις

Άσκηση 3.33

Η Μαρία Χακερίδου ισχυρίζεται ότι μπορεί να σχεδιάσει εκ νέου το τμήμα συνδυαστικής λογικής μεταξύ των καταχωρητών ώστε να είναι πιο γρήγορο.

- Το βελτιωμένο κύκλωμά της χρησιμοποιεί επίσης πύλες XOR με δύο εισόδους, οι οποίες όμως έχουν διαφορετική διάταξη
- (a) Ποια είναι η μέγιστη συχνότητά λειτουργίας του κυκλώματος;
- (b) Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα προτού αντιμετωπίσει μία πιθανή παραβίαση του χρόνου διατήρησης;



Συνδυαστική λογική:

$$t_{pd} = 2 \times 100 \text{ ps} = 200 \text{ ps}$$

$$t_{cd} = 2 \times 55 \text{ ps} = 110 \text{ ps}$$

Χαρ/κά χρονισμού

Καταχωρητή:

$$t_{ccq} = 50 \text{ ps}$$

$$t_{pcq} = 70 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 20 \text{ ps}$$

Πύλης XOR

$$t_{pd} = 100 \text{ ps}$$

$$t_{cd} = 55 \text{ ps}$$

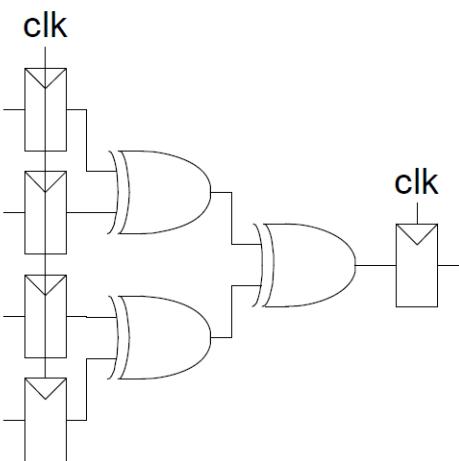
(a) $T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} = (70+200+60) \text{ ps} = 330 \text{ ps}$
 $f = 1 / 330 \text{ ps} = 3,03 \text{ GHz}$

Επιλεγμένες ασκήσεις

Άσκηση 3.33

Η Μαρία Χακερίδου ισχυρίζεται ότι μπορεί να σχεδιάσει εκ νέου το τμήμα συνδυαστικής λογικής μεταξύ των καταχωρητών ώστε να είναι πιο γρήγορο.

- Το βελτιωμένο κύκλωμά της χρησιμοποιεί επίσης πύλες XOR με δύο εισόδους, οι οποίες όμως έχουν διαφορετική διάταξη
- Ποιο είναι το σχηματικό διάγραμμα του κυκλώματός της;
- (a) Ποια είναι η μέγιστη συχνότητά λειτουργίας του κυκλώματος;
- (b) Πόση απόκλιση του ρολογιού μπορεί να ανέχεται το κύκλωμα προτού αντιμετωπίσει μία πιθανή παραβίαση του χρόνου διατήρησης;



Συνδυαστική λογική:

$$t_{pd} = 2 \times 100 \text{ ps} = 200 \text{ ps}$$

$$t_{cd} = 2 \times 55 \text{ ps} = 110 \text{ ps}$$

Χαρ/κά χρονισμού

Καταχωρητή:

$$t_{ccq} = 50 \text{ ps}$$

$$t_{pcq} = 70 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 20 \text{ ps}$$

Πύλης XOR

$$t_{pd} = 100 \text{ ps}$$

$$t_{cd} = 55 \text{ ps}$$

(b) $t_{\text{hold}} + t_{\text{skew}} < t_{ccq} + t_{cd} \Rightarrow t_{\text{skew}} < t_{ccq} + t_{cd} - t_{\text{hold}} \Rightarrow$
 $t_{\text{skew}} < (50 + 110 - 20) \text{ ps} = 140 \text{ ps}$

Επιλεγμένες ασκήσεις

■ Άσκηση 3.35

Μια προγραμματιζόμενη από τον χρήστη διάταξη πυλών FPGA, υλοποιεί κυκλώματα συνδυαστικής λογικής χρησιμοποιώντας διαμορφώσιμες λογικές δομικές μονάδες (configurable logic blocks, CLB) στη θέση των λογικών πυλών.

- Η διάταξη Spartan 3 της εταιρείας Xilinx έχει καθυστέρηση διάδοσης και μόλυνσης ίση με 0,61 και 0,30 ns, αντίστοιχα, για κάθε CLB.
- Περιέχει επίσης D Flip-Flop με καθυστερήσεις διάδοσης και μόλυνσης ίσες με 0,72 και 0,50 ns, και χρόνους σταθεροποίησης και διατήρησης ίσους με 0,53 και 0 ns, αντίστοιχα.

Αν κατασκευάζετε ένα σύστημα που πρέπει να λειτουργεί στα 40 MHz, πόσες διαδοχικές μονάδες CLB μπορείτε να χρησιμοποιήσετε μεταξύ δύο D Flip-Flop;

- Υποθέστε ότι το ρολόι δεν παρουσιάζει απόκλιση και δεν παρατηρείται καθυστέρηση κατά τη διέλευση μέσα από τα σύρματα μεταξύ των CLB

Επιλεγμένες ασκήσεις

■ Άσκηση 3.35

Μια προγραμματιζόμενη από τον χρήστη διάταξη πυλών FPGA, υλοποιεί κυκλώματα συνδυαστικής λογικής χρησιμοποιώντας διαμορφώσιμες λογικές δομικές μονάδες (configurable logic blocks, CLB) στη θέση των λογικών πυλών.

- Η διάταξη Spartan 3 της εταιρείας Xilinx έχει καθυστέρηση διάδοσης και μόλυνσης ίση με 0,61 και 0,30 ns, αντίστοιχα, για κάθε CLB.
- Περιέχει επίσης D Flip-Flop με καθυστερήσεις διάδοσης και μόλυνσης ίσες με 0,72 και 0,50 ns, και χρόνους σταθεροποίησης και διατήρησης ίσους με 0,53 και 0 ns, αντίστοιχα.

Αν κατασκευάζετε ένα σύστημα που πρέπει να λειτουργεί στα 40 MHz, πόσες διαδοχικές μονάδες CLB μπορείτε να χρησιμοποιήσετε μεταξύ δύο D Flip-Flop;

- Υποθέστε ότι το ρολόι δεν παρουσιάζει απόκλιση και δεν παρατηρείται καθυστέρηση κατά τη διέλευση μέσα από τα σύρματα μεταξύ των CLB

$$T_c = 1 / f = 1 / 40 \text{ MHz} = 25 \text{ ns}$$

$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} + t_{\text{skew}} \Rightarrow T_c \geq t_{pcq} + N t_{pdCLB} + t_{\text{setup}} + t_{\text{skew}} \Rightarrow$$

$$N \leq [T_c - (t_{pcq} + t_{\text{setup}} + t_{\text{skew}})] / t_{pdCLB} \Rightarrow N \leq [25 - (0,72 + 0,53 + 0)] \text{ ns} / 0,61 \text{ ns} \Rightarrow$$

$$N \leq 38,9 \Rightarrow N = 38$$

Επιλεγμένες ασκήσεις

■ Άσκηση 3.35

Μια προγραμματιζόμενη από τον χρήστη διάταξη πυλών FPGA, υλοποιεί κυκλώματα συνδυαστικής λογικής χρησιμοποιώντας διαμορφώσιμες λογικές δομικές μονάδες (configurable logic blocks, CLB) στη θέση των λογικών πυλών.

- Η διάταξη Spartan 3 της εταιρείας Xilinx έχει καθυστέρηση διάδοσης και μόλυνσης ίση με 0,61 και 0,30 ns, αντίστοιχα, για κάθε CLB.
- Περιέχει επίσης D Flip-Flop με καθυστερήσεις διάδοσης και μόλυνσης ίσες με 0,72 και 0,50 ns, και χρόνους σταθεροποίησης και διατήρησης ίσους με 0,53 και 0 ns, αντίστοιχα.

Πόση απόκλιση του ρολογιού μπορεί να ανεχτεί η διάταξη FPGA χωρίς να παραβιάζει τον χρόνο διατήρησης;

- Υποθέστε ότι όλες οι διαδρομές μεταξύ D Flip-Flop περνούν από τουλάχιστον μία μονάδα CLB.

Επιλεγμένες ασκήσεις

■ Άσκηση 3.35

Μια προγραμματιζόμενη από τον χρήστη διάταξη πυλών FPGA, υλοποιεί κυκλώματα συνδυαστικής λογικής χρησιμοποιώντας διαμορφώσιμες λογικές δομικές μονάδες (configurable logic blocks, CLB) στη θέση των λογικών πυλών.

- Η διάταξη Spartan 3 της εταιρείας Xilinx έχει καθυστέρηση διάδοσης και μόλυνσης ίση με 0,61 και 0,30 ns, αντίστοιχα, για κάθε CLB.
- Περιέχει επίσης D Flip-Flop με καθυστερήσεις διάδοσης και μόλυνσης ίσες με 0,72 και 0,50 ns, και χρόνους σταθεροποίησης και διατήρησης ίσους με 0,53 και 0 ns, αντίστοιχα.

Πόση απόκλιση του ρολογιού μπορεί να ανεχτεί η διάταξη FPGA χωρίς να παραβιάζει τον χρόνο διατήρησης;

- Υποθέστε ότι όλες οι διαδρομές μεταξύ D Flip-Flop περνούν από τουλάχιστον μία μονάδα CLB.

$$\begin{aligned} t_{\text{hold}} + t_{\text{skew}} &< t_{ccq} + t_{cd} \Rightarrow t_{\text{skew}} < t_{ccq} + t_{cd} - t_{\text{hold}} \Rightarrow \\ t_{\text{skew}} &< (0,50 + 0,30 - 0) \text{ ns} = 0,8 \text{ ns} = 800 \text{ ps} \end{aligned}$$

Παραλληλισμός: Βασικοί ορισμοί

- **Οντότητα δεδομένων:** μια ομάδα δεδομένων στην είσοδο ενός συστήματος, που υπόκεινται σε επεξεργασία εντός του συστήματος, ώστε να παραχθεί μια άλλη ομάδα δεδομένων στην έξοδο του συστήματος
- **Λανθάνων χρόνος (latency):** ο χρόνος που χρειάζεται η οντότητα δεδομένων για να διέλθει από το σύστημα, από την αρχή έως το τέλος (από τον πρώτο κύκλο ρολογιού μέχρι τον τελευταίο)
- **Διεκπεραιωτική ικανότητα (throughput):** το πλήθος των οντοτήτων δεδομένων που μπορούν να παράγονται ανά μονάδα χρόνου (π.χ. ανά sec)
- **Παραλληλισμός (parallelism):** η δυνατότητα του συστήματος να επεξεργάζεται πολλές οντότητες δεδομένων ταυτόχρονα
 - **Χωρικός παραλληλισμός:** πλεονασμός στο υλικό, ώστε να είναι εφικτή η ταυτόχρονη επεξεργασία πολλών οντοτήτων δεδομένων
 - **Χρονικός παραλληλισμός ή διοχέτευση (pipelining):** η επεξεργασία εντός του συστήματος χωρίζεται σε στάδια, όπως στη γραμμή παραγωγής
 - Σε κάθε δεδομένη στιγμή μια διαφορετική οντότητα δεδομένων θα βρίσκεται σε κάθε στάδιο, οπότε πολλές οντότητες δεδομένων, όσες και τα στάδια, υπόκεινται σε ταυτόχρονη επεξεργασία

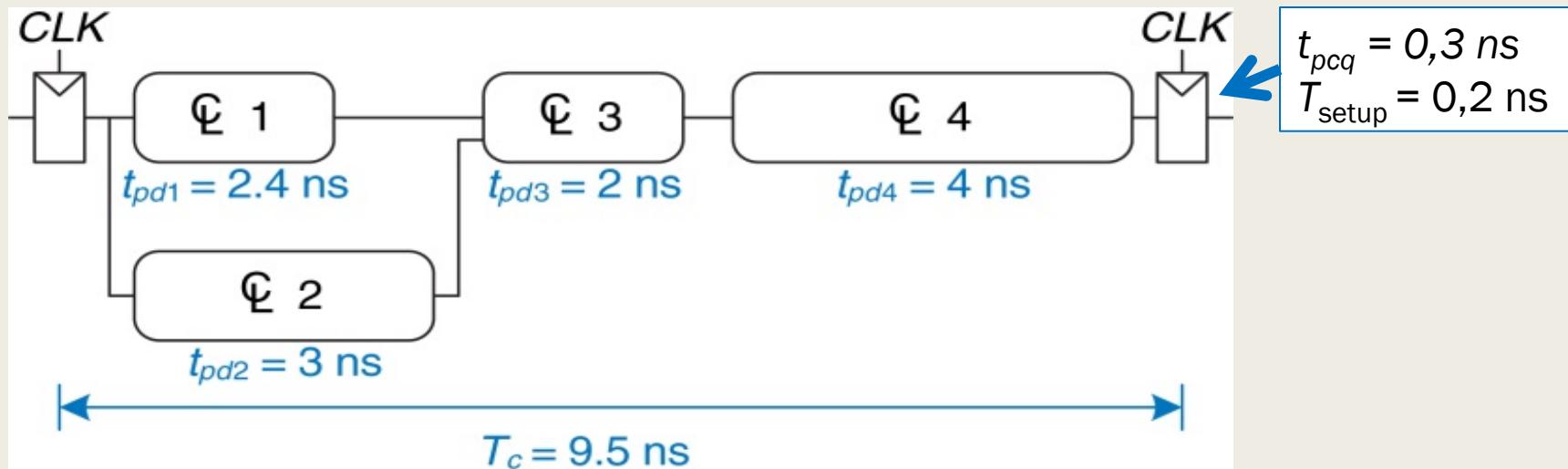
Διοχέτευση

- Η διοχέτευση είναι ιδιαίτερα ελκυστική επειδή αυξάνει την ταχύτητα ενός κυκλώματος χωρίς να απαιτεί πλεονασμό του υλικού.
- Αντ' αυτού, τοποθετούνται καταχωρητές ανάμεσα σε τμήματα συνδυαστικής λογικής, ώστε να διαιρεθεί η λογική σε μικρότερα στάδια τα οποία μπορούν να εκτελούνται με γρηγορότερο ρολόι
 - *Η περίοδος του ρολογιού εξαρτάται από το πιο αργό στάδιο*
- Οι καταχωρητές δεν επιτρέπουν σε μια οντότητα δεδομένων που βρίσκεται σε ένα στάδιο διοχέτευσης να «προφθάσει» και να αλλοιώσει την οντότητα δεδομένων που βρίσκεται στο επόμενο στάδιο

Διοχέτευση: παράδειγμα

■ Κύκλωμα χωρίς διοχέτευση

- Περιέχει τέσσερα τμήματα λογικής μεταξύ των καταχωρητών
 - Η κρίσιμη διαδρομή περνάει από τα τμήματα 2, 3 και 4



$$T_c \geq (t_{pcq} + t_{\text{setup}}) + t_{pd} = (0.5 + 2 + 3 + 4) \text{ ns} = 9.5 \text{ ns}$$

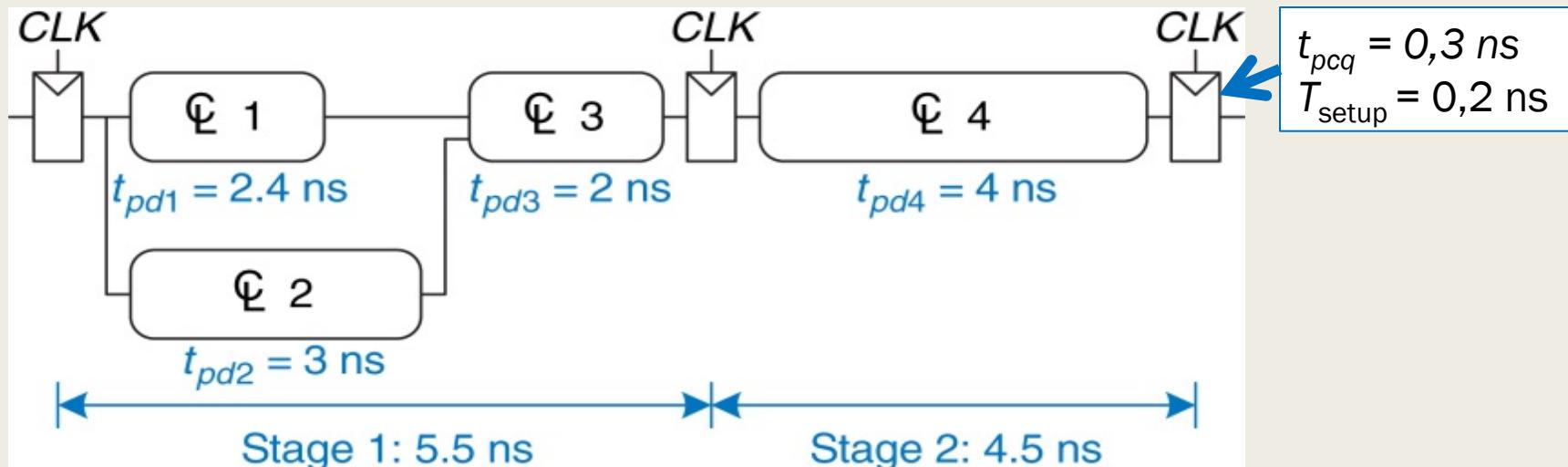
Επιβάρυνση δημιουργίας ακολουθίας
(sequencing overhead): $t_{pcq} + t_{\text{setup}} = 0.5 \text{ ns}$

Latency = 9,5 ns
Throughput = 1/9,5 ns = 105 MHz

Διοχέτευση: παράδειγμα

■ Κύκλωμα με διοχέτευση δύο σταδίων

- Προσθήκη επιπλέον καταχωρητή ανάμεσα στα τμήματα 3 και 4
 - Η κρίσιμη διαδρομή περνάει από τα τμήματα 2 και 3 (στάδιο 1)



$$T_c \geq (t_{pcq} + t_{\text{setup}}) + t_{pd} = (0,5 + 2 + 3) \text{ ns} = 5,5 \text{ ns}$$

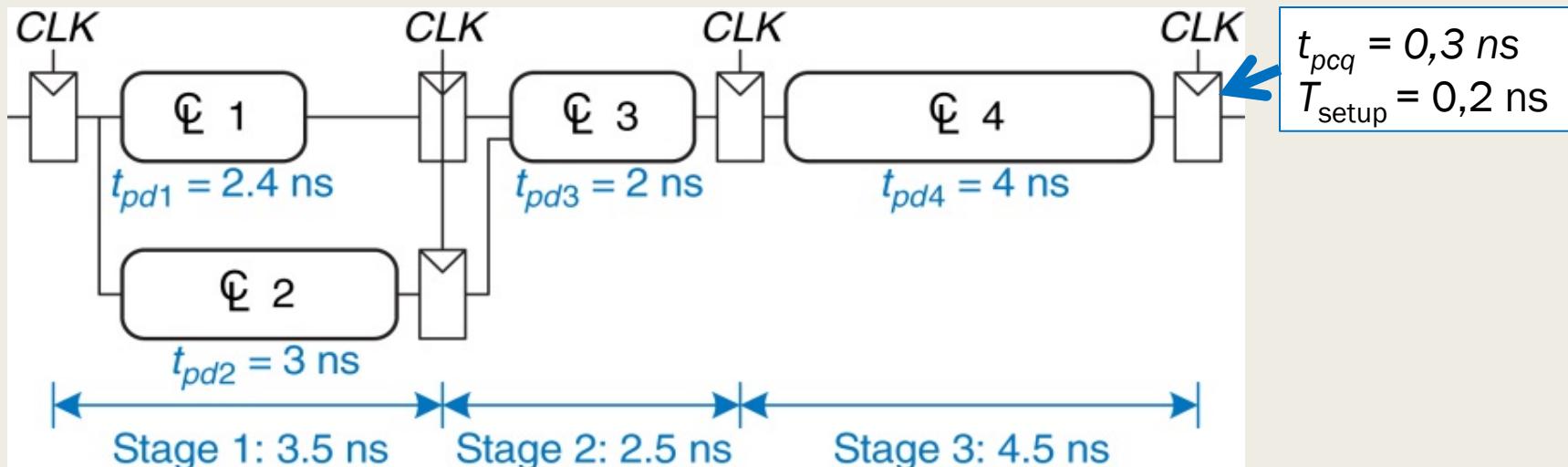
Επιβάρυνση δημιουργίας ακολουθίας
(sequencing overhead): $t_{pcq} + t_{\text{setup}} = 0,5 \text{ ns}$

Latency = $2 \times 5,5 \text{ ns} = 11 \text{ ns}$
Throughput = $1/5,5 \text{ ns} = 182 \text{ MHz}$

Διοχέτευση: παράδειγμα

■ Κύκλωμα με διοχέτευση τριών σταδίων

- Προσθήκη επιπλέον καταχωρητή ανάμεσα στα τμήματα 3 και 4
 - Η κρίσιμη διαδρομή περνάει από τα τμήματα 2 και 3 (στάδιο 1)



$$T_c \geq (t_{pcq} + t_{\text{setup}}) + t_{pd} = (0,5 + 4) \text{ ns} = 4,5 \text{ ns}$$

Επιβάρυνση δημιουργίας ακολουθίας
(sequencing overhead): $t_{pcq} + t_{\text{setup}} = 0,5 \text{ ns}$

Latency = $3 \times 4,5 \text{ ns} = 13,5 \text{ ns}$
Throughput = $1/4,5 \text{ ns} = 222 \text{ MHz}$

Διοχέτευση: συμπεράσματα

- Κύκλωμα χωρίς διοχέτευση

Latency = 9,5 ns

Throughput = 1/9,5 ns = 105 MHz

- Κύκλωμα με διοχέτευση δύο σταδίων

Latency = $2 \times 5,5$ ns = 11 ns

Throughput = 1/5,5 ns = 182 MHz

- Κύκλωμα με διοχέτευση τριών σταδίων

Latency = $3 \times 4,5$ ns = 13,5 ns

Throughput = 1/4,5 ns = 222 MHz

- Σε ένα πραγματικό κύκλωμα, η διοχέτευση με N στάδια πολλαπλασιάζει σχεδόν κατά N φορές τη διεκπεραιωτική ικανότητα (throughput) και αυξάνει ανεκτά τον λανθάνοντα χρόνο (latency).

- Η ιδανική διοχέτευση θα πολλαπλασίαζε ακριβώς κατά N φορές τη διεκπεραιωτική ικανότητα χωρίς να αύξανε τον λανθάνοντα χρόνο
- Η διαφορά που παρατηρείται οφείλεται στο γεγονός ότι:
 - το κύκλωμα δεν μπορεί να χωριστεί σε N στάδια, όπου όλα τα στάδια να έχουν ακριβώς την ίδια καθυστέρηση διάδοσης,
 - που να ισούται με το $1/N$ της συνολικής καθυστέρησης διάδοσης που είχε το κύκλωμα πριν την εφαρμογής της διοχέτευσης
 - οι καταχωρητές της διοχέτευσης εισάγουν πρόσθετη επιβάρυνση δημιουργίας ακολουθίας $N(t_{pcq} + t_{setup})$ (όταν $t_{skew} = 0$)

Κεφάλαιο 4

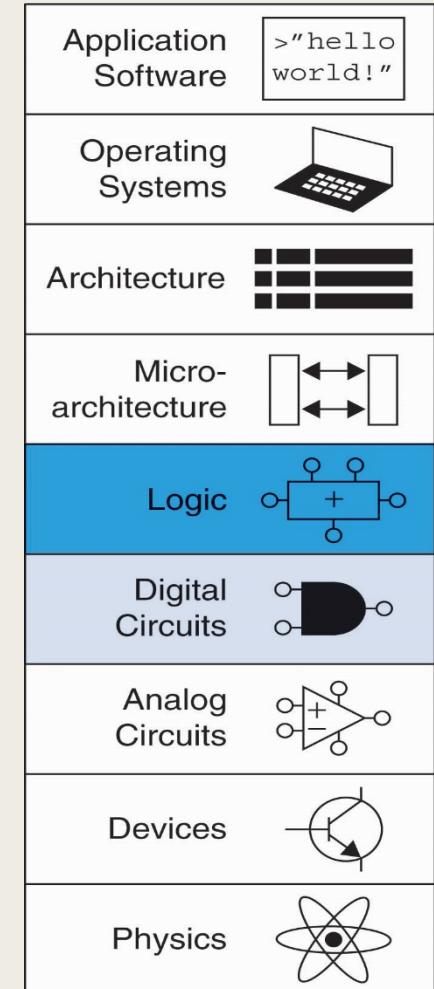
Γλώσσες περιγραφής υλικού

Γιώργος Παπαδημητρίου, Αντώνης Πασχάλης,
Βασιλόπουλος Διονύσης



Περιεχόμενα κεφαλαίου 4

- Τεχνολογία FPGA
- Φάσεις σχεδίασης υλικού
- Τα θεμέλια της γλώσσας VHDL
 - Σήματα και μεταβλητές τύπου *std_logic*
 - Περιγραφή δομής
 - Περιγραφή *dataflow*
 - Περιγραφή συμπεριφοράς
 - Η εντολή *IF*
 - Συνδυαστική λογική
 - Ακολουθιακή λογική
 - Ασύγχρονη
 - Σύγχρονη



Προγραμματιζόμενες από τον χρήστη διατάξεις πυλών (FPGA)

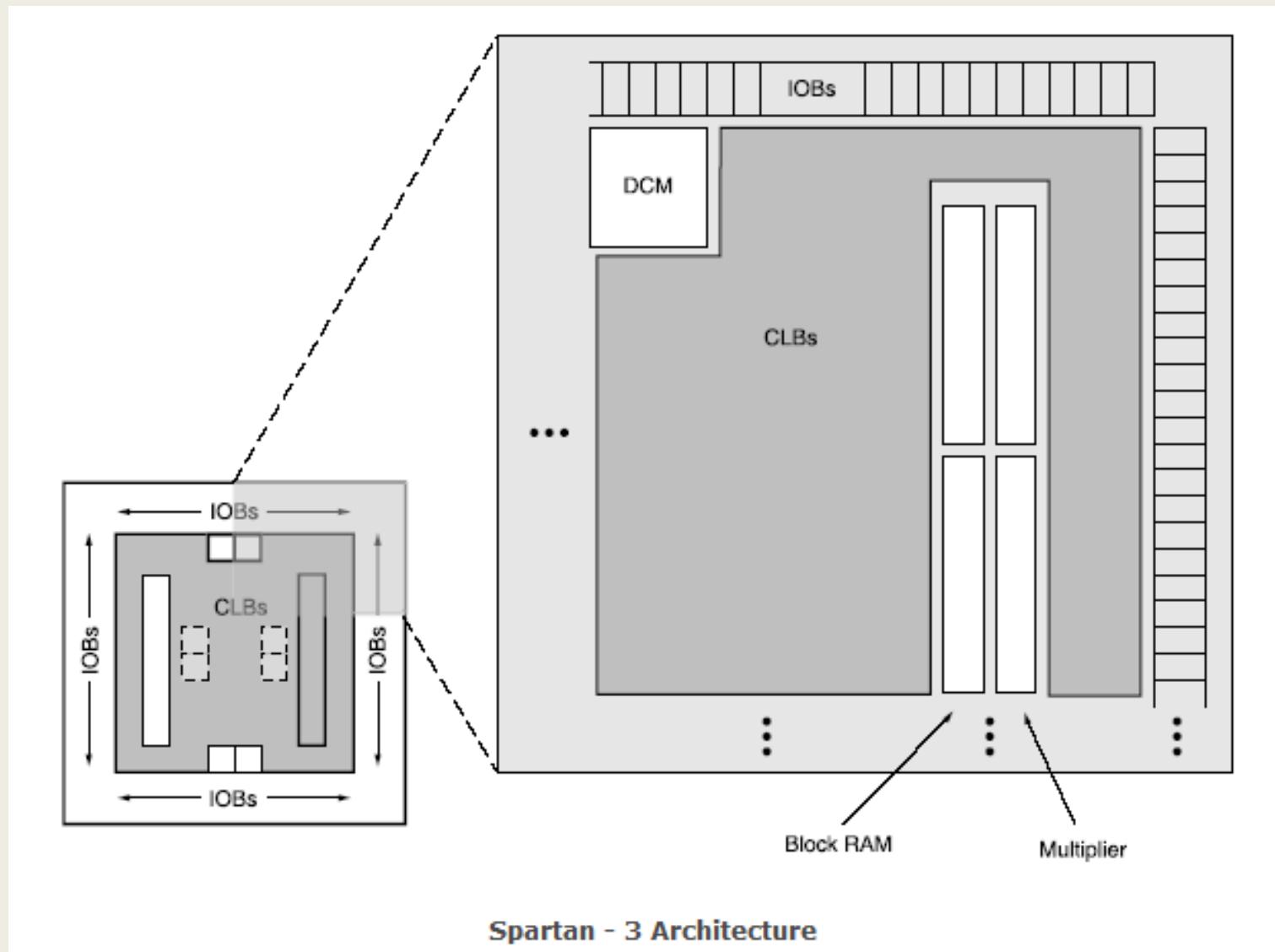
- Παρέχουν δυνατότητα σχεδίασης **VLSI κυκλωμάτων χαμηλού κόστους** στο πεδίο, με τη χρήση σχετικά φθηνών εργαλείων λογισμικού (CAD)
 - κόστος ανεκτό από μικρές εταιρείες που δραστηριοποιούνται στην ανάπτυξη επιταχυντών υλικού και γενικότερα πυρήνων IP
- Είναι **επαναπρογραμματιζόμενες** και **επαναδιατάξιμες** (ολικώς ή μερικώς) ακόμα και κατά τη διάρκεια της κανονικής λειτουργίας
 - Παρέχουν **μεγάλη ευελιξία** στη σχεδίαση ψηφιακών συστημάτων
- Διαθέτουν **ενσωματωμένες μνήμες, πολλαπλασιαστές, ειδικές μονάδες για ψηφιακή επεξεργασία σήματος, εισόδους/εξόδους υψηλών ταχυτήτων, μετατροπείς δεδομένων** (όπως ADC, DAC, RF), ακόμα και **πυρήνες επεξεργαστών** σε κάποιες περιπτώσεις
- Η γενικότερη τεχνολογική εξέλιξη σε θέματα κόστους, αποδόσεων, κατανάλωσης ισχύος και αξιοπιστίας έχει σαν αποτέλεσμα οι σύγχρονες διατάξεις FPGA να χρησιμοποιούνται ευρέως σε **εμπορικές, βιομηχανικές, αμυντικές και διαστημικές εφαρμογές**

Η αρχιτεκτονική των FPGAs της XILINX

- Παράδειγμα: Η αρχιτεκτονική του **Spartan III**
 - **Configurable Logic Blocks (CLBs)**
που περιέχουν πίνακες αναζήτησης (LUT), που υλοποιούν συνδυαστική λογική, και στοιχεία αποθήκευσης που μπορούν να χρησιμοποιηθούν ως D Flip-flops ή Latches
 - **Input/Output Blocks (IOBs)**
που ελέγχουν την ροή δεδομένων μεταξύ των I/O pins και της εσωτερικής λογικής
 - **Block RAMs**
που παρέχουν αποθηκευτικό χώρο μνήμης, μεγέθους για παράδειγμα 18Kbit (το μέγεθος εξαρτάται από την τεχνολογία)
 - **Multiplier Blocks**
σε πολλές οικογένειες οι πολλαπλασιαστές έχουν εξελιχθεί σε ειδικές μονάδες ψηφιακής επεξεργασίας σήματος
 - **Digital Clock Manager (DCM) Blocks**
που παρέχουν αυτορρυθμιζόμενες πλήρως ψηφιακές λύσεις για κατανομή, καθυστέρηση, διαίρεση και ρύθμιση της φάσης των ρολογιών
 - Τα blocks διασυνδέονται μέσω **προγραμματιζόμενων πινάκων διακοπτών**(switch matrices)

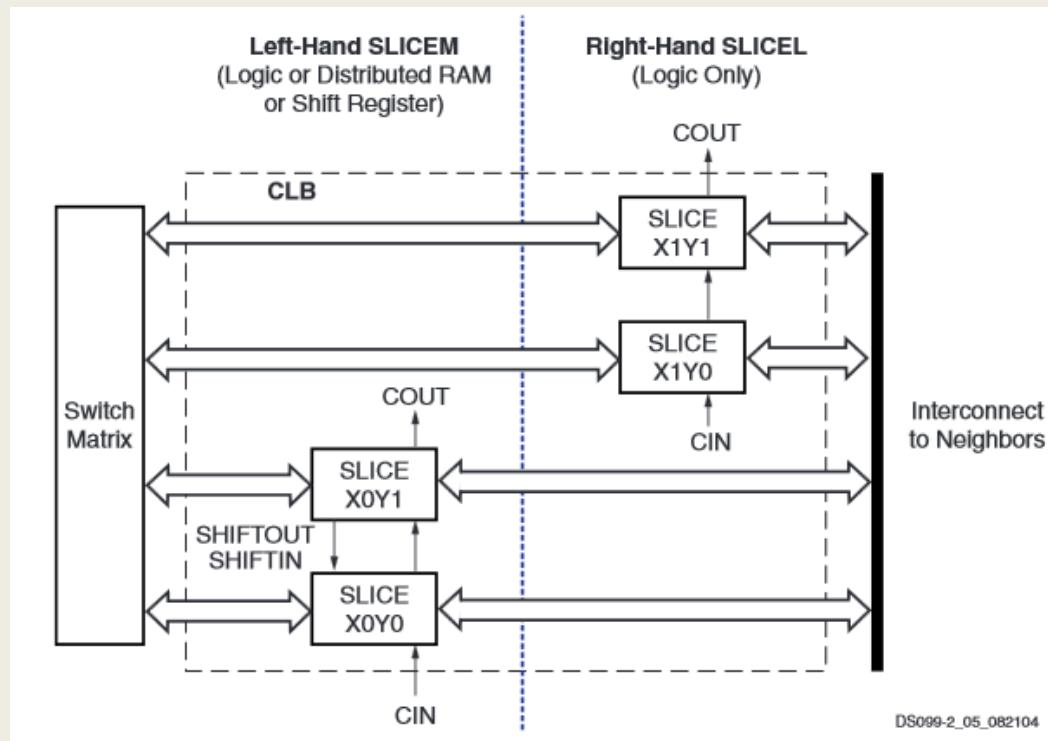
Η αρχιτεκτονική των FPGAs της XILINX

- Παράδειγμα: Η αρχιτεκτονική του Spartan III



Η αρχιτεκτονική των FPGAs της XILINX

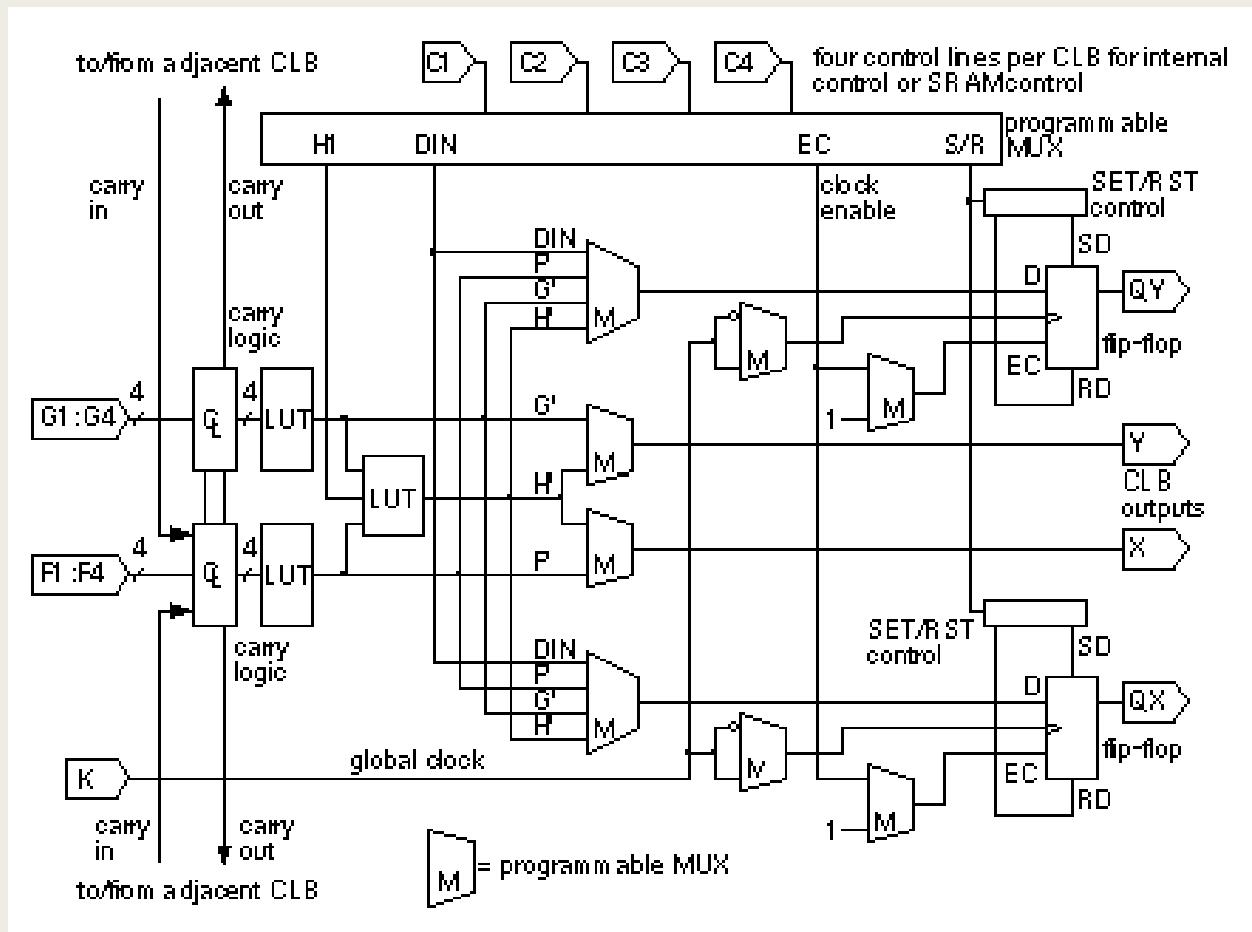
- Παράδειγμα: Η αρχιτεκτονική του **Spartan III**
 - Κάθε CLB αποτελείται από τέσσερα συνδεδεμένα *slices* τα οποία ομαδοποιούνται σε ζευγάρια και κάθε ζευγάρι σχηματίζει μια ανεξάρτητη αλυσίδα διάδοσης κρατουμένου.
 - Το αριστερό ζευγάρι υλοποιεί λογική, κατανεμημένη μνήμη ή καταχωρητή ολίσθησης
 - Το δεξιό ζευγάρι υλοποιεί αποκλειστικά λογική



Η αρχιτεκτονική των FPGAs της XILINX

■ Παράδειγμα: Η αρχιτεκτονική του Spartan III

- Κάθε slice του CLB έχει : 2 πίνακες αναζήτησης (LUT) των 4 εισόδων, 2 στοιχεία αποθήκευσης, πολυπλέκτες, αλυσίδα διάδοσης κρατούμενου και πύλες για αριθμητική λογική

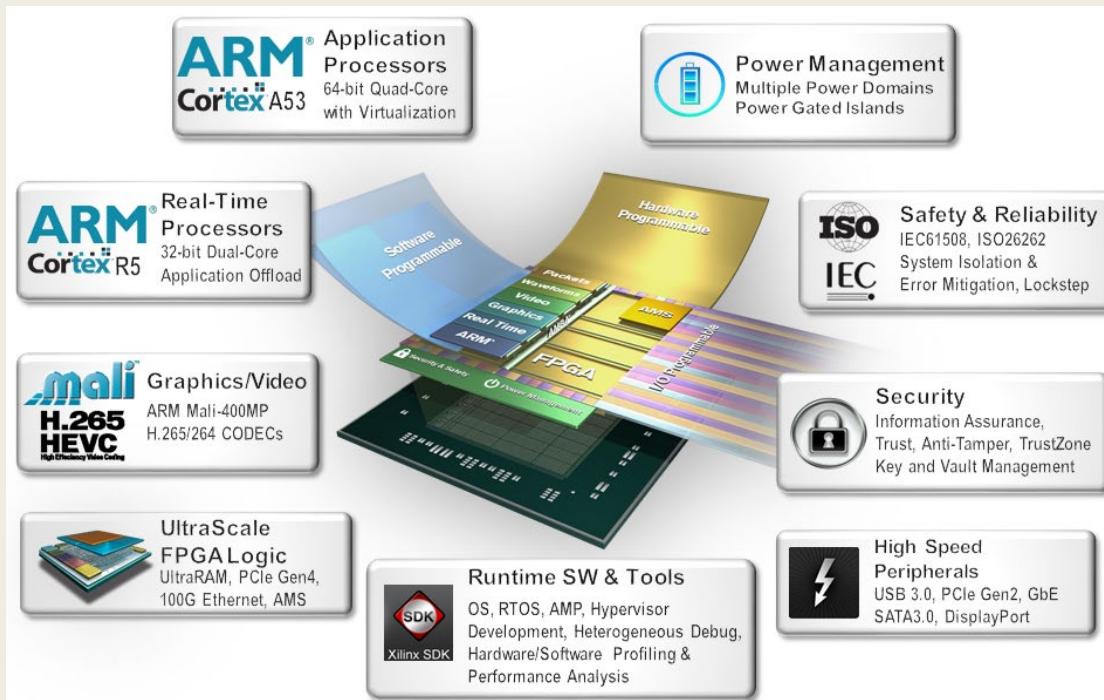


Επιταχυντές Υλικού για ψηφιακά συστήματα υψηλής απόδοσης σε τεχνολογία FPGA

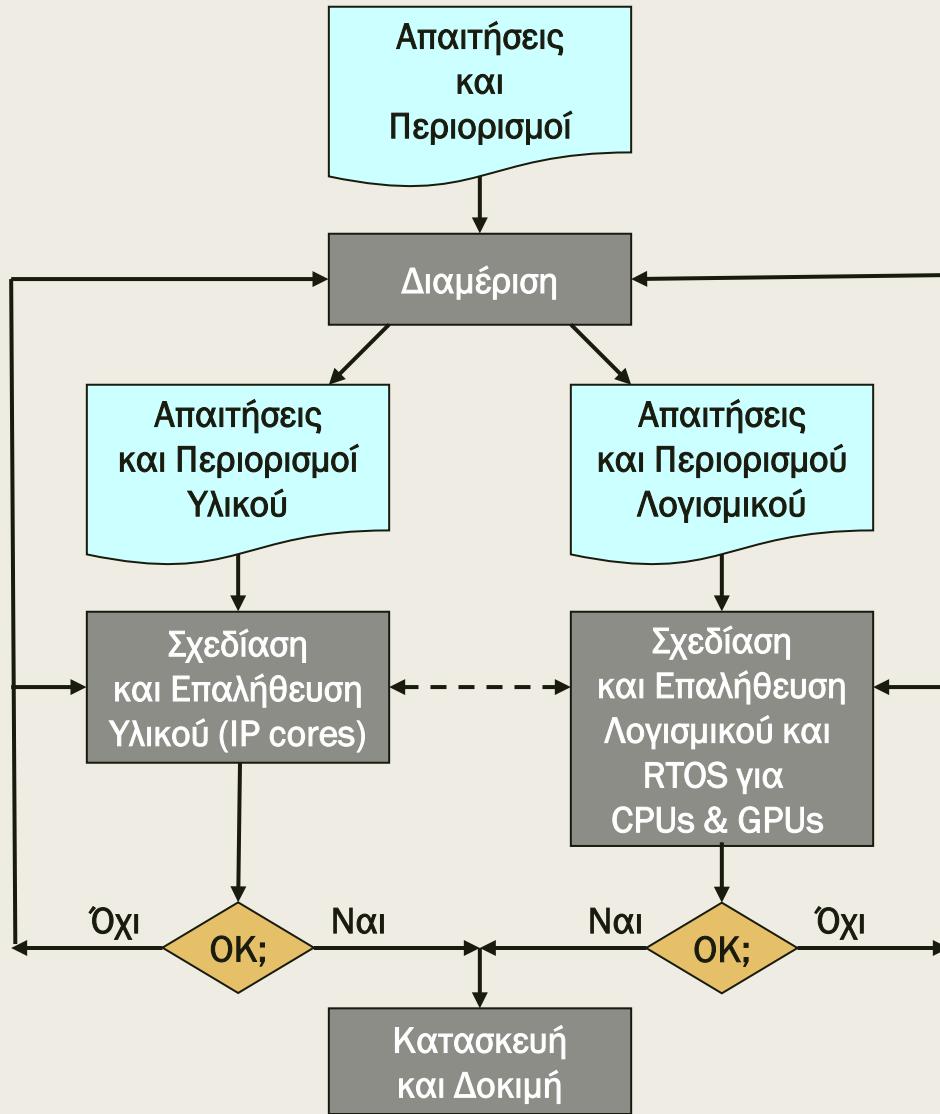
- Κανόνας 90/10
 - Συχνά, το **90% του χρόνου εκτέλεσης** και της κατανάλωσης ισχύος ενός προγράμματος δαπανάται **από το 10% του κώδικα**
 - Μικρά κομμάτια μιας εφαρμογής αποτελούν το **bottleneck της απόδοσης**
 - Αφορούν κυρίως επεξεργασία δεδομένων χωρίς πολύπλοκο έλεγχο (dataflow processing), όπως επεξεργασία και συμπίεση εικόνας 3D και βίντεο, κρυπτογραφία, μηχανική μάθηση, κλπ.
 - Οι **επεξεργαστές γραφικών (GPUs)** επιταχύνουν σημαντικά το κρίσιμο μέρος της εφαρμογής που υλοποιεί αλγορίθμους με διανυσματικές πράξεις που εκτελούνται παράλληλα χωρίς εξαρτήσεις δεδομένων
 - Οι **επιταχυντές υλικού** (ως **πυρήνες IP σε προγραμματιζόμενη λογική**) επιταχύνουν σημαντικά το κρίσιμο μέρος της εφαρμογής που υλοποιεί σύνθετους αλγορίθμους με εξαρτήσεις δεδομένων
 - Οι **πυρήνες επεξεργαστών (CPUs)** υλοποιούν τα λιγότερο κρίσιμα μέρη με τεχνικές παράλληλης επεξεργασίας

Επιταχυντές Υλικού για ψηφιακά συστήματα υψηλής απόδοσης σε τεχνολογία FPGA

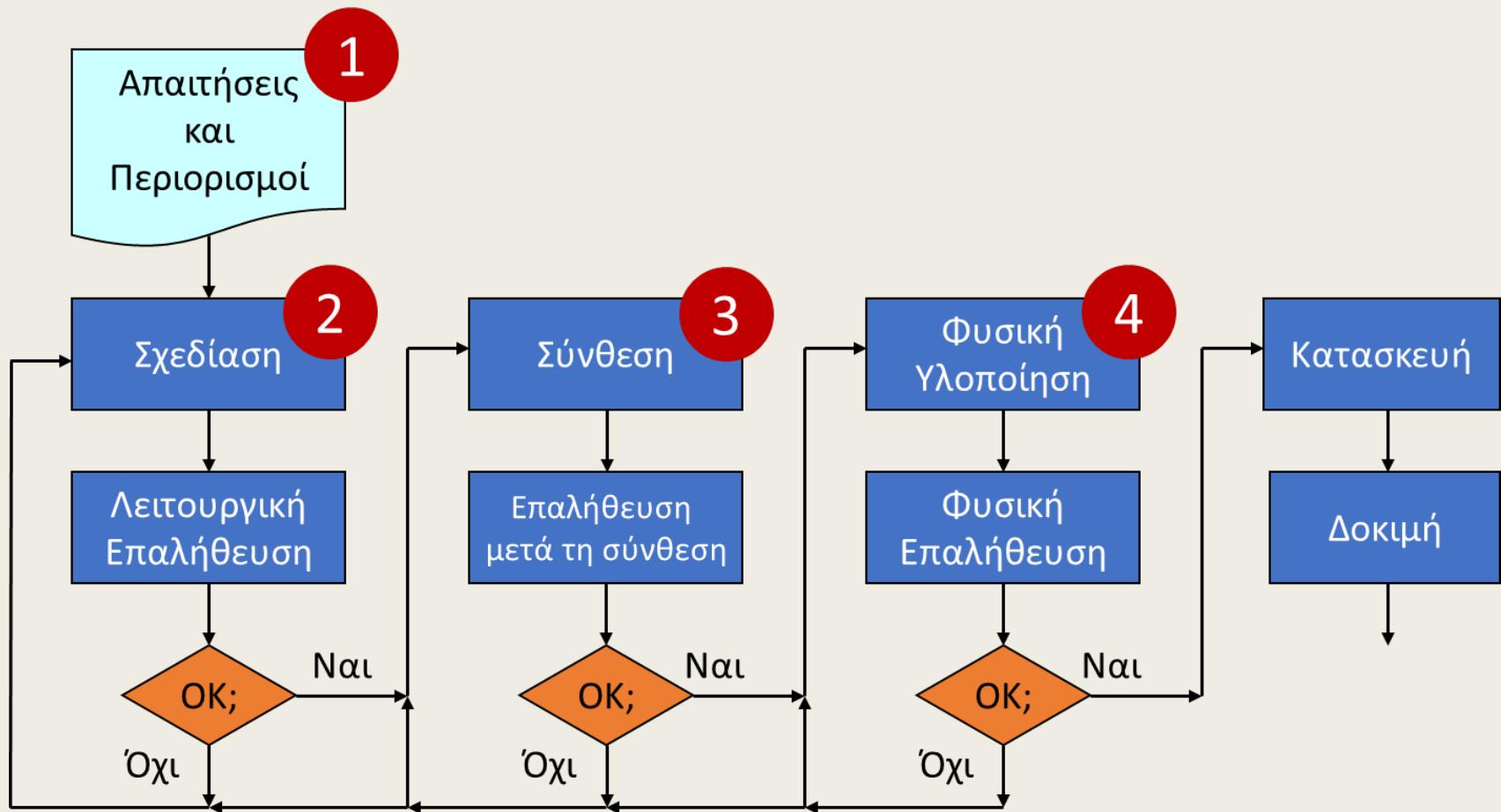
- Η τεχνολογική εξέλιξη που αλλάζει την κλασσική προσέγγιση του **υλικού** και τα όρια της **συ-σχεδίασης υλικού-λογισμικού**
 - προγραμματιζόμενη λογική (για πυρήνες IP) + μνήμες + επεξεργαστές ARM + επεξεργαστές γραφικών + έτοιμοι επιταχυντές υλικού + συνδεσιμότητα υψηλής ταχύτητας σε ένα **ΤΣΙΠ (Multi-Processing System on Chip)**!
 - Το ενσωματωμένο λογισμικό μπορεί να εκτελεστεί από την SRAM στο FPGA
 - Λύση ενός ΤΣΙΠ μειωμένου κόστους που αποφεύγει το υψηλό NRE του ASIC



Συ-σχεδίαση υλικού-λογισμικού στα σύγχρονα ψηφιακά συστήματα



Φάσεις σχεδίασης υλικού



Φάσεις σχεδίασης υλικού

- Σχεδίαση επιταχυντών υλικού σε τεχνολογία FPGA με χρήση εργαλείων CAD (computer-aided design)
 - Προσδιορισμός απαιτήσεων και περιορισμοί
 - Εισαγωγή σχεδίασης (*design entry*)
 - Προγραμματισμός σε γλώσσα περιγραφής υλικού (HDL), αντί για σχηματικά διαγράμματα
 - Λειτουργική επαλήθευση της σχεδίασης με προσομοίωση (simulation) και τυπικές μεθόδους (formal methods)
 - Σύνθεση (*synthesis*)
 - Αυτόματη παραγωγή ιεραρχικού σχηματικού διαγράμματος και gate-level netlist
 - Επαλήθευση μετά τη σύνθεση με προσομοίωση (λειτουργική και χρονική)
 - Φυσική υλοποίηση (*implementation*)
 - Υλοποίηση στην τεχνολογία FPGA (διαδικασίες map, place και route)
 - Φυσική επαλήθευση (λειτουργική και χρονική)
 - Ικανοποίηση απαιτήσεων και περιορισμών
 - Προγραμματισμός FPGA και δοκιμή (κατασκευή)

Προσδιορισμός απαιτήσεων και περιορισμοί

- **Ανάλυση των απαιτήσεων και καθορισμός των προδιαγραφών**
 - λειτουργία, εσωτερικές και εξωτερικές διασυνδέσεις
 - περιβαλλοντολογικές απαιτήσεις (θερμοκρασία, ακτινοβολία, κλπ.)
 - απαιτήσεις επίδοσης και λειτουργίας σε πραγματικό χρόνο
 - απαιτήσεις κατανάλωσης ισχύος και ενέργειας
 - απαιτήσεις ποιότητας προϊόντος (φερεγγυότητα, ασφάλεια, QA)
- **Προκαταρτική σχεδίαση σε υψηλό επίπεδο**
 - περιγραφή σε υψηλό επίπεδο διαγραμμάτων με μπλοκ
 - προσδιορισμός λειτουργικών μονάδων και ιεραρχίας
 - προσδιορισμός διασυνδέσεων (είσοδοι, έξοδοι)
 - αρχιτεκτονική περιγραφή συνήθως σε γλώσσα υψηλού επιπέδου (π.χ. C-like, SystemC) και δημιουργία ενός *golden model*

Εισαγωγή σχεδίασης

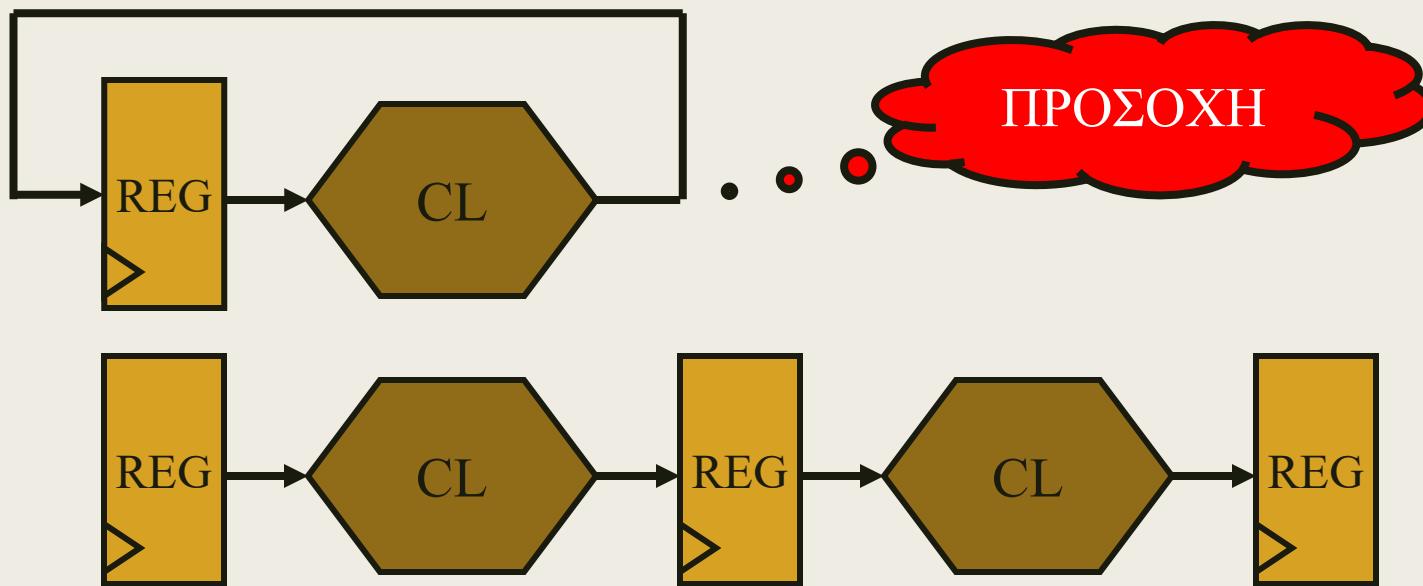
- Προγραμματισμός σε **γλώσσα περιγραφής υλικού (Hardware Description Language – HDL)**, αντί για σχηματικά διαγράμματα
 - Υπερτερεί σε σύγκριση με τα σχηματικά διαγράμματα
 - Η περιγραφή σε HDL γίνεται πιο κατανοητή από ένα σχηματικό διάγραμμα, λόγω καλλίτερης διαχείρισης της πολυπλοκότητας
 - Η μοντελοποίηση του συστήματος μπορεί να γίνει σε όλα τα επίπεδα (από τα υψηλότερα ως τα χαμηλότερα) με κατάλληλη αφαίρεση από επίπεδο σε επίπεδο
 - Η περιγραφή σε HDL είναι ανεξάρτητη (?) από τις βιβλιοθήκες σχεδίασης (design libraries) και τα εργαλεία CAD
 - Υπερτερεί σε σύγκριση με τις γλώσσες προγραμματισμού (SW)
 - Παρέχει δομές που είναι συνθέσιμες και περιγράφουν καλύτερα το υλικό, όπως τις μνήμες
 - Υποστηρίζει την παράλληλη εκτέλεση εντολών (δεν περιορίζεται στην ακολουθιακή εκτέλεση εντολών)
 - Παρέχει τη δυνατότητα για περιγραφή χρονισμών
 - Αλλά θέλει προσοχή!

Εισαγωγή σχεδίασης

- Προγραμματισμός σε **γλώσσα περιγραφής υλικού (Hardware Description Language – HDL)**, αντί για σχηματικά διαγράμματα
 - Άλλα θέλει προσοχή!
 - Οι γλώσσες HDL μαθαίνονται εύκολα, αλλά εφαρμόζονται σωστά δύσκολα
 - Οι προγραμματιστές τείνουν να γράφουν κώδικα HDL που μοιάζει με τα προγράμματα λογισμικού με χρήση πολλών μεταβλητών και πολλών βρόχων που μπορεί να μην είναι συνθέσιμα ή να συντίθενται με μη ικανοποιητικό αποτέλεσμα
 - Πάντα έχουμε στο μυαλό μας το ψηφιακό κύκλωμα που αντιστοιχεί στον κώδικα HDL που γράφουμε
 - Διαχειρίζομαστε την πολυπλοκότητα με κατάλληλη αφαίρεση και εφαρμόζοντας την ιεραρχία, την τμηματικότητα και την κανονικότητα
 - Η περιγραφή γίνεται πάντα στο επίπεδο μεταφοράς καταχωρητή, (register transfer level – RTL), ώστε να είναι συνθέσιμη
 - Υψηλότερο επίπεδο αφαίρεσης από τις πύλες

Περιγραφή Ψηφιακού Συστήματος σε Επίπεδο Μεταφοράς Καταχωρητή - RTL

- Περιγράφεται κάθε καταχωρητής (REG) του συστήματος, καθώς και η συνδυαστική λογική (CL) ανάμεσα στους καταχωρητές



Γλώσσες περιγραφής υλικού (HDL)

- Κατά τη δεκαετία του 1990 οι σχεδιαστές ανακάλυψαν ότι ήταν πολύ πιο παραγωγικό να δουλεύουν σε ένα υψηλότερο επίπεδο αφαίρεσης από τις πύλες αφήνοντας το έργο της ελαχιστοποίησης των πυλών σε ένα εργαλείο CAD
- Οι δύο κορυφαίες γλώσσες περιγραφής υλικού είναι:
 - *SystemVerilog, για εμπορικές εφαρμογές (C-like)*
 - *VHDL, για στρατιωτικές και διαστημικές εφαρμογές (ADA-like)*που βασίζονται σε παρόμοιες αρχές, αλλά έχουν διαφορετική σύνταξη
- Η VHDL είναι περισσότερο αναλυτική (απαιτεί περισσότερο κώδικα) και είναι πιο πολύπλοκη, αλλά και πιο ακριβής από τη SystemVerilog,
 - όπως θα περιμένατε ίσως από μια γλώσσα που έχει αναπτυχθεί από κάποια ειδική επιτροπή για αμυντικές και διαστημικές εφαρμογές

SystemVerilog

- Η Verilog αναπτύχθηκε το 1984 από την εταιρεία Gateway Design Automation, αρχικά, για την επαλήθευση λογικής με προσομοίωση
- Το 1989 η εταιρεία Gateway αγοράστηκε από την εταιρεία Cadence, και το 1990 η Verilog μετατράπηκε σε ανοικτό πρότυπο υπό την εποπτεία του οργανισμού Open Verilog International
- Το 1995 η γλώσσα έγινε πρότυπο (standard) του Ινστιτούτου IEEE
- Το 2005 η γλώσσα επεκτάθηκε, με απώτερο σκοπό τη βελτιστοποίηση κάποιων εκκεντρικών χαρακτηριστικών και την καλύτερη υποστήριξη για τη μοντελοποίηση, σύνθεση και επαλήθευση
- Οι συγκεκριμένες επεκτάσεις έχουν συγχωνευθεί σε ένα γλωσσικό πρότυπο, το οποίο πλέον ονομάζεται SystemVerilog (IEEE STD 1800-2009).
- Τα ονόματα αρχείων της SystemVerilog συνήθως έχουν προέκταση .sv

VHDL

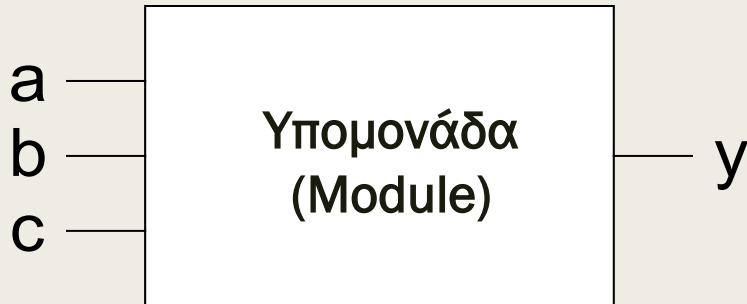
- Το αρκτικόλεξο VHDL προέρχεται από τα αρχικά της έκφρασης:

Very High Speed Integrated Circuits
Hardware Description L

- Δημιουργήθηκε αρχικά στα πλαίσια του ερευνητικού προγράμματος Very High Speed Integrated Circuits, που χρηματοδότησε το Υπουργείο Αμύνης των ΗΠΑ (στις αρχές του 1980) με σκοπό την περιγραφή της δομής και της λειτουργίας του υλικού, που παραλαμβάνεται από πολλούς διαφορετικούς κατασκευαστές
- Αν και αρχικά επινοήθηκε για σκοπούς τεκμηρίωσης, γρήγορα χρησιμοποιήθηκε για την **περιγραφή**, τη **μοντελοποίηση**, την **επαλήθευση λογικής με προσομοίωση** και τη **σύνθεση** των ψηφιακών συστημάτων
- Τυποποιήθηκε από το Ινστιτούτου IEEE, αρχική έκδοση 1987, τελική έκδοση 1993, επεκτάσεις στην έκδοση 2008 (IEEE STD 1076-2008)
 - Ελέγχουμε, εάν υποστηρίζεται από το εργαλείο CAD η έκδοση 2008
- Τα ονόματα αρχείων της VHDL συνήθως έχουν προέκταση.vhd

Υπομονάδα (Module)

- Ένα τμήμα υλικού με εισόδους και εξόδους ονομάζεται **υπομονάδα**

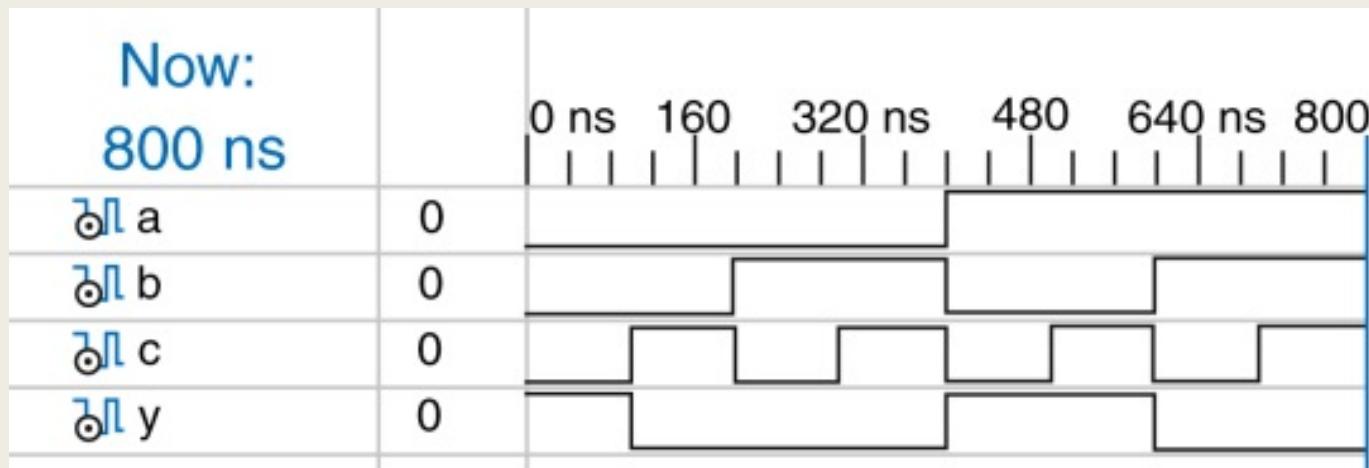


- Η λειτουργικότητα των υπομονάδων περιγράφεται με δύο τρόπους:
 - ο πρώτος τρόπος βασίζεται στην **περιγραφή της συμπεριφοράς** της υπομονάδας
 - ο άλλος τρόπος βασίζεται στην **περιγραφή της δομής** της υπομονάδας.
- Αντίστοιχα, προκύπτουν δύο διακριτά **μοντέλα περιγραφής της λειτουργικότητας** των υπομονάδων:
 - τα **μοντέλα περιγραφής συμπεριφοράς** (*behavioral models*) που περιγράφουν τι κάνει μια υπομονάδα,
 - και τα **μοντέλα περιγραφής δομής** (*structural models*), που περιγράφουν πώς κατασκευάζεται η υπομονάδα από απλούστερα στοιχεία

Προσομοίωση

- Παρακάτω φαίνεται η κυματομορφή που προέκυψε από μία προσομοίωση της υπομονάδας που υλοποιεί την εξίσωση Boole:

$$Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$$



- Εξετάζοντας όλες τις κυματορμοφές καταλήγουμε ότι η υπομονάδα όντως λειτουργεί σωστά
- Η έξοδος Y έχει την τιμή TRUE όταν οι είσοδοι A, B και C είναι 000, 100 ή 101, óπως δηλαδή ορίζει η εξίσωση Boole

Σφάλματα

- Τα λάθη κατά τη σχεδίαση υλικού ονομάζονται **σφάλματα (bugs)**
- Η διαδικασία της **δοκιμής (testing)** που χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας ενός συστήματος είναι χρονοβόρα
- Ο **εντοπισμός της αιτίας των λαθών** κατά τη δοκιμή ενός συστήματος μπορεί να αποδειχθεί **εξαιρετικά δύσκολος**, επειδή μπορούν να παρατηρηθούν μόνο σήματα που δρομολογούνται στους ακροδέκτες του τσιπ
 - Για να παρατηρήσει κανείς απευθείας **τι συμβαίνει μέσα στο τσιπ** πρέπει να ενσωματώσει ένα *Logic Analysis Core*
- Η **προσομοίωση της λογικής** είναι απολύτως απαραίτητη για την **επαλήθευση της ορθής σχεδίασης και την αποσφαλμάτωση** ενός ψηφιακού συστήματος προτού αυτό κατασκευαστεί
 - Στη συνέχεια το ψηφιακό σύστημα **υλοποιείται σε τεχνολογία FPGA**

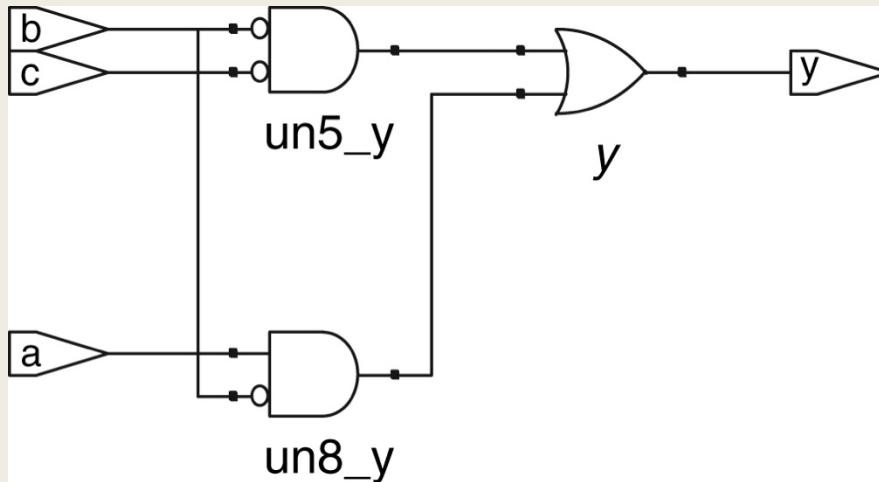
Σύνθεση

- Η σύνθεση της λογικής μετασχηματίζει τον κώδικα HDL σε μια **λίστα συνδέσεων στο επίπεδο της λογικής πύλης** (gate-level netlist)
 - *Περιγράφει το υλικό (τις λογικές πύλες και τα σύρματα που τις συνδέουν)*
- Το εργαλείο σύνθεσης λογικής (logic synthesizer) μπορεί να προχωρήσει σε **βελτιστοποιήσεις** για
 - *Να μειώσει την ποσότητα του απαιτούμενου υλικού*
 - *Να αυξήσει τη συχνότητα λειτουργίας*
- Η λίστα συνδέσεων μπορεί να έχει τη μορφή αρχείου κειμένου ή μπορεί να σχεδιάζεται σαν ένα **σχηματικό διάγραμμα** ώστε να διευκολύνεται η οπτικοποίηση του ψηφιακού κυκλώματος

Σύνθεση

- Στο παρακάτω σχήμα φαίνεται το **αποτέλεσμα της σύνθεσης** της λογικής για την υπομονάδα που υλοποιεί την εξίσωση Boole:

$$Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$$



- Οι τρεις πύλες AND, με τρεις εισόδους η καθεμία, ελαχιστοποιούνται σε δύο πύλες AND, με δύο εισόδους η καθεμία
- $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C = \bar{B}\bar{C} + A\bar{B}$

Πρόγραμμα Δοκιμών

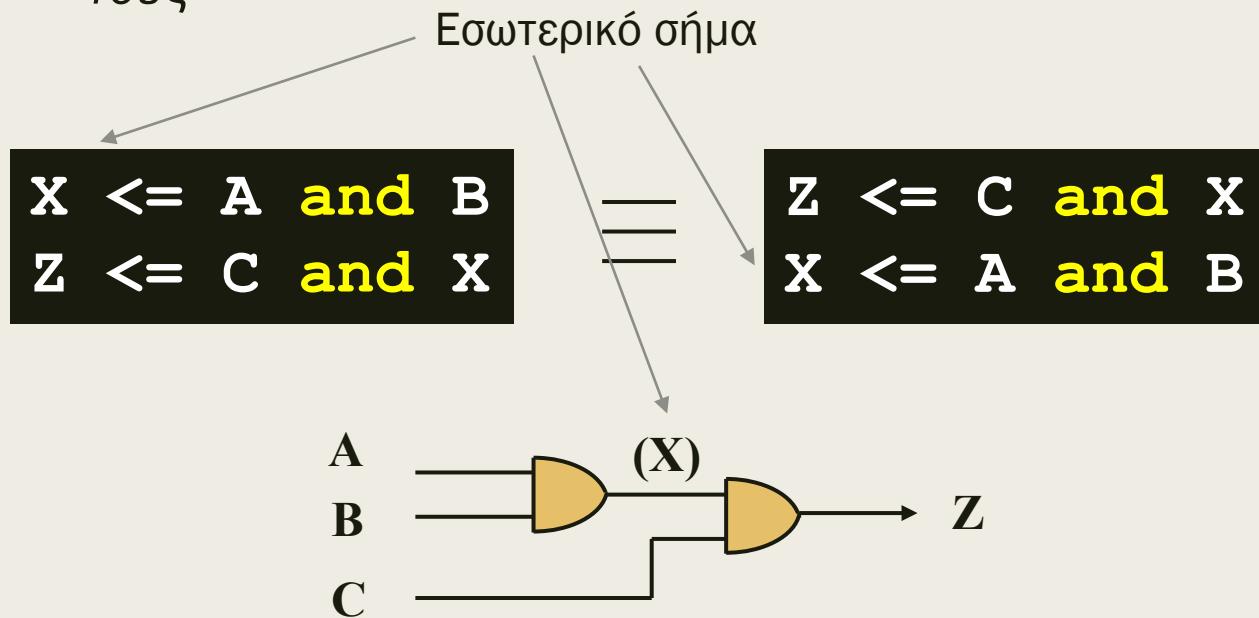
- Οι περιγραφές κυκλωμάτων σε γλώσσες περιγραφής υλικού HDL **μοιάζουν** με τον κώδικα των γλωσσών προγραμματισμού
- Δεν είναι όμως εφικτή η σύνθεση όλων των εντολών των γλωσσών HDL σε υλικό
 - Για παράδειγμα, μια εντολή που τυπώνει αποτελέσματα στην οθόνη κατά τη διάρκεια της προσομοίωσης δεν «μεταφράζεται» σε υλικό
- Για την προσομοίωση των υπομονάδων χρησιμοποιούμε τα **προγράμματα δοκιμών** (testbench)
 - Περιέχουν τον κώδικα σε HDL με τον οποίο τροφοδοτούμε τις εισόδους μιας υπομονάδας, ώστε να ελέγξουμε αν τα αποτελέσματα στις εξόδους είναι σωστά, και θα τυπώσουμε τυχόν εμφανιζόμενες διαφορές στις κυματομορφές (αναμενόμενες έναντι πραγματικές) που προκύπτουν κατά την προσομοίωση
 - Ο κώδικας ενός προγράμματος δοκιμών **προορίζεται μόνο για προσομοίωση** και **δεν είναι συνθέσιμος**

Ιδιωματισμοί

- Ο καλύτερος τρόπος για να μάθετε μια γλώσσα περιγραφής υλικού είναι μέσα από **παραδείγματα κωδίκων HDL**
- Οι γλώσσες περιγραφής υλικού διαθέτουν συγκεκριμένους τρόπους με τους οποίους περιγράφουν διάφορα είδη λογικής που ονομάζονται **ιδιωματισμοί** (idioms)
- Όταν πρέπει να περιγράψετε ένα συγκεκριμένο είδος λογικής, αναζητήστε κάποιο παρόμοιο παράδειγμα και προσαρμόστε το στις δικές σας ανάγκες
- Οι ιδιωματισμοί που θα παραθέσουμε στη συνέχεια είναι σε γλώσσα VHDL και στοχεύουν στη **σωστή σύνθεση**, είναι συνθέσιμοι **σε όλα τα εργαλεία CAD** και είναι **ανεξάρτητοι της τεχνολογίας υλοποίησης** (technology agnostic)
 - Χρησιμοποιούμε ένα **μικρό υποσύνολο** των εντολών της γλώσσας VHDL

Ταυτόχρονες εντολές στη γλώσσα VHDL

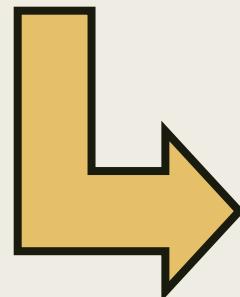
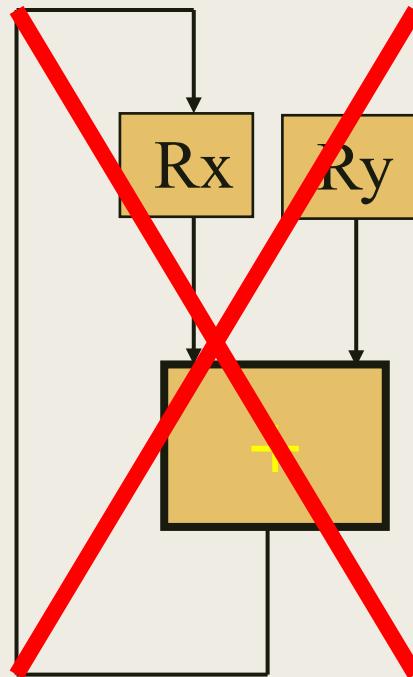
- **Ταυτόχρονες** εντολές (concurrent statements)
 - εκτελούνται στον ίδιο χρόνο παράλληλα
 - η συμπεριφορά τους είναι **ανεξάρτητη** από τη σειρά εμφάνισής τους



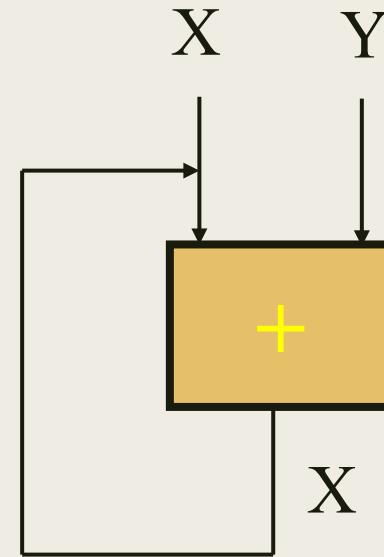
Η φύση του υλικού (hardware) απαιτεί την υποστήριξη ταυτόχρονων εντολών

Ταυτόχρονες εντολές – Προσοχή!

$$X \leq X + Y$$



ανάδραση



Δεν είναι όπως στο λογισμικό (software)

Ακολουθιακές εντολές στη γλώσσα VHDL

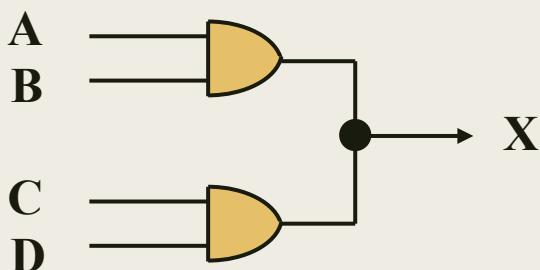
■ Ακολουθιακές εντολές (sequential statements)

- εμφανίζονται μέσα σε **δομή διεργασίας (process)** για να ξεχωρίζουν από τις ταυτόχρονες εντολές
- εκτελούνται **μόνο μία φορά** στη σειρά
- η συμπεριφορά τους εξαρτάται από τη σειρά εμφάνισής τους
- αλγοριθμική περιγραφή, όπως στο λογισμικό

Ταυτόχρονες έναντι ακολουθιακών εντολών

Ταυτόχρονες εντολές

```
X <= A and B  
X <= C and D
```

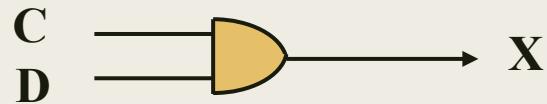


Av A=B=C=D='0' => X='0'
Av A=B=C=D='1' => X='1'
Av A=B='1', C=D='0' => X='X'

Ακολουθιακές εντολές
μέσα σε δομή process

```
X <= A and B  
X <= C and D
```

αγνοείται



Η οντότητα (entity) στη γλώσσα VHDL

- Περιγράφει τη διασύνδεση μίας ιεραρχικής υπομονάδας, **χωρίς να προσδιορίζει τη συμπεριφορά της** σαν μαύρο κουτί (black box)
- Η διασύνδεση της υπομονάδας περιγράφεται με μία δήλωση των διαύλων (ports - signals)

```
entity entity_name is -- σχόλια
  port (
    signal_name: mode signal_type;
    signal_name: mode signal_type;
    ...
    signal_name: mode signal_type);
end entity_name;
```

Η οντότητα (entity) στη γλώσσα VHDL

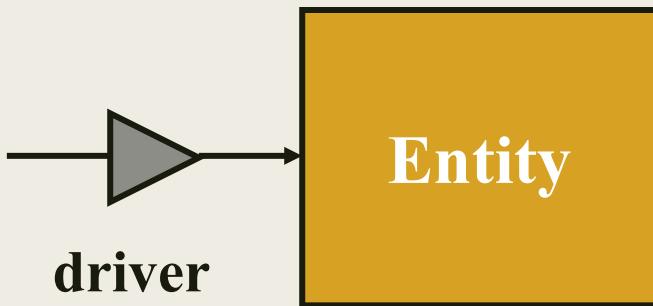
- **entity_name**: το όνομα της οντότητας
- **signal_name**: το όνομα του σήματος
(εάν είναι πολλά σήματα χωρίζονται με κόμμα)
- **mode**: η κατεύθυνση του **driver** του σήματος
 - **in**: είσοδος της οντότητας
 - **out**: έξοδος της οντότητας
 - **inout**: είσοδος ή έξοδος της οντότητας (*bidirectional*),
- **signal_type**: ο τύπος του σήματος (STD_LOGIC)

Ονόματα και ετικέτες στη γλώσσα VHDL

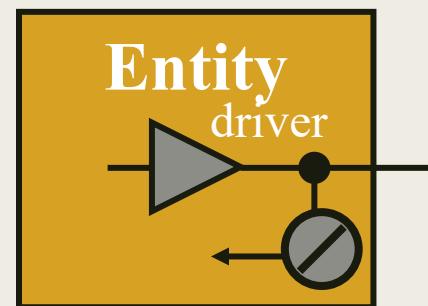
- Είναι **μοναδικά** μέσα σε μία συγκεκριμένη οντότητα (και αρχιτεκτονική)
- Χρησιμοποιούνται οι χαρακτήρες: **a-z, A-Z, 0-9, " "**
- Δεν χρησιμοποιούνται οι χαρακτήρες, όπως: **+, -, !, &**
- Δεν χρησιμοποιούνται ούτε **σημεία στίξης** στα ονόματα και τις ετικέτες, ούτε διπλό **"_**, δηλαδή **"__"**
- Δεν διαχωρίζονται κεφαλαία γράμματα από μικρά
- Ο πρώτος χαρακτήρας είναι **αλφαριθμητικός**
- Το μέγεθος περιορίζεται συνήθως στους **32 χαρακτήρες**
- **Προσοχή στις δεσμευμένες λέξεις!**
- **Δεν παίζουν ρόλο τα κενά και τα carriage returns**
 - **Η εντολή τελειώνει με ";"**
- **Τα σχόλια σε μία γραμμή έπονται του διπλού "-"**

Σημασία του mode στο port signal

Mode **in**

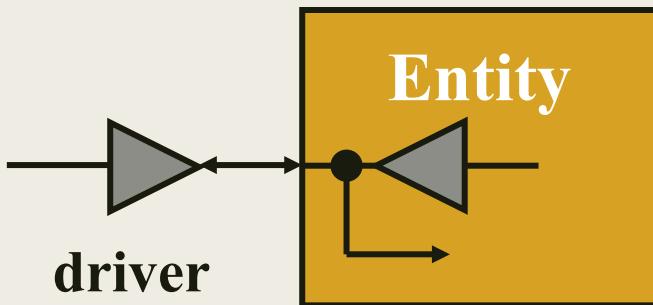


Mode **out**

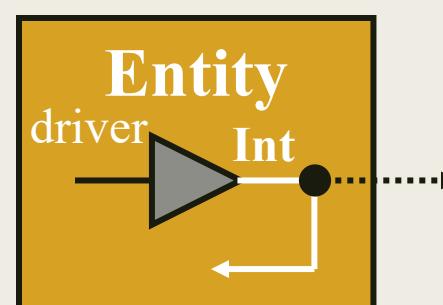


Προσοχή! Τα σήματα εξόδου δεν χρησιμοποιούνται σαν είσοδοι εντός της οντότητας

Mode **inout**



Mode **out**



Απαιτείται χρήση εσωτερικού σήματος (int) που χρησιμοποιείται σαν είσοδος εντός της οντότητας και συνδέεται με σήμα εξόδου

Σημασία του signal_type στο port signal

Σημασία τιμών στα σήματα τύπου std_logic		
Τιμή	Modeling for simulation	Synthesis
U	Uninitialized	Uninitialized
X	Strong driven unknown	Don't care
0	Strong driven 0	0
1	Strong driven 1	1
Z	High impedance	High impedance
W	Weakly driven unknown	Don't care
L	Weakly driven 0	0
H	Weakly driven 1	1
-	Don't care	Don't care

Ο τύπος του σήματος STD_LOGIC είναι μέρος του πακέτου IEEE.std_logic_1164 της βιβλιοθήκης IEEE. Για να χρησιμοποιηθεί όλο το πακέτο δηλώνουμε:

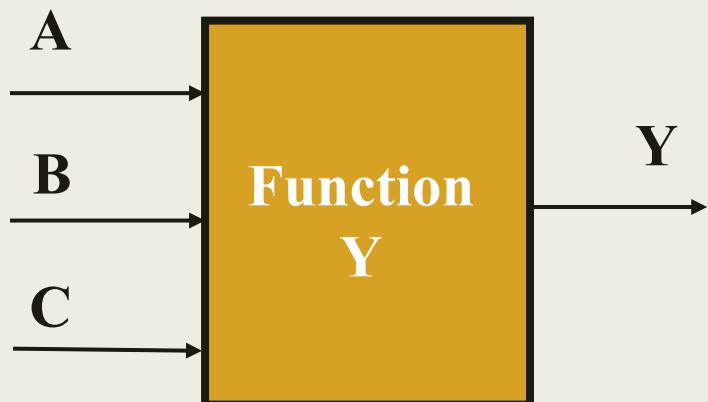
```
library IEEE;  
use IEEE.std_logic_1164.all;
```

Οντότητα μίας υπομονάδας στη γλώσσα VHDL

- Παρακάτω φαίνεται η οντότητα της υπομονάδας με όνομα Function_Y που υλοποιεί την εξίσωση Boole:

$$Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$$

```
entity Function_Y is
  port (
    A: in STD_LOGIC;
    B: in STD_LOGIC;
    C: in STD_LOGIC;
    Y: out STD_LOGIC);
end Function_Y;
```



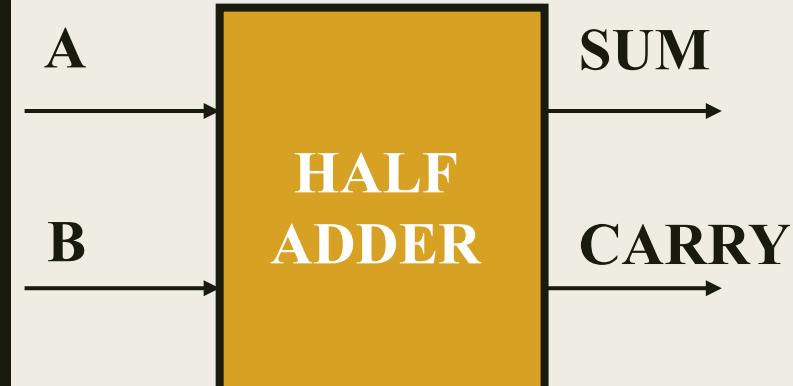
Η οντότητα του ημιαθροιστή στη VHDL

- Παρακάτω φαίνεται η οντότητα του ημιαθροιστή που υλοποιεί τις εξισώσεις Boole:

$$\text{SUM} = A \oplus B$$

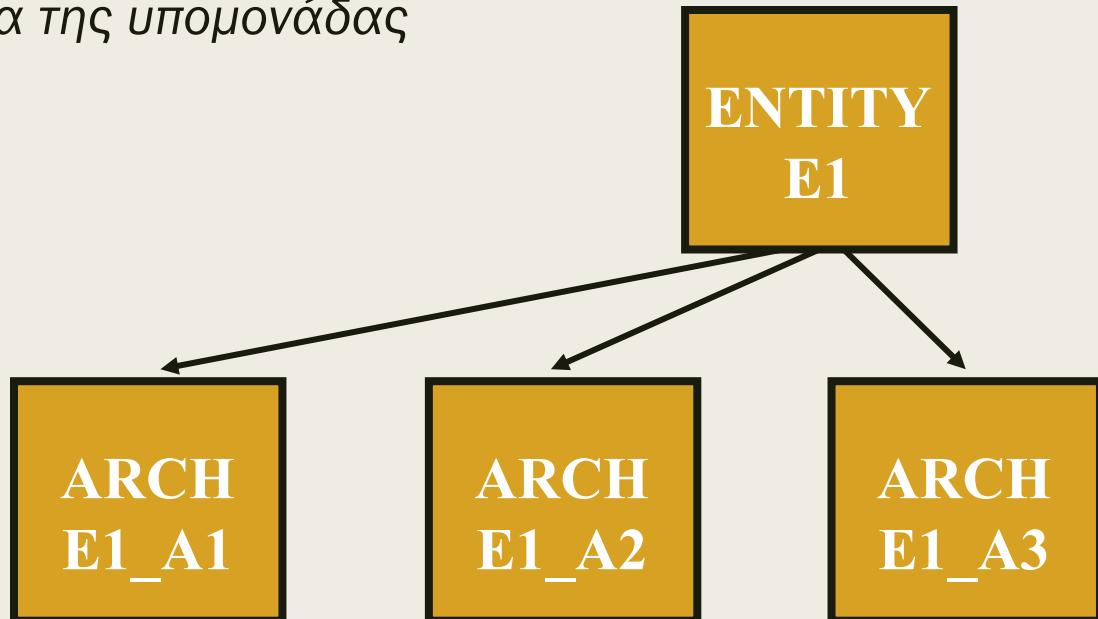
$$\text{CARRY} = AB$$

```
entity HALF_ADDER is
  port (
    A: in STD_LOGIC;
    B: in STD_LOGIC;
    SUM: out STD_LOGIC;
    CARRY: out STD_LOGIC) ;
end HALF_ADDER;
```



Οντότητα και αρχιτεκτονική στη γλώσσα VHDL

- Στη γλώσσα VHDL υπάρχουν **δύο κύρια στοιχεία** που περιγράφουν μια υπομονάδα (module)
 - **Η οντότητα**, η οποία περιγράφει τη διασύνδεση της υπομονάδας (είσοδοι, έξοδοι), και
 - **Η αρχιτεκτονική**, η οποία περιγράφει τη λειτουργία της υπομονάδας



Η αρχιτεκτονική (architecture) στη VHDL

- Περιγράφει τη λειτουργία μίας υπομονάδας με έναν από τους ακόλουθους τρόπους:
 - ένα σύνολο από **ταυτόχρονες εντολές ανάθεσης** (concurrent assignment statements) για **περιγραφή dataflow**
 - ένα σύνολο από **ακολουθιακές εντολές ανάθεσης** (sequential assignment statements) μέσα σε **δομές διεργασίας (process)** για **περιγραφή συμπεριφοράς (behavioral)**
 - ένα σύνολο από **διασυνδεδεμένα στοιχεία (components)** για **περιγραφή δομής (structural)**, όπως γίνεται σε μία σχεδίαση με σχηματικό διάγραμμα
 - κάθε συνδυασμός από τα πιο πάνω

Περιγραφή dataflow στη VHDL

```
architecture arch_name of entity_name is
    signal signal_name: signal_type;
begin
    concurrent_statement;
    ...
    concurrent_statement;
end arch_name;
```

- **arch_name:** το όνομα της αρχιτεκτονικής
- **entity_name:** το όνομα της οντότητας
- **signal_name:** το όνομα του σήματος
(εάν είναι πολλά σήματα χωρίζονται με κόμμα)
 - στις δηλώσεις σημάτων (μετά το *signal*) το σήμα είναι **μία εσωτερική διασύνδεση** της υπομονάδας
- **signal_type:** ο τύπος του σήματος (STD_LOGIC)

Περιγραφή dataflow στη VHDL

- **Ταυτόχρονες εντολές ανάθεσης σήματος**
(concurrent_signal_assignment_statements)

```
signal_name <= expression;
```

- **<=:** τελεστής ανάθεσης τιμής σε σήμα
- **expression:** έκφραση με σήματα και τελεστές
- **signal_name:** το όνομα του σήματος
 - στις ταυτόχρονες εντολές ανάθεσης σήματος :
 - στην έκφραση προσδιορίζονται σήματα που είναι **είσοδοι** στην υπομονάδα και δηλώνονται κατά τη δήλωση των διαύλων της οντότητας, και **εσωτερικές διασυνδέσεις** της υπομονάδας που δηλώνονται κατά τη δήλωση σημάτων
 - στο αριστερό μέρος της εντολής προσδιορίζεται σήμα που είναι **έξοδος** της υπομονάδας και δηλώνεται κατά τη δήλωση των διαύλων της οντότητας, ή **εσωτερική διασύνδεση** της υπομονάδας που δηλώνεται κατά τη δήλωση σημάτων

Περιγραφή dataflow στη VHDL

- **Εκτέλεση** ταυτόχρονων εντολών ανάθεσης σήματος
(concurrent_signal_assignment_statements)

```
signal_name <= expression;
```

- Οι ταυτόχρονες εντολές ανάθεσης σήματος **εκτελούνται μόνο όταν υπάρξει αλλαγή τιμής στις εισόδους** (στα σήματα της δεξιάς πλευράς της ταυτόχρονης εντολής ανάθεσης σήματος).
- Δεν προσδιορίζεται καθυστέρηση διάδοσης άλλη, εκτός από μία απειροελάχιστη καθυστέρηση διάδοσης, την **καθυστέρηση δέλτα δ_{delay}** , που δεν επηρεάζει τον χρονισμό του κυκλώματος
- Η πραγματική καθυστέρηση διάδοσης θα προσδιορισθεί με την υλοποίηση σε μία συγκεκριμένη τεχνολογία

Η **καθυστέρηση δέλτα δ_{delay}** δεν είναι πραγματική καθυστέρηση που επηρεάζει την προσομοίωση, αλλά απλώς ιεραρχεί τις μεταβάσεις που συμβαίνουν στα σήματα την ίδια χρονική στιγμή.

Λογικοί τελεστές και προτεραιότητα λογικών πράξεων στη VHDL

a and b

$$a \cdot b$$



a or b

$$a + b$$



a nand b

$$\overline{a \cdot b}$$



a nor b

$$\overline{a + b}$$



a xor b

$$a \oplus b$$



a xnor b

$$\overline{a \oplus b}$$



not a

$$\overline{a}$$



■ Προτεραιότητα

- το **not** έχει την υψηλότερη προτεραιότητα
- οι υπόλοιποι τελεστές έχουν **ίση προτεραιότητα**
- οι λογικές πράξεις εκτελούνται **από αριστερά προς τα δεξιά**
- χρησιμοποιούμε **παρενθέσεις** για να ξεκαθαρίσουμε τη σειρά εκτέλεσης των λογικών πράξεων

■ Τιμές bit στην VHDL

- '0' και '1'

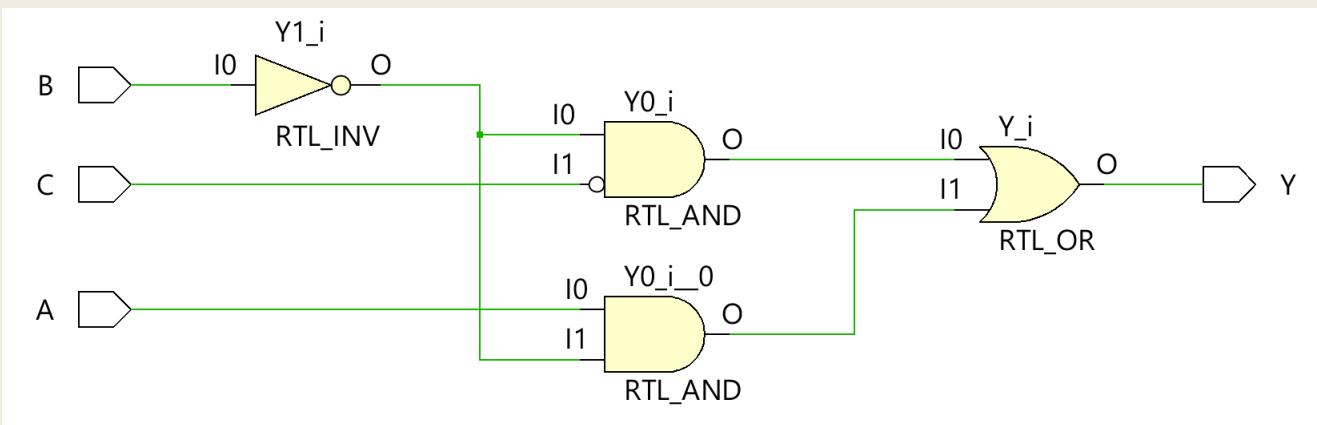
Η αρχιτεκτονική μίας υπομονάδας στη VHDL Περιγραφή dataflow

- Παρακάτω φαίνεται η αρχιτεκτονική της υπομονάδας σε περιγραφή dataflow που υλοποιεί την εξίσωση Boole:

$$- \quad Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C = \overline{B}\overline{C} + A\overline{B}$$

```
architecture Y_DATAFLOW of Function_Y is
begin
    Y <= ((not B) and (not C)) or
          (A and (not B));
end Y_DATAFLOW;
```

- Σχηματικό διάγραμμα RTL



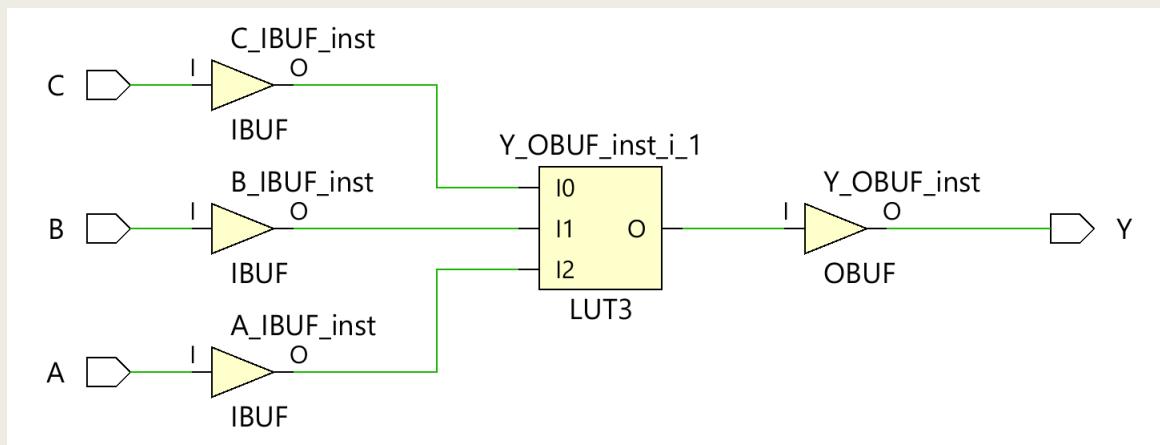
Η αρχιτεκτονική μίας υπομονάδας στη VHDL Περιγραφή dataflow

- Παρακάτω φαίνεται η αρχιτεκτονική της υπομονάδας σε περιγραφή dataflow που υλοποιεί την εξίσωση Boole:

$$- \quad Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C = \overline{B}\overline{C} + A\overline{B}$$

```
architecture Y_DATAFLOW of Function_Y is
begin
    Y <= ((not B) and (not C)) or
          (A and (not B));
end Y_DATAFLOW;
```

- Σχηματικό διάγραμμα σε τεχνολογία FPGA



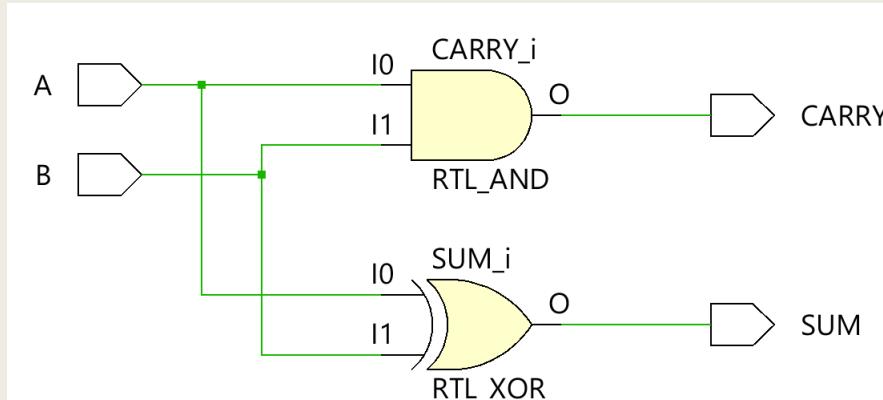
Η αρχιτεκτονική του ημιαθροιστή στη VHDL

Περιγραφή dataflow

- Παρακάτω φαίνεται η αρχιτεκτονική του ημιαθροιστή σε περιγραφή dataflow που υλοποιεί τις εξισώσεις Boole:
 - **SUM = A \oplus B, CARRY = AB**

```
architecture HA_DATAFLOW of HALF_ADDER is
begin
    SUM <= A xor B;
    CARRY <= A and B;
end HA_DATAFLOW;
```

- Σχηματικό διάγραμμα RTL



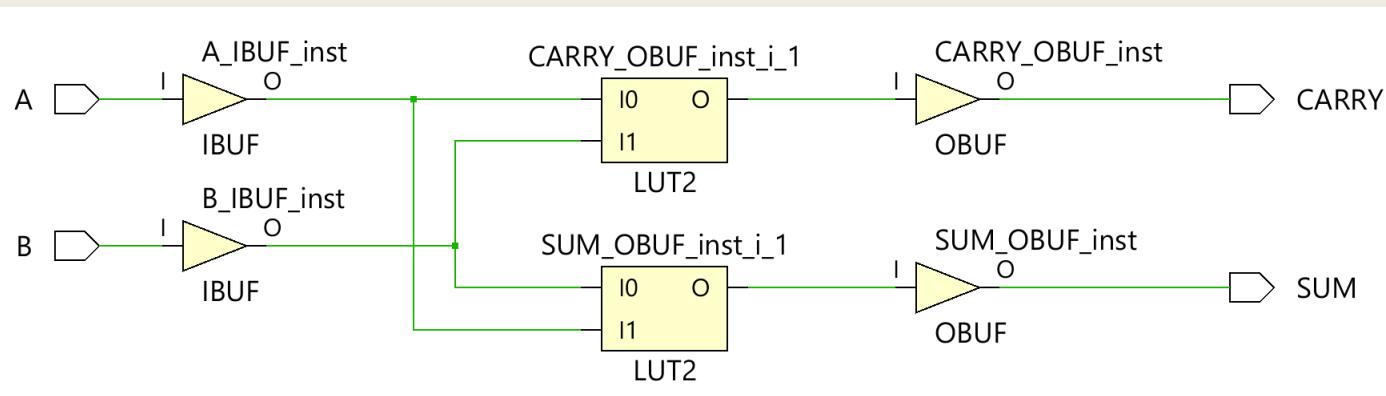
Η αρχιτεκτονική του ημιαθροιστή στη VHDL

Περιγραφή dataflow

- Παρακάτω φαίνεται η αρχιτεκτονική του ημιαθροιστή σε περιγραφή dataflow που υλοποιεί τις εξισώσεις Boole:
 - **SUM = A ⊕ B, CARRY = AB**

```
architecture HA_DATAFLOW of HALF_ADDER is
begin
    SUM <= A xor B;
    CARRY <= A and B;
end HA_DATAFLOW;
```

- Σχηματικό διάγραμμα σε τεχνολογία FPGA



Η αρχιτεκτονική του ημιαθροιστή στη VHDL

Περιγραφή dataflow για προγράμματα δοκιμής

```
architecture HA_DATAFLOW of HALF_ADDER is
begin
    SUM <= A xor B after 10 ns;
    CARRY <= A and B after 5 ns;
end HA_DATAFLOW;
```

Στα προγράμματα δοκιμής μπορεί να προσδιορισθεί για τις ανάγκες τις προσομοίωσης και καθυστέρηση διάδοσης με τη φράση **after**. Ο χρόνος αρχίζει να μετράει με την αλλαγή της τιμής σε μία από τις εισόδους. Το **after** αγνοείται κατά τη σύνθεση.

Περιγραφή συμπεριφοράς (behavioral) στη VHDL

```
architecture arch_name of entity_name is
begin
    label: process (signal_name, ..., signal_name)
        variable variable_name: variable_type;
    begin
        sequential_statement;
        ...
        sequential_statement;
    end process;
end arch_name;
```

Με την περιγραφή συμπεριφοράς προσδιορίζουμε τη λειτουργικότητα και όχι τη δομή της υπομονάδας.

Αν και οι εντολές μέσα σε μία διεργασία (process) αξιολογούνται ακολουθιακά, μετά τη σύνθεση προκύπτει ταυτόχρονη λογική

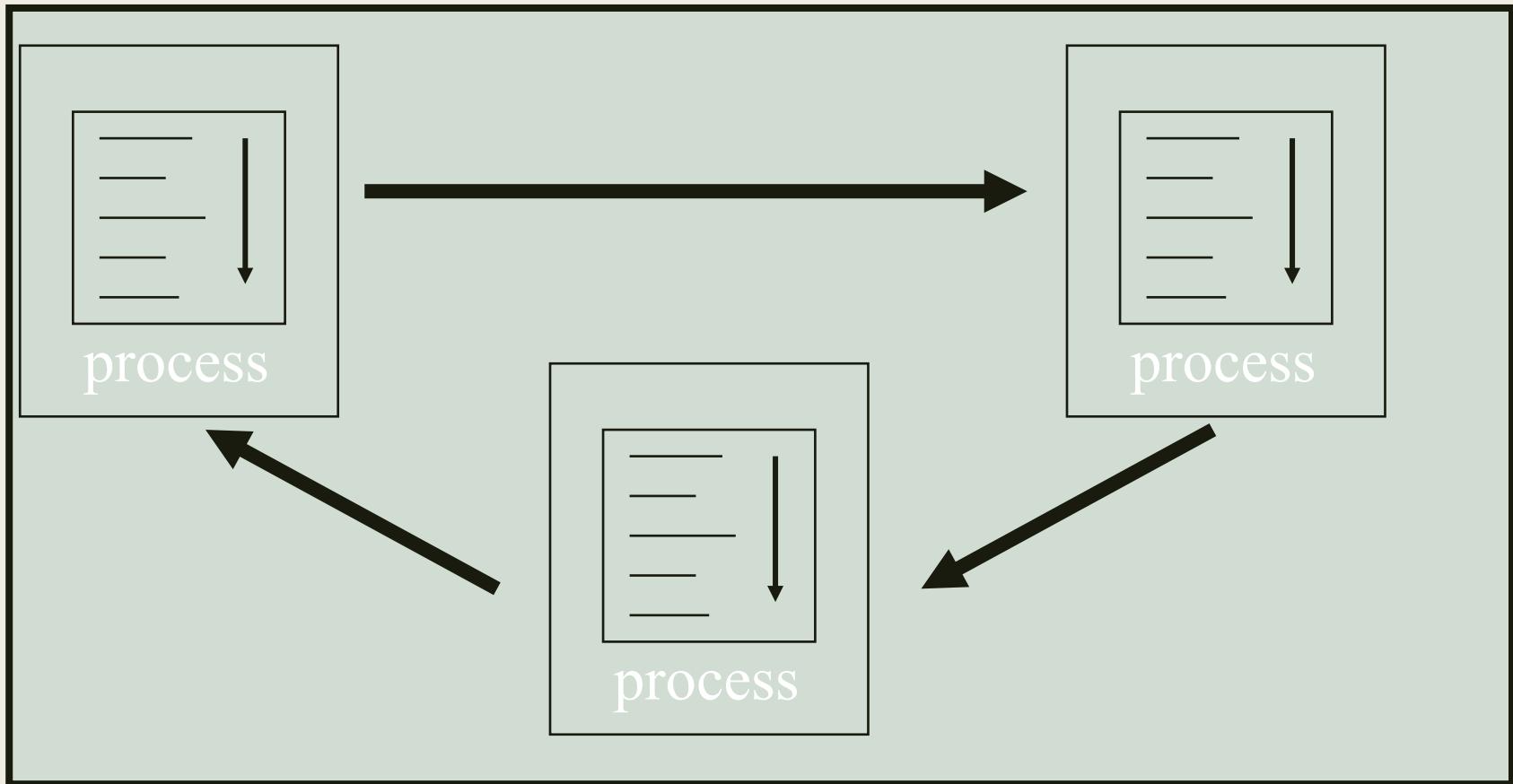
Περιγραφή συμπεριφοράς (behavioral) στη VHDL

- **arch_name**: το όνομα της αρχιτεκτονικής
- **entity_name**: το όνομα της οντότητας
- **label**: οι μοναδικές ετικέτες των διεργασιών
 - *μη υποχρεωτικό, αλλά χρήσιμο κατά τη σύνθεση γιατί τα παραγόμενα σήματα συνήθως συμπεριλαμβάνουν στο όνομά τους και την ετικέτα της διεργασίας από την οποία προέρχονται*
- **process**: η διεργασία περιέχει μία ομάδα από εντολές που εκτελούνται ακολουθιακά
 - *Οι ακολουθιακές εντολές εμπεριέχουν σήματα και μεταβλητές*

Περιγραφή συμπεριφοράς (behavioral) στη VHDL

- **signal_name**: το όνομα του σήματος
(εάν είναι πολλά σήματα χωρίζονται με κόμμα)
 - στις δηλώσεις των σημάτων (μετά το process) που απαρτίζουν τη **λίστα ευαισθησίας** (sensitivity list) το σήμα είναι **είσοδος** της υπομονάδας που δηλώνεται κατά τη δήλωση των διαύλων της οντότητας ή **εσωτερική διασύνδεση** της υπομονάδας
 - κάθε αλλαγή τιμής σήματος εισόδου που ανήκει στη **λίστα ευαισθησίας** οδηγεί στην ακολουθιακή εκτέλεση των εντολών της διεργασίας **μία φορά**
 - εάν περισσότερες από μία εντολές αναθέτουν τιμή σε κάποιο σήμα λαμβάνεται υπόψη μόνο η **τελευταία ακολουθιακή εντολή**
 - **Προσοχή στη δήλωση των σημάτων στη λίστα ευαισθησίας**
 - μία ελλιπής δήλωση σημάτων στη λίστα ευαισθησίας θα οδηγήσει είτε σε μη σωστή σύνθεση, είτε σε ασυμφωνία της προσομοίωσης πριν και μετά τη σύνθεση και την υλοποίηση

Περιγραφή συμπεριφοράς (behavioral) στη VHDL



Κάθε διεργασία (process) εκτελεί τις εντολές της ακολουθιακά, ενώ πολλές διεργασίες μαζί αλληλεπιδρούν ταυτόχρονα. Επίσης, ταυτόχρονα αλληλεπιδρούν εντολές ταυτόχρονης ανάθεσης και διεργασίες.

Περιγραφή συμπεριφοράς (behavioral) στη VHDL

- **variable_name**: το όνομα της μεταβλητής
(εάν είναι πολλές μεταβλητές χωρίζονται με κόμμα)
 - **μέσα** στις διεργασίες ορίζονται **ΤΟΠΙΚΕΣ μεταβλητές** και **OXI εσωτερικά σήματα**
 - στις δηλώσεις μεταβλητών (μετά το **variable**) προσδιορίζονται μεταβλητές που μπορεί να μην έχουν τη φυσική σημασία του σήματος
- **variable_type**: ο τύπος της μεταβλητής (STD_LOGIC)

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Σήματα vs μεταβλητές

- Διαφορά στην εφαρμογή της τιμής μίας ακολουθιακής εντολής ανάθεσης μεταβλητής ή σήματος μέσα σε μία διεργασία κατά την προσομοίωση:
 - η μεταβλητή παίρνει νέα τιμή **άμεσα** με τον τελεστή ανάθεσης `:=`, αμέσως μόλις εκτελεστεί η αντίστοιχη εντολή μέσα στη διεργασία
 - **Δεν χρησιμοποιείται στην υλοποίηση ακολουθιακής λογικής**
 - σε αντίθεση, το σήμα παίρνει νέα τιμή **με καθυστέρηση δέλτα δ_{delay}** με τον τελεστή ανάθεσης `<=`, στο **τέλος** της εκτέλεσης της διεργασίας
 - το σήμα **θυμάται την τιμή του** μέχρι να φτάσει το τέλος της εκτέλεσης της διεργασίας και να λάβει μία νέα τιμή
 - **Χρησιμοποιείται στην υλοποίηση ακολουθιακής λογικής**



Η χρήση των μεταβλητών μειώνει σημαντικά το χρόνο της προσομοίωσης

Περιγραφή συμπεριφοράς (behavioral) στη VHDL

- Ακολουθιακές εντολές ανάθεσης σήματος (με μη άμεση εφαρμογή τιμής) (sequential_signal_assignment_statements)

```
signal_name <= expression;
```

- **signal_name**: το όνομα του σήματος
- **expression**: έκφραση με σήματα, μεταβλητές και τελεστές
 - στις ακολουθιακές εντολές ανάθεσης σήματος :
 - στην έκφραση προσδιορίζονται **σήματα**, που ανήκουν ή δεν ανήκουν στη λίστα ευαισθησίας, και **μεταβλητές** που δηλώνονται κατά τη δήλωση μεταβλητών
 - στο αριστερό μέρος της εντολής προσδιορίζεται σήμα που είναι **έξοδος** της υπομονάδας ή **εσωτερική διασύνδεση** της υπομονάδας

Περιγραφή συμπεριφοράς (behavioral) στη VHDL

- Ακολουθιακές εντολές ανάθεσης μεταβλητής (με άμεση εφαρμογή τιμής) (sequential_variable_assignment_statements)

```
variable_name := expression;
```

- **variable_name**: το όνομα της μεταβλητής
- **expression**: έκφραση με σήματα, μεταβλητές και τελεστές
 - στις ακολουθιακές εντολές ανάθεσης μεταβλητής :
 - στην έκφραση προσδιορίζονται **σήματα**, που ανήκουν ή δεν ανήκουν στη λίστα ευαισθησίας, και **μεταβλητές** που δηλώνονται κατά τη δήλωση μεταβλητών
 - στο αριστερό μέρος της εντολής προσδιορίζεται **μεταβλητή** που δηλώνεται κατά τη δήλωση των μεταβλητών
 - οι μεταβλητές επιδρούν **ΤΟΠΙΚΑ** εντός της διεργασίας

Δεν χρησιμοποιείται στην υλοποίηση ακολουθιακής λογικής

Η αρχιτεκτονική του ημιαθροιστή στη VHDL

Περιγραφή συμπεριφοράς

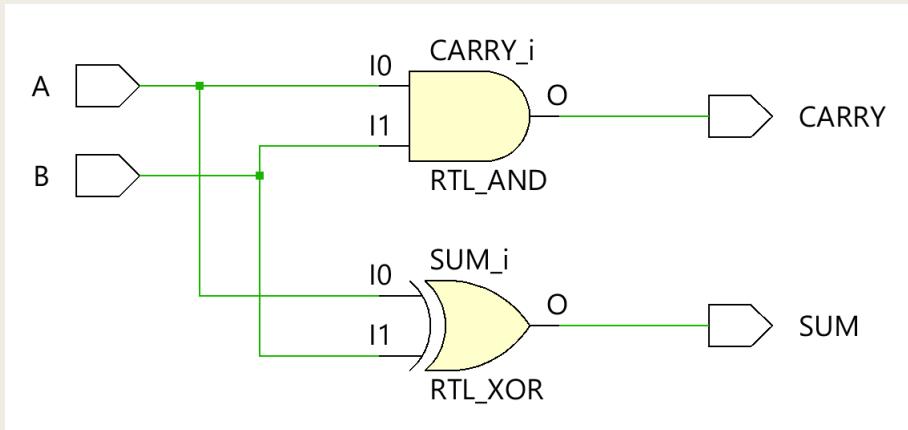
```
architecture HA_BEHAVIORAL of HALF_ADDER is
begin
  process (A, B)
  begin
    SUM <= A xor B;
    CARRY <= A and B;
  end process;
end HA_BEHAVIORAL;
```

Οι ακολουθιακές εντολές ανάθεσης σήματος εκτελούνται η μία μετά την άλλη, μόνο όταν υπάρξει αλλαγή τιμής στις εισόδους που δηλώνονται στη λίστα ευαισθησίας (στα σήματα της δεξιάς πλευράς)

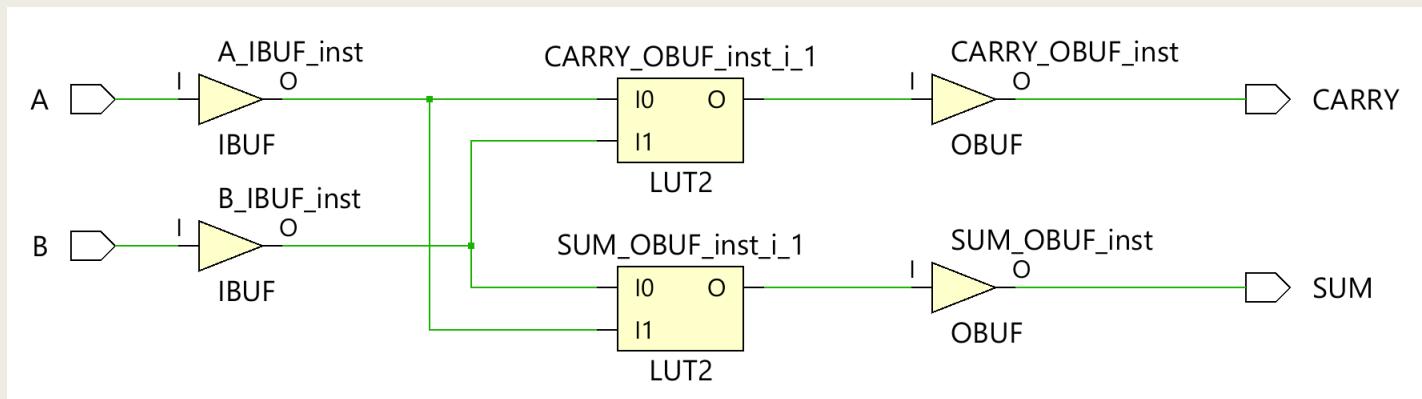
Η αρχιτεκτονική του ημιαθροιστή στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA



Περιγραφή δομής στη VHDL

```
architecture arch_name of entity_name is
    signal signal_name: signal_type;
    component comp_name
        port (
            signal_name: mode signal_type;
            ...
            signal_name: mode signal_type);
    end component;
    ...
begin
    concurrent_component_statement;
    ...
    concurrent_component_statement;
end arch_name;
```

Περιγραφή δομής στη VHDL

- **arch_name**: το όνομα της αρχιτεκτονικής
- **entity_name**: το όνομα της οντότητας
- **comp_name**: το όνομα του **στοιχείου (component)** που χρησιμοποιείται στην αρχιτεκτονική της οντότητας
 - *To στοιχείο είναι μία ήδη **προκαθορισμένη οντότητα***
- **signal_name**: το όνομα του σήματος
(εάν είναι πολλά σήματα χωρίζονται με κόμμα)
 - *στις δηλώσεις σημάτων (μετά το signal) το σήμα είναι μία **εσωτερική διασύνδεση** της υπομονάδας*
 - *στις δηλώσεις των διαύλων του στοιχείου (component) το σήμα είναι **είσοδος ή έξοδος του στοιχείου**, όπως προκύπτει από τη δήλωση των διαύλων της οντότητας του συγκεκριμένου στοιχείου*
- **signal_type**: ο τύπος του σήματος (STD_LOGIC)

Περιγραφή δομής στη VHDL

- Ταυτόχρονες εντολές στοιχείων (concurrent_component_statements)

```
label: comp_name port map (signal_name, . .) ;
```

- **label**: οι μοναδικές ετικέτες των στοιχείων
- **comp_name**: το όνομα του στοιχείου που χρησιμοποιείται στην αρχιτεκτονική της οντότητας
- **signal_name**: το όνομα του σήματος
(εάν είναι πολλά σήματα χωρίζονται με κόμμα)
 - το σήμα είναι μία διασύνδεση που αφορά τη συγκεκριμένη αρχιτεκτονική της οντότητας που χρησιμοποιεί το στοιχείο
 - αντιστοιχεί αμφιμονοσήμαντα στο αντίστοιχο σήμα της δήλωσης των διαύλων του στοιχείου
 - Προσοχή στη διατήρηση της σειράς **Των σημάτων**
 - Εναλλακτικά περιγράφουμε την αμφιμονοσήμαντη αντιστοιχία, ώστε ο κώδικας να διαβάζεται πιο εύκολα
component_signal_name => entity_signal_name

Η αρχιτεκτονική του ημιαθροιστή στη VHDL

Περιγραφή δομής

- Παρακάτω φαίνεται η αρχιτεκτονική του ημιαθροιστή σε περιγραφή δομής (structural) που υλοποιεί τις εξισώσεις Boole:

$$\text{SUM} = A \oplus B$$

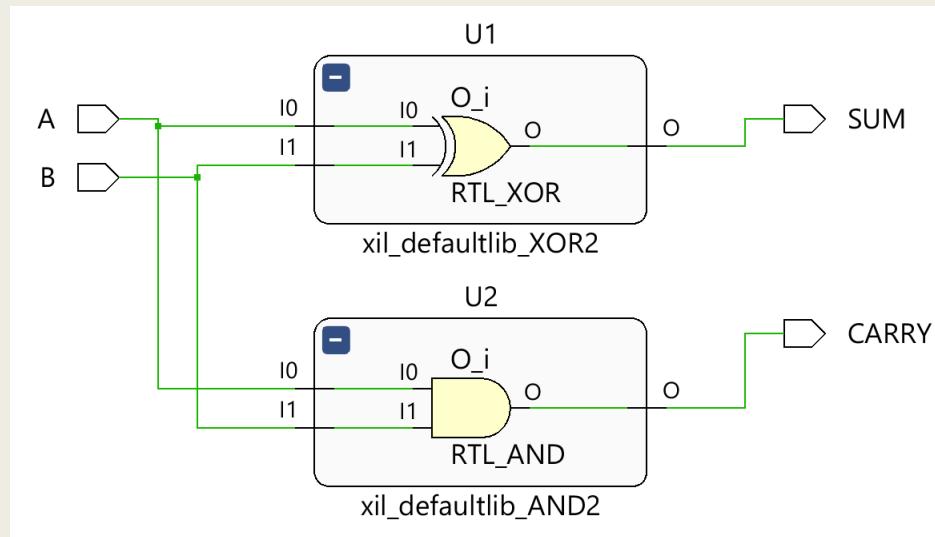
$$\text{CARRY} = AB$$

```
architecture HA_STRUCTURAL of HALF_ADDER is
component XOR2
    port (O : out STD_LOGIC; I0 : in STD_LOGIC; I1 : in STD_LOGIC);
end component;
component AND2
    port (O : out STD_LOGIC; I0 : in STD_LOGIC; I1 : in STD_LOGIC);
end component;
begin
    U1: XOR2 port map (O => SUM, I0 => A, I1 => B);
    U2: AND2 port map (O => CARRY, I0 => A, I1 => B);
end HA_STRUCTURAL;
```

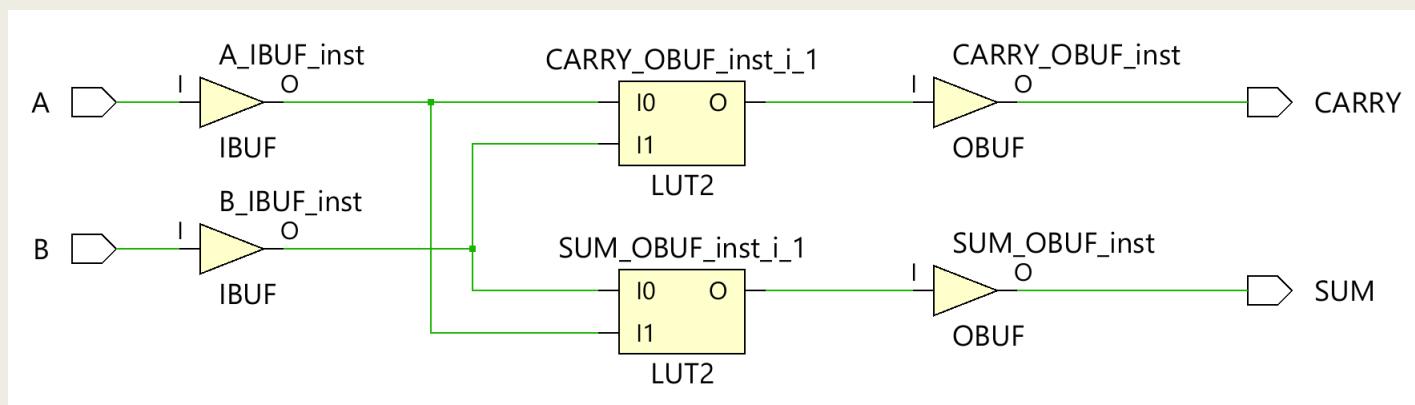
Τα στοιχεία XOR2 και AND2 έχουν ήδη προκαθοριστεί σαν οντότητες του project

Η αρχιτεκτονική του ημιαθροιστή στη VHDL Σύνθεση περιγραφής δομής

- Σχηματικό διάγραμμα RTL (φαίνεται και η ιεραρχία)



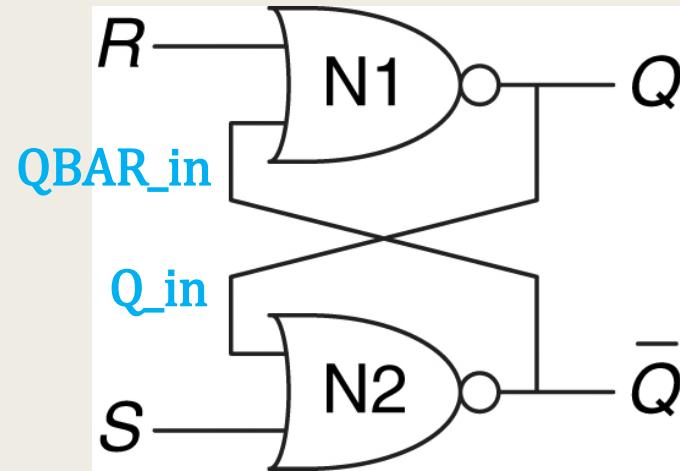
- Σχηματικό διάγραμμα σε τεχνολογία FPGA



To S-R Latch (Active High) στη VHDL

Περιγραφή Dataflow

```
entity SRL is port (
    S, R: in STD_LOGIC;
    Q, QBAR: out STD_LOGIC);
end SRL;
architecture SRL_DF of SRL is
signal Q_in, QBAR_in: STD_LOGIC;
begin
    QBAR_in <= S nor Q_in;
    Q_in <= R nor QBAR_in;
    QBAR <= QBAR_in;
    Q <= Q_in;
end SRL_DF;
```



Q_in, QBAR_in:
Εσωτερικά σήματα (signals)

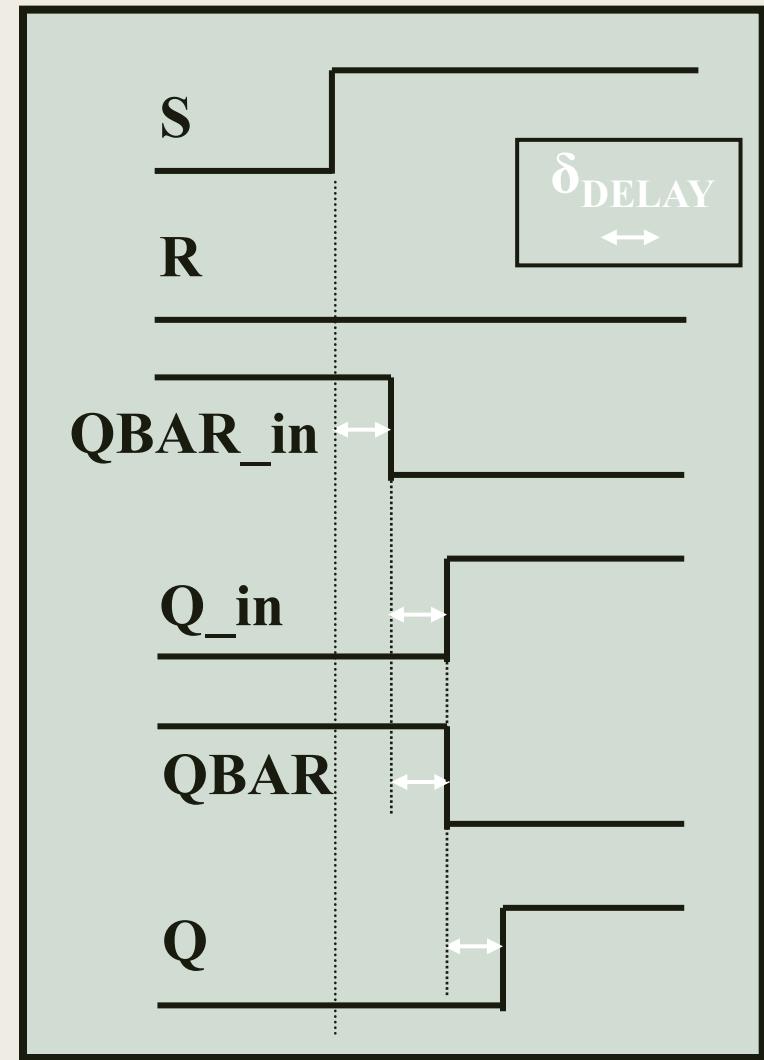
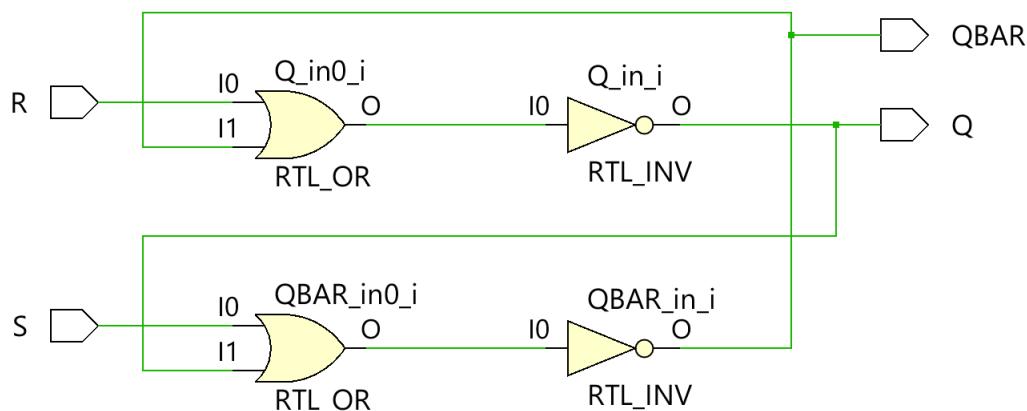
To S-R Latch (Active High) στη VHDL

Περιγραφή Dataflow

```

entity SRL is port (
    S, R: in STD_LOGIC;
    Q, QBAR: out STD_LOGIC);
end SRL;
architecture SRL_DF of SRL is
signal Q_in, QBAR_in: STD_LOGIC;
begin
    QBAR_in <= S nor Q_in;
    Q_in <= R nor QBAR_in;
    QBAR <= QBAR_in;
    Q <= Q_in;
end SRL_DF;

```

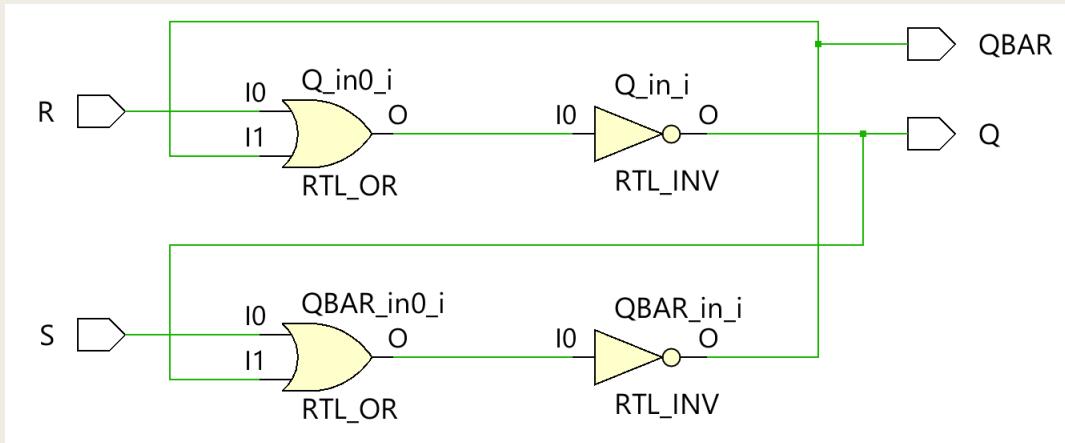


Η ενημέρωση των σημάτων γίνεται με καθυστέρηση δέλτα δ_{delay} αμέσως μόλις εκτελεστεί η αντίστοιχη εντολή

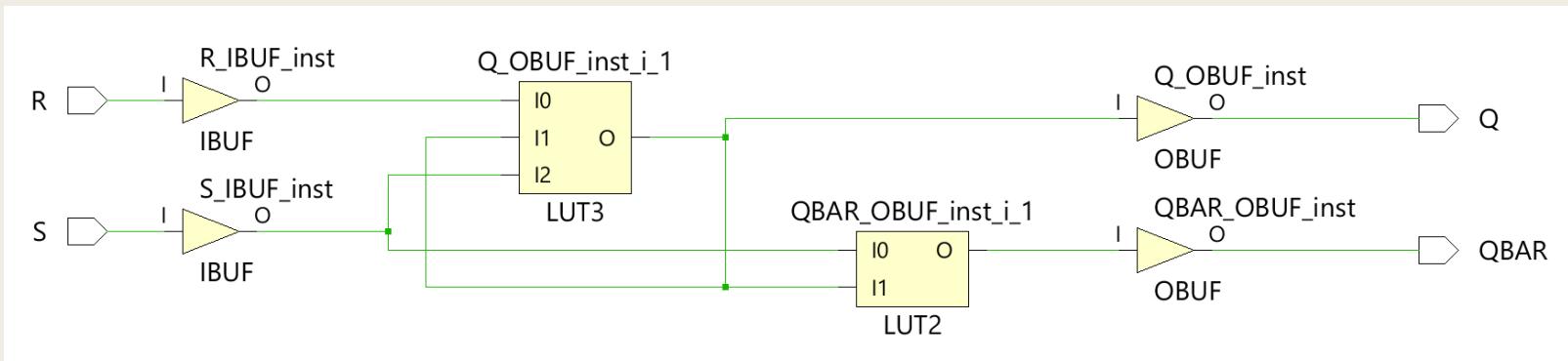
To SR Latch στη VHDL

Σύνθεση περιγραφής Dataflow

■ Σχηματικό διάγραμμα RTL



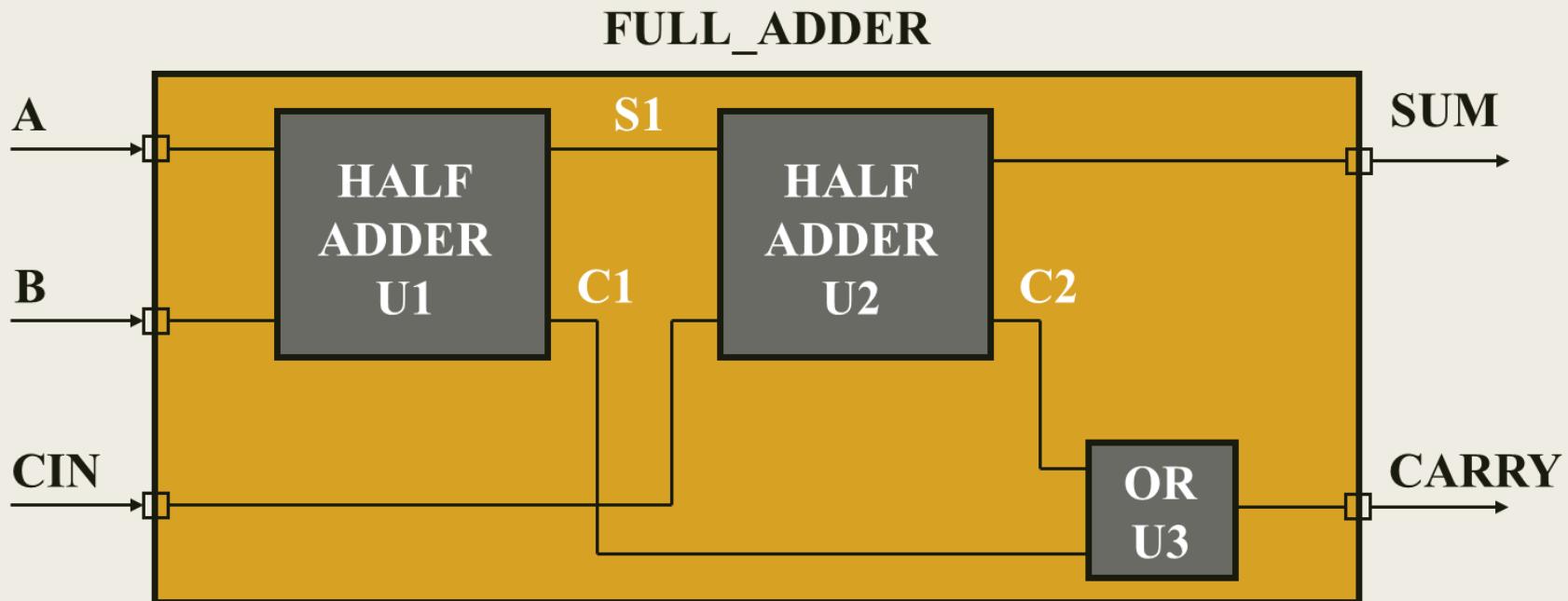
■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



Δημιουργία ιεραρχικής δομής στη VHDL

Περιγραφή δομής

- Παράδειγμα: Πλήρης αθροιστής που αποτελείται από δύο ημιαθροιστές και μία πύλη OR

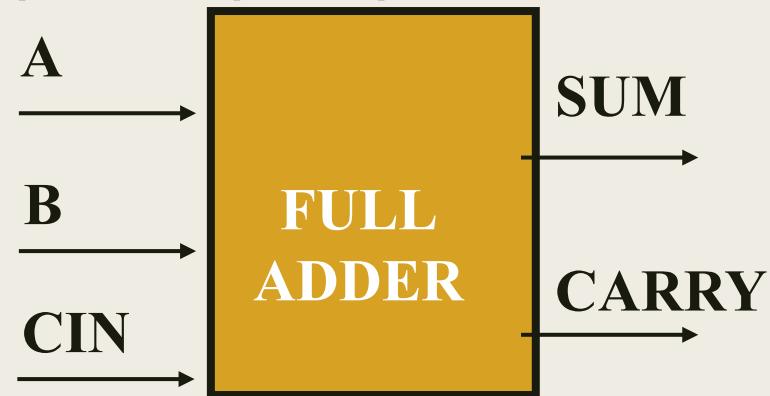


Η οντότητα του πλήρους αθροιστή στη VHDL

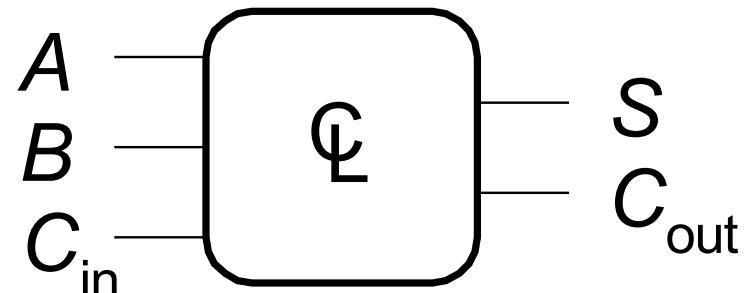
```

entity FULL_ADDER is
    port (
        A: in STD_LOGIC;
        B: in STD_LOGIC;
        CIN: in STD_LOGIC;
        SUM: out STD_LOGIC;
        CARRY: out STD_LOGIC);
end FULL_ADDER;

```



C_{in}	A	B	C_{out}	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



$$\begin{aligned}
 S &= A \oplus B \oplus C_{in} \\
 C_{out} &= AB + AC_{in} + BC_{in}
 \end{aligned}$$

Η αρχιτεκτονική του πλήρους αθροιστή στη VHDL Περιγραφή dataflow

- Με εσωτερικά σήματα (δηλώνονται στο signal)

```
architecture FA_DATAFLOW1 of FULL_ADDER is
    signal S1, P1, P2, P3: STD_LOGIC;
begin
    S1 <= A xor B;
    SUM <= CIN xor S1;
    P1 <= A and B;
    P2 <= A and CIN;
    P3 <= B and CIN;
    CARRY <= P1 or P2 or P3;
end FA_DATAFLOW1;
```

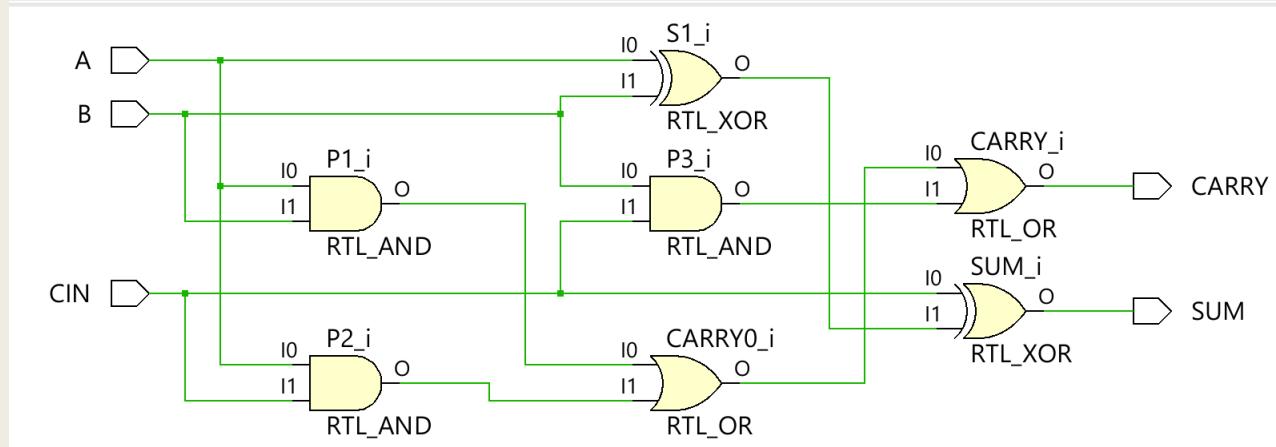
- Χωρίς εσωτερικά σήματα

```
architecture FA_DATAFLOW2 of FULL_ADDER is
begin
    SUM <= A xor B xor CIN;
    CARRY <= (A and B) or (A and CIN) or (B and CIN);
end FA_DATAFLOW2;
```

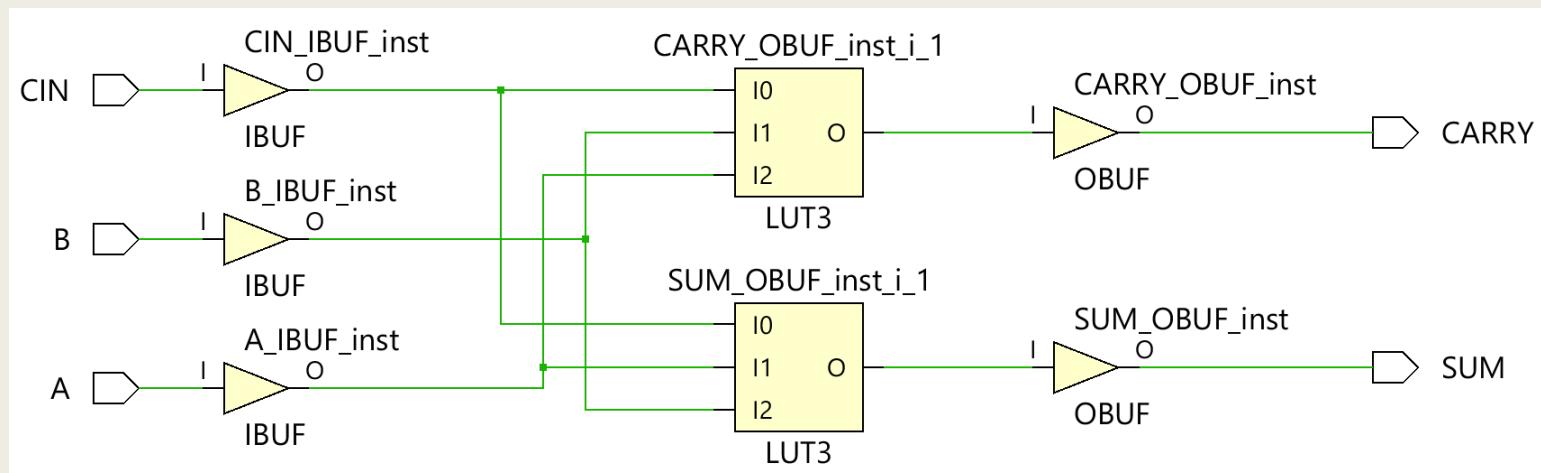
Η αρχιτεκτονική του πλήρους αθροιστή στη VHDL

Περιγραφή dataflow

■ Σχηματικό διάγραμμα RTL



■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



Η αρχιτεκτονική του πλήρους αθροιστή στη VHDL

Περιγραφή δομής

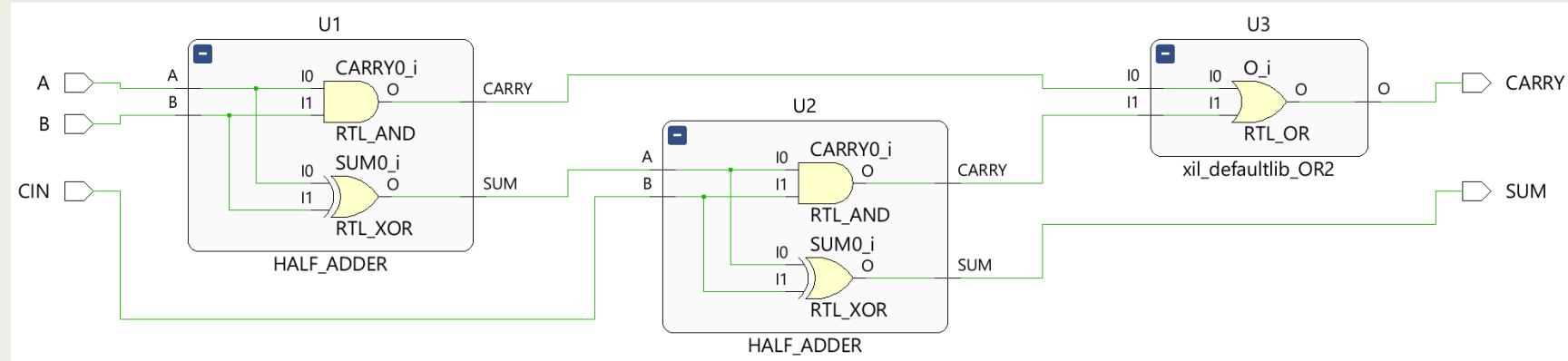
```
architecture FA_STRUCTURAL of FULL_ADDER is
    signal S1, C1, C2: STD_LOGIC;
    component HALF_ADDER
        port (A : in STD_LOGIC; B : in STD_LOGIC;
              SUM : out STD_LOGIC; CARRY : out STD_LOGIC);
    end component;
    component OR2
        port (O : out STD_LOGIC; I0 : in STD_LOGIC; I1 : in STD_LOGIC);
    end component;
begin
    U1: HALF_ADDER port map (A => A, B => B, SUM => S1, CARRY => C1);
    U2: HALF_ADDER port map (A => S1, B => CIN, SUM => SUM, CARRY => C2);
    U3: OR2 port map (O => CARRY, I0 => C1, I1 => C2);
end FA_STRUCTURAL;
```

Τα στοιχεία HALF_ADDER και OR2 έχουν ήδη προκαθοριστεί
σαν οντότητες (προσέγγιση σχεδίασης **bottom-up**) του project

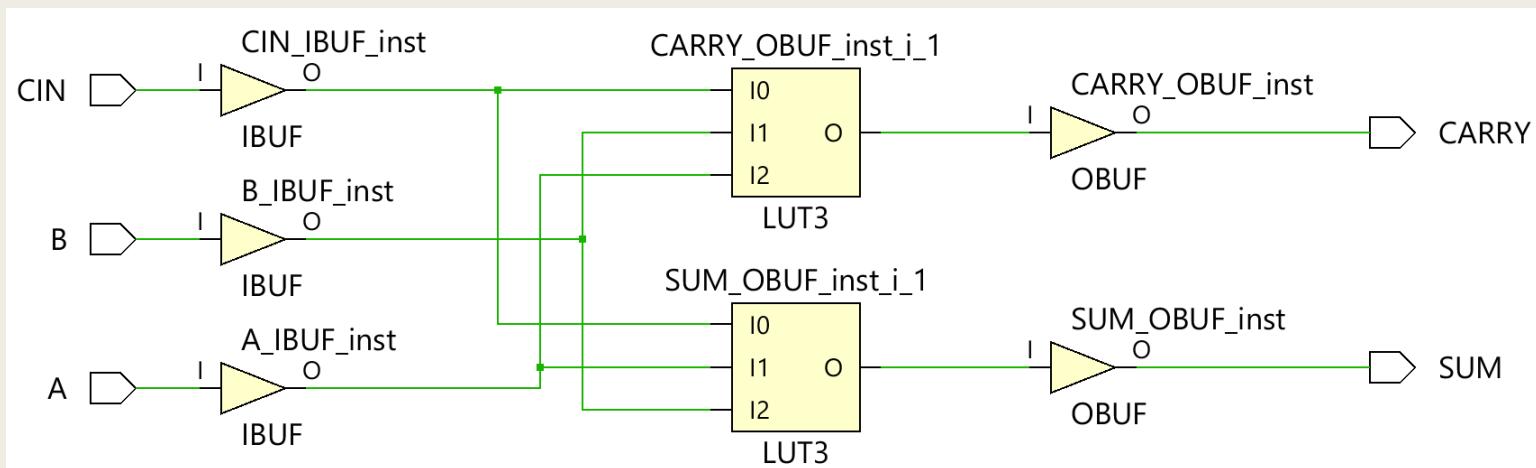
Η αρχιτεκτονική του πλήρους αθροιστή στη VHDL

Περιγραφή δομής

■ Σχηματικό διάγραμμα RTL



■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



Η αρχιτεκτονική του πλήρους αθροιστή στη VHDL

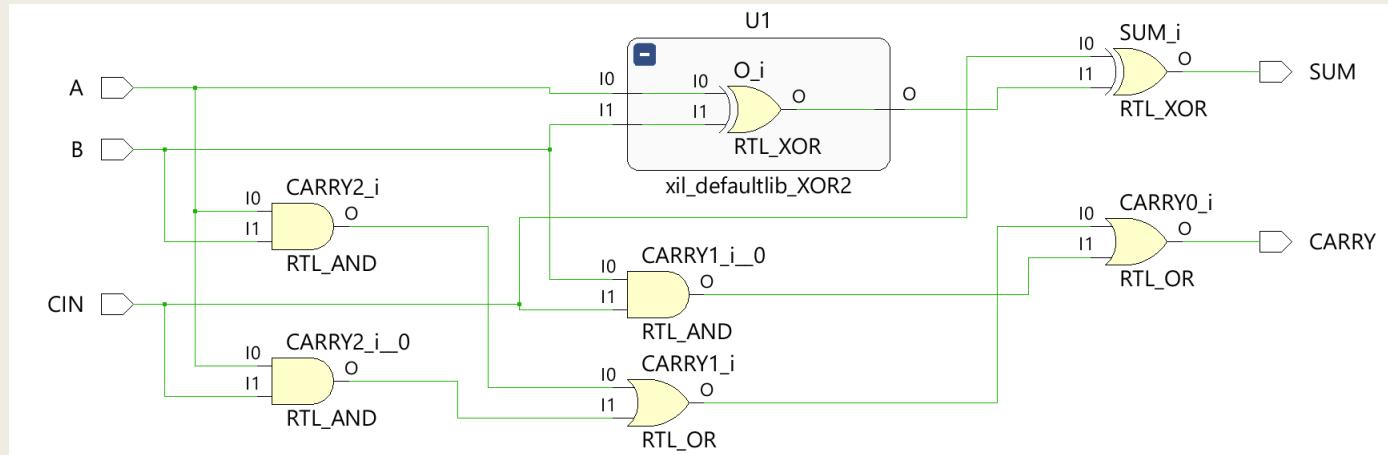
Μικτή περιγραφή (δομής, dataflow, συμπεριφοράς)

```
architecture FA_MIXED of FULL_ADDER is
    signal S1: STD_LOGIC;
    component XOR2
        port (O : out STD_LOGIC;
              I0 : in STD_LOGIC; I1 : in STD_LOGIC);
    end component;
begin
    U1: XOR2 port map (O => S1, I0 => A, I1 => B);
    SUM <= CIN xor S1;
    process (A, B, CIN)
        variable V1, V2, V3: STD_LOGIC;
    begin
        V1 := A and B;
        V2 := A and CIN;
        V3 := B and CIN;
        CARRY <= V1 or V2 or V3;
    end process;
end FA_MIXED;
```

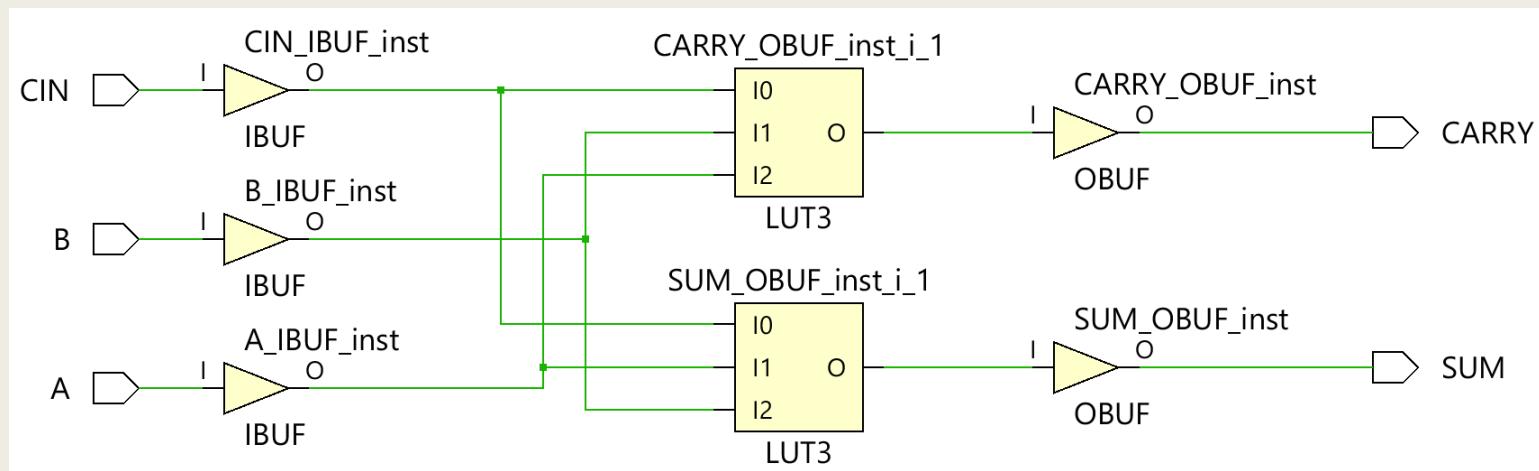
Η αρχιτεκτονική του πλήρους αθροιστή στη VHDL

Μικτή περιγραφή (δομής, dataflow, συμπεριφοράς)

■ Σχηματικό διάγραμμα RTL



■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Εντολές με συνθήκη

```
architecture arch_name of entity_name is
begin
  process (signal_name, . . . , signal_name)
    variable variable_name: variable_type;
  begin
    sequential_signal_assignment_statement;
    sequential_variable_assignment_statement;
    conditional_signal_assignment_statement;
    conditional_variable_assignment_statement;
  end process;
end arch_name;
```

Με την περιγραφή συμπεριφοράς προσδιορίζουμε τη λειτουργικότητα και όχι τη δομή της υπομονάδας.

Αν και οι εντολές μέσα σε μία διεργασία (process) αξιολογούνται ακολουθιακά, μετά τη σύνθεση προκύπτει ταυτόχρονη λογική

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Η εντολή IF

- Από τις πιο σημαντικές ακολουθιακές εντολές ανάθεσης σήματος (ή μεταβλητής) **με συνθήκη** που χρησιμοποιούνται μέσα σε **process**
 - Η εντολή IF εξετάζει μία συνθήκη και, εάν αληθεύει, εκτελεί την ακολουθιακή εντολή 1, αλλιώς, εκτελεί την ακολουθιακή εντολή 2 (εάν ορίζεται)
- Δομή εντολής IF

```
if boolean_expression (condition) then
    sequential_statement_1;
end if;
```

```
if boolean_expression (condition) then
    sequential_statement_1;
else
    sequential_statement_2;
end if;
```

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Η εντολή IF

- Η εντολή IF επιτρέπει να εξετασθεί μία διατεταγμένη σειρά από συνθήκες με τη χρήση της φράσης **elsif**
 - Εκτελείται μόνο η ακολουθιακή εντολή για την οποία αληθεύει η συνθήκη (Στη γενική περίπτωση μπορούν να εκτελεστούν και 2 ή περισσότερες εντολές)
 - Η σειρά με την οποία γράφονται οι εντολές είναι σημαντική
 - στην περίπτωση που περισσότερες από μία συνθήκες αληθεύουν ταυτόχρονα, θα εκτελεσθεί εκείνη η εντολή για την οποία η συνθήκη αληθεύει πρώτη
- Δομή εντολής IF

```
if boolean_expression_1 (condition_1) then
    sequential_statement_1;
elsif boolean_expression_2 (condition_2) then
    sequential_statement_2;
else
    sequential_statement_3;
end if;
```

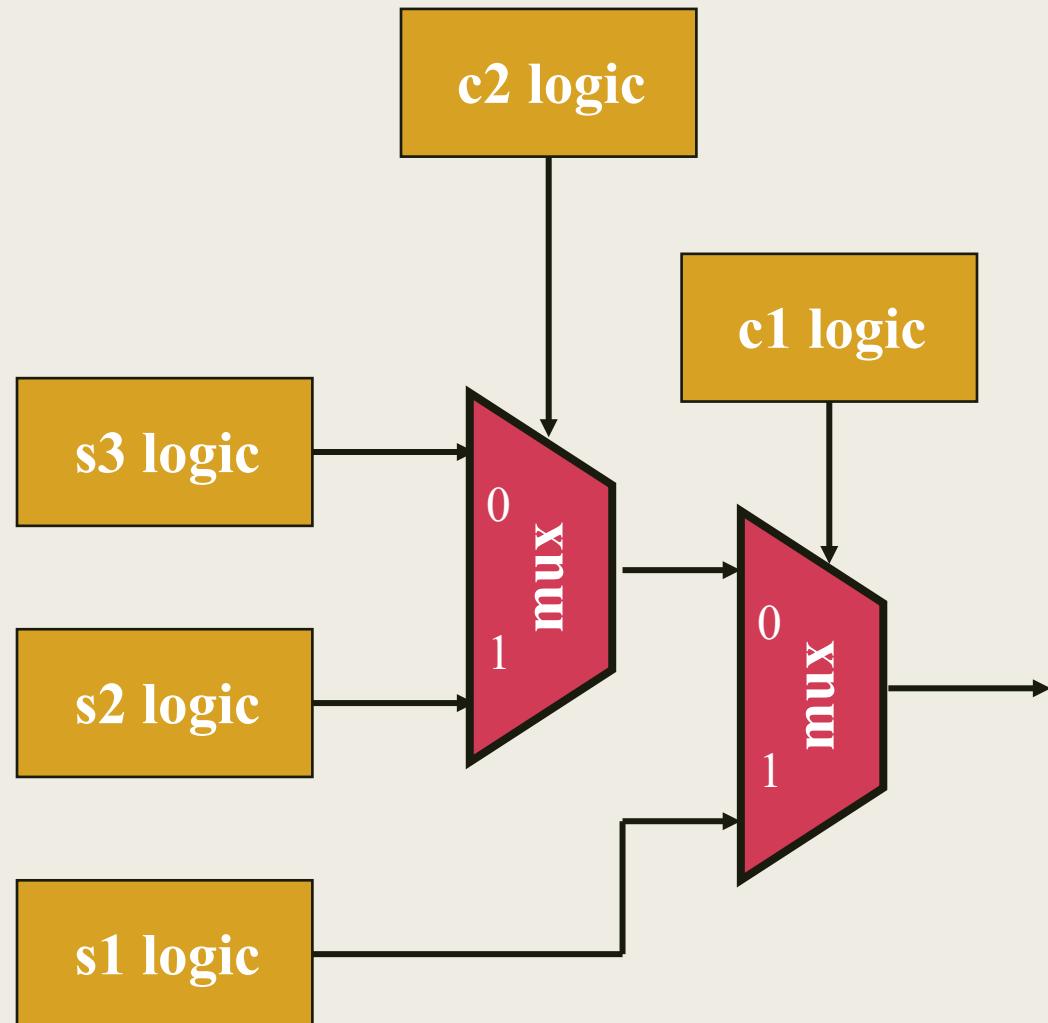
Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Η εντολή IF

- Boolean Expression
 - υλοποιεί μία συνθήκη
$$(A = B)$$
 - ή μια σειρά από συνθήκες που συνδέονται μεταξύ τους με λογικούς τελεστές (*and, or, not, nand, nor, xor, xnor*)
$$(A = "000" \text{ or } \text{RESET} = '1')$$
- Συνθήκη (condition)
 - επιστρέφει μία boolean τιμή (*TRUE, FALSE*)
 - περιγράφεται με δύο τελεστέους του *ιδίου τύπου*, που συγκρίνονται με τη χρήση ενός τελεστή σύγκρισης
$$(<, <=, >, >=, =, /=)$$

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Η εντολή IF

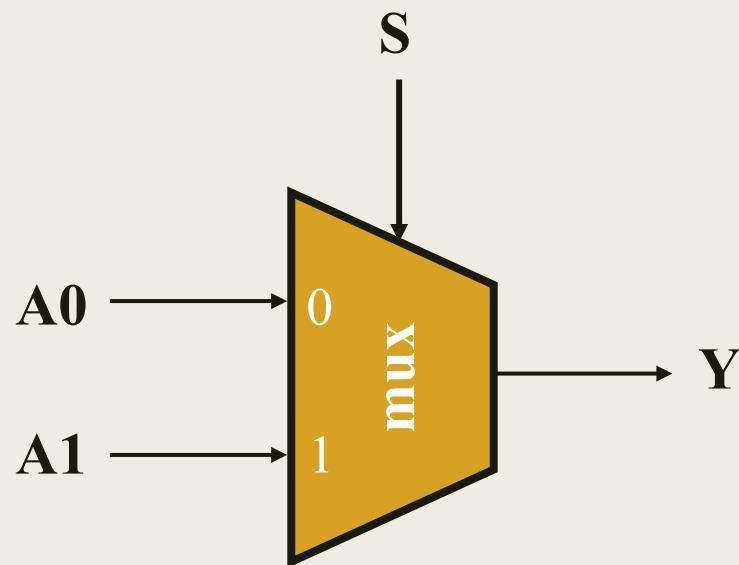
- Υλοποίηση εντολής IF με τη χρήση πολυπλεκτών 2 σε 1

```
if c1 then  
    s1;  
elsif c2 then  
    s2;  
else  
    s3;  
end if;
```



Η οντότητα του πολυπλέκτη 2 σε 1 στη VHDL

```
entity MUX_2_to_1 is
    port (
        A0: in STD_LOGIC;
        A1: in STD_LOGIC;
        S: in STD_LOGIC;
        Y: out STD_LOGIC);
end MUX_2_to_1;
```



Η αρχιτεκτονική του πολυπλέκτη 2 σε 1 στη VHDL

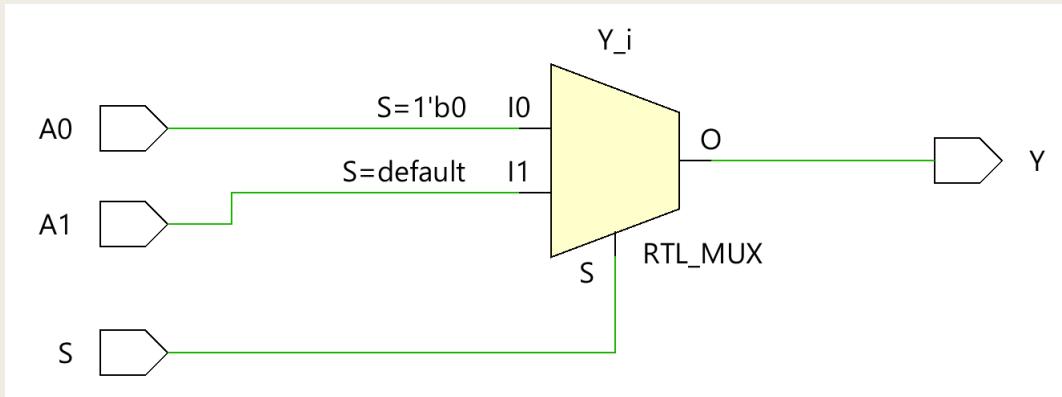
Περιγραφή συμπεριφοράς

```
architecture BEHAVIORAL of MUX_2_to_1 is
begin
  process (A0, A1, S)
  begin
    if (S = '0') then
      Y <= A0;
    else
      Y <= A1;
    end if;
  end process;
end BEHAVIORAL;
```

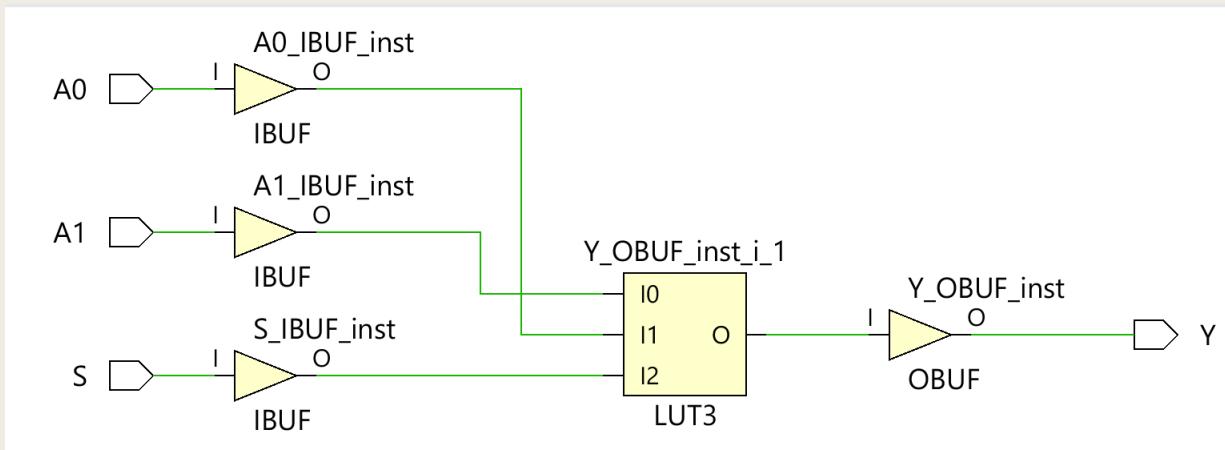
Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες οι είσοδοι του συνδυαστικού κυκλώματος

Η αρχιτεκτονική του πολυπλέκτη 2 σε 1 στη VHDL Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL

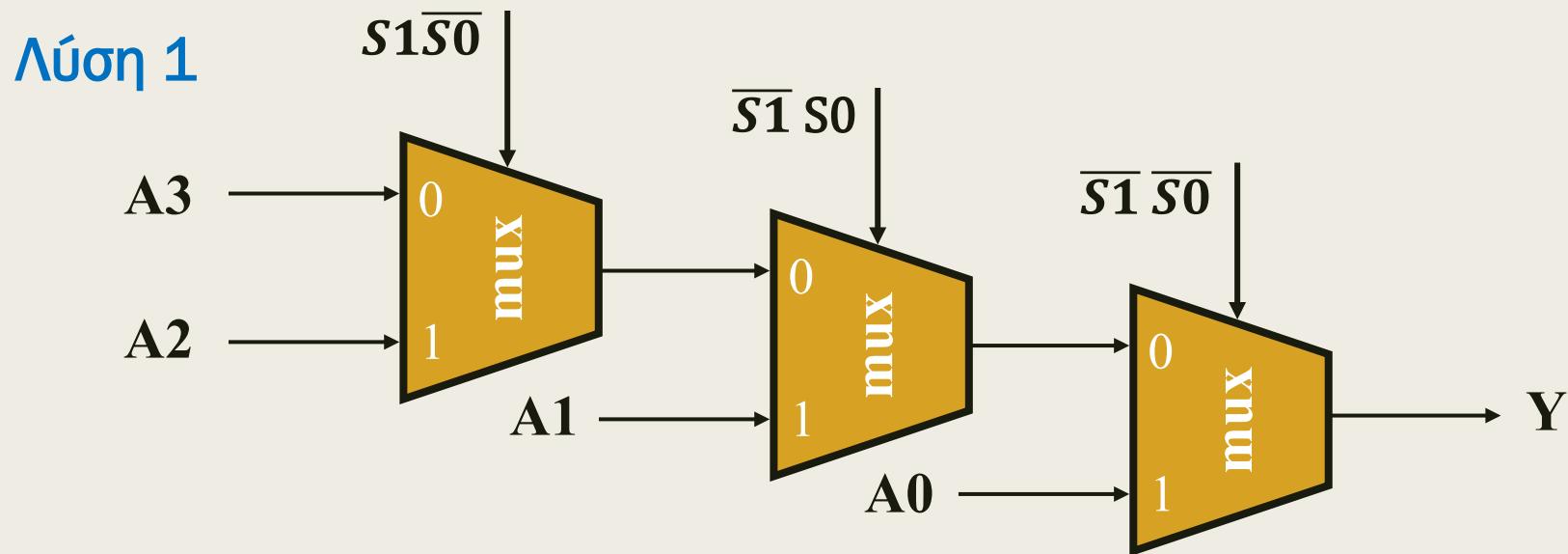
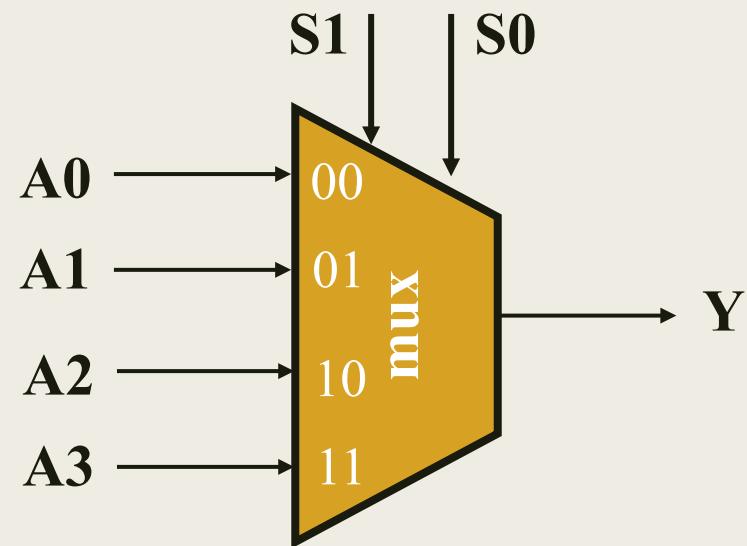


- Σχηματικό διάγραμμα σε τεχνολογία FPGA



Η οντότητα του πολυπλέκτη 4 σε 1 στη VHDL

```
entity MUX_4_to_1 is
  port (
    A0: in STD_LOGIC;
    A1: in STD_LOGIC;
    A2: in STD_LOGIC;
    A3: in STD_LOGIC;
    S0: in STD_LOGIC;
    S1: in STD_LOGIC;
    Y: out STD_LOGIC);
end MUX_4_to_1;
```



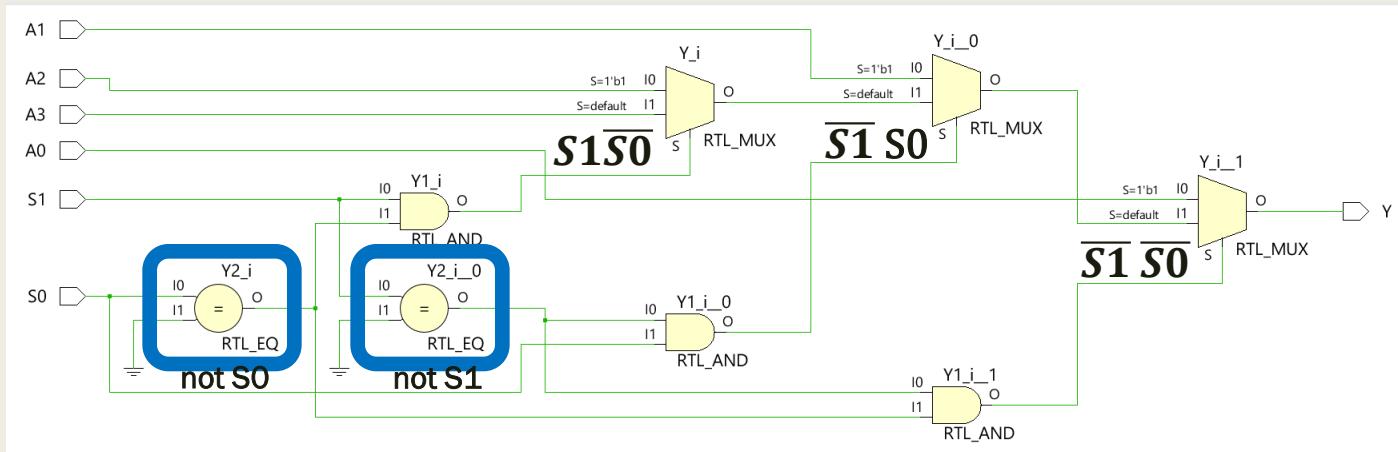
Η αρχιτεκτονική του πολυπλέκτη 4 σε 1 στη VHDL Περιγραφή συμπεριφοράς – Λύση 1

```
architecture BEHAVIORAL of MUX_4_to_1 is
begin
  process (A0, A1, A2, A3, S0, S1)
  begin
    if      (S1 = '0' and S0 = '0') then Y <= A0;
    elsif  (S1 = '0' and S0 = '1') then Y <= A1;
    elsif  (S1 = '1' and S0 = '0') then Y <= A2;
    else
      Y <= A3;
    end if;
  end process;
end BEHAVIORAL;
```

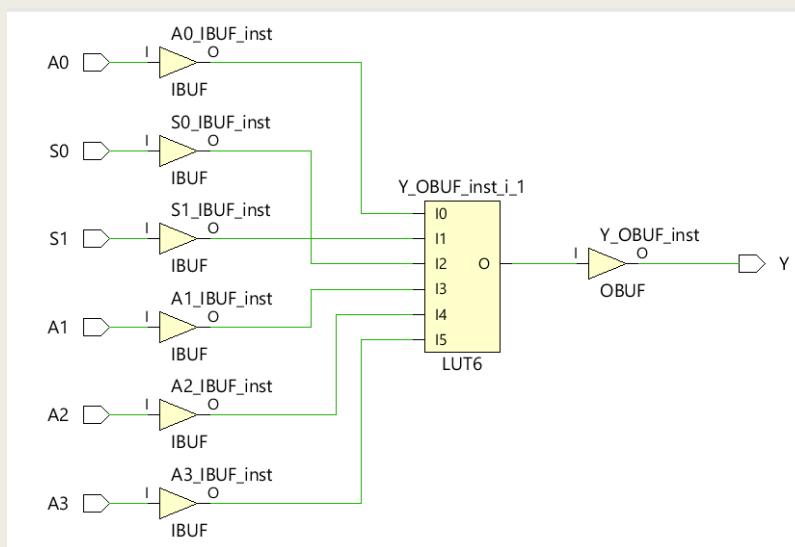
Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες
οι είσοδοι του συνδυαστικού κυκλώματος

Η αρχιτεκτονική του πολυπλέκτη 4 σε 1 στη VHDL Περιγραφή συμπεριφοράς – Λύση 1

■ Σχηματικό διάγραμμα RTL

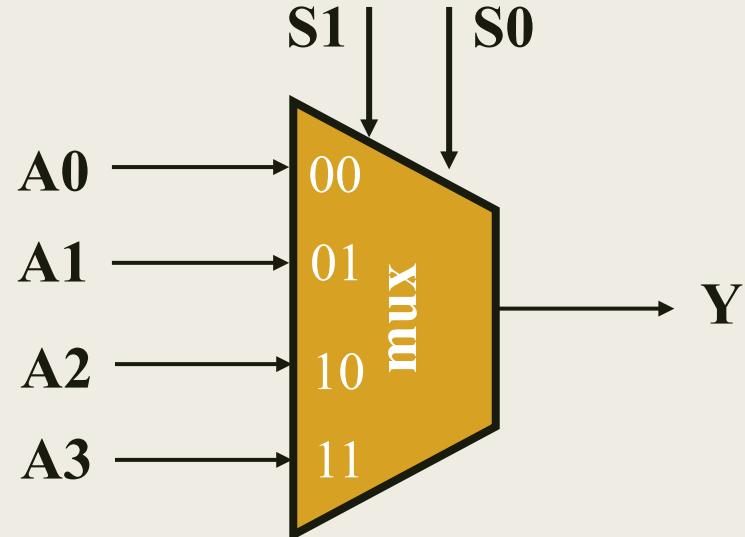


■ Σχηματικό διάγραμμα σε τεχνολογία FPGA

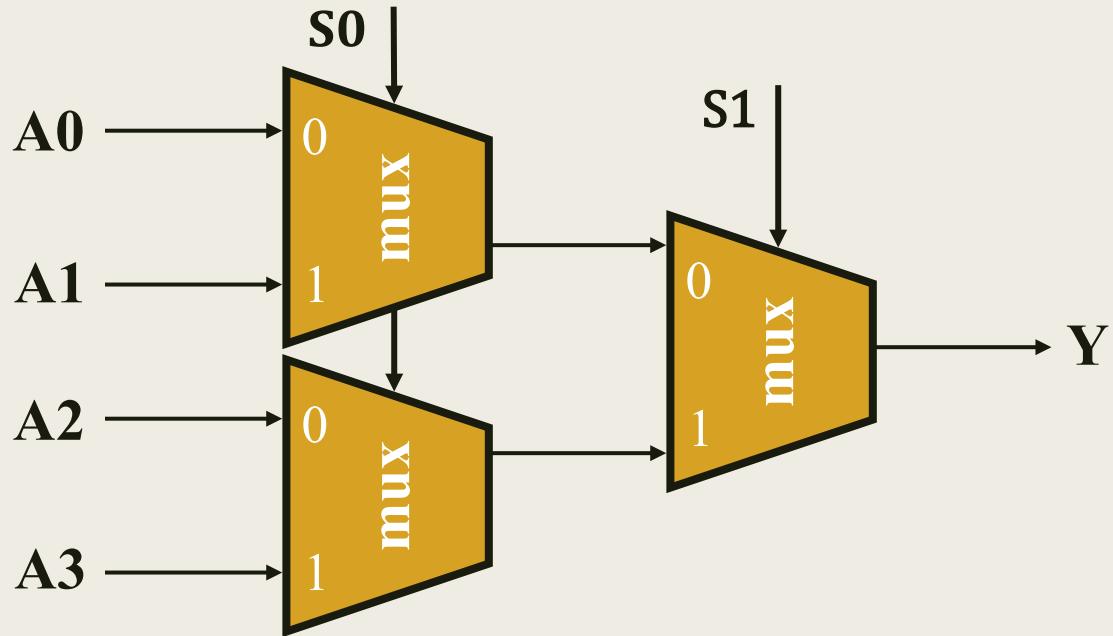


Η οντότητα του πολυπλέκτη 4 σε 1 στη VHDL

```
entity MUX_4_to_1 is
  port (
    A0: in STD_LOGIC;
    A1: in STD_LOGIC;
    A2: in STD_LOGIC;
    A3: in STD_LOGIC;
    S0: in STD_LOGIC;
    S1: in STD_LOGIC;
    Y: out STD_LOGIC);
end MUX_4_to_1;
```



Λύση 2



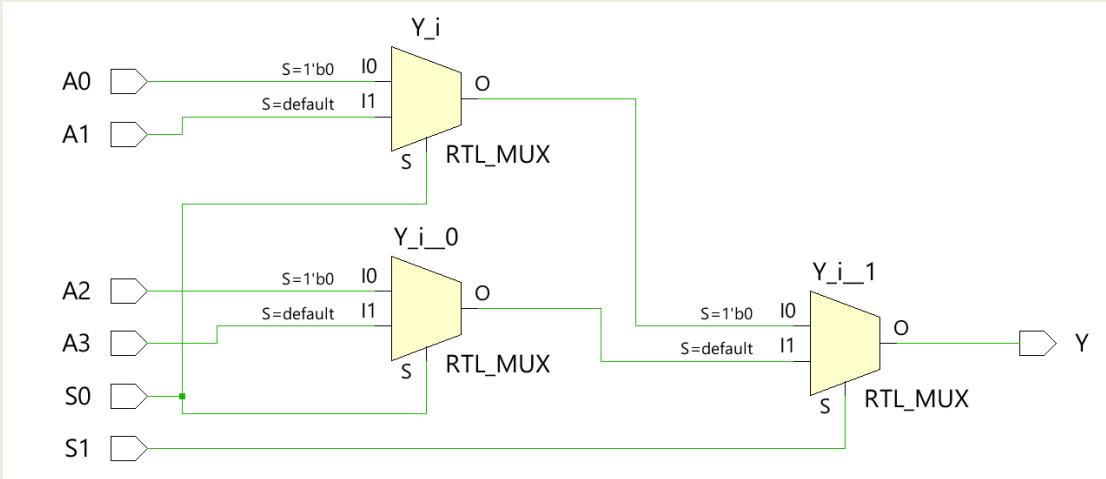
Η αρχιτεκτονική του πολυπλέκτη 4 σε 1 στη VHDL Περιγραφή συμπεριφοράς – Λύση 2

```
architecture BEHAVIORAL of MUX_4_to_1 is
begin
  process (A0, A1, A2, A3, S0, S1)
  begin
    if (S1 = '0') then
      if (S0 = '0') then Y <= A0;
      else Y <= A1;
      end if;
    else
      if (S0 = '0') then Y <= A2;
      else Y <= A3;
      end if;
    end if;
  end process;
end BEHAVIORAL;
```

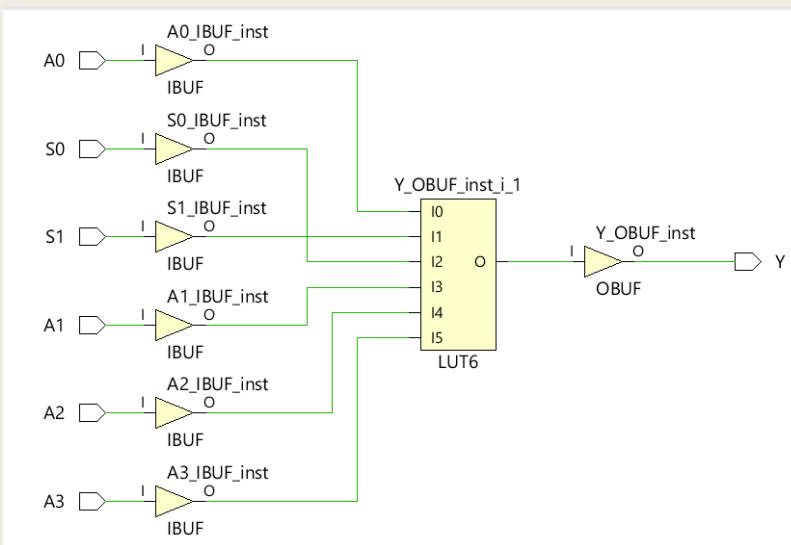
Στη λίστα ευαισθησίας συμπεριλαμβάνονται όλες
οι είσοδοι του συνδυαστικού κυκλώματος

Η αρχιτεκτονική του πολυπλέκτη 4 σε 1 στη VHDL Περιγραφή συμπεριφοράς – Λύση 2

■ Σχηματικό διάγραμμα RTL



■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



Και οι 2 λύσεις έχουν
την ίδια υλοποίηση σε
τεχνολογία FPGA με LUT6

Επιλεγμένη άσκηση: συνδυαστική λογική σε περιγραφή συμπεριφοράς στη VHDL

- Ποιος είναι ο **πίνακας αλήθειας** και η **εξίσωση Boole** του συνδυαστικού κυκλώματος, του οποίου η συμπεριφορά περιγράφεται στη VHDL ως εξης;

```
entity Exercise is port (
    A,B,C,D: in STD_LOGIC;
    Y: out STD_LOGIC);
end Exercise;
architecture BEHAVIORAL of Exercise is
begin
    process (A,B,C,D) begin
        if ((A = '0') and (B = '0')) then Y <= '1';
        elsif (C = D) then Y <= '1';
        else Y <= '0';
        end if;
    end process;
end BEHAVIORAL;
```

Επιλεγμένη άσκηση: συνδυαστική λογική σε περιγραφή συμπεριφοράς στη VHDL

- Πίνακας αλήθειας

```
if ((A = '0') and (B = '0')) then Y <= '1';
```

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	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	0

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	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	0

A	B	C	D	Y
0	0	0	0	1
0	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
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

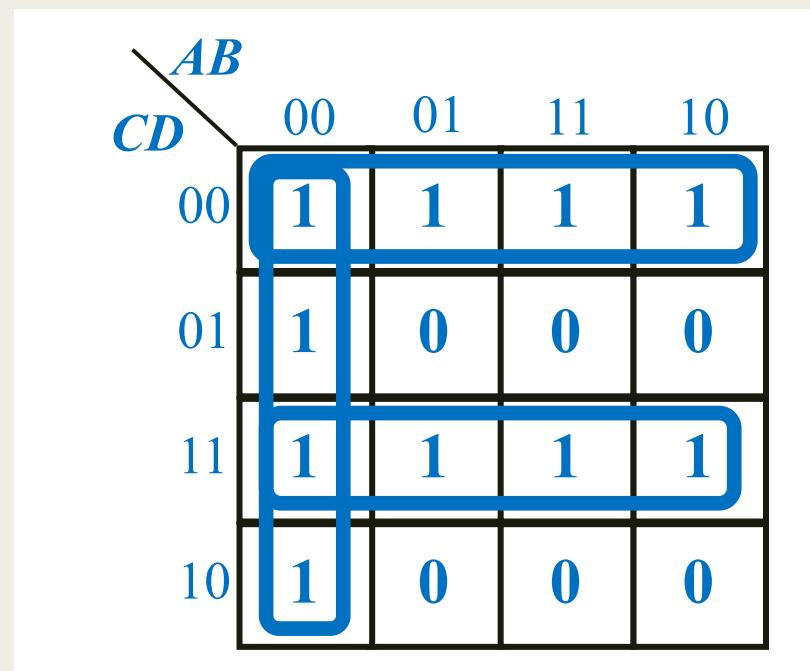
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

```
elsif (C = D) then Y <= '1';
```

Επιλεγμένη άσκηση: συνδυαστική λογική σε περιγραφή συμπεριφοράς στη VHDL

- Πίνακας αλήθειας

A B C D	Y	A B C D	Y
0 0 0 0	1	1 0 0 0	1
0 0 0 1	1	1 0 0 1	0
0 0 1 0	1	1 0 1 0	0
0 0 1 1	1	1 0 1 1	1
0 1 0 0	1	1 1 0 0	1
0 1 0 1	0	1 1 0 1	0
0 1 1 0	0	1 1 1 0	0
0 1 1 1	1	1 1 1 1	1



- Εξίσωση Boole

$$Y = \bar{A} \bar{B} + \bar{C} \bar{D} + C D$$

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση σημάτων

- Η ενημέρωση των σημάτων εξόδου (ή εσωτερικών σημάτων) με τη νέα τους τιμή γίνεται στο **τέλος του process** με **καθυστέρηση δέλτα δ_{delay}**
- Μέχρι το τέλος του process τα σήματα «θυμούνται» την τρέχουσα τιμή τους, δηλαδή τα **σήματα έχουν μνήμη μέσα στο process**
- Εάν μέσα σε ένα process γίνεται ανάθεση τιμών με πολλές εντολές στο ίδιο σήμα εξόδου (ή εσωτερικό σήμα), **μόνο η τελευταία εντολή ανάθεσης τιμής λαμβάνεται υπόψη**
- Εάν μέσα σε ένα process γίνεται ανάθεση τιμής σε ένα **εσωτερικό σήμα**, που στη συνέχεια χρησιμοποιείται και ως **είσοδος στο ίδιο process**, τότε αυτό το **εσωτερικό σήμα** πρέπει να δηλώνεται και στη **λίστα ευαισθησίας του process**
 - για να συμφωνούν οι προσομοιώσεις πριν και μετά τη σύνθεση

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση εσωτερικών σημάτων

```
entity XOR_single is port (
    A, B, C : in STD_LOGIC;
    X, Y : out STD_LOGIC);
end XOR_single;
architecture beh of XOR_sig is
signal D : STD_LOGIC;
begin
process (A,B,C,D)
begin
    D <= A; -- ignored
    X <= C xor D;
    D <= B; -- considered
    Y <= C xor D;
end process;
end beh;
```

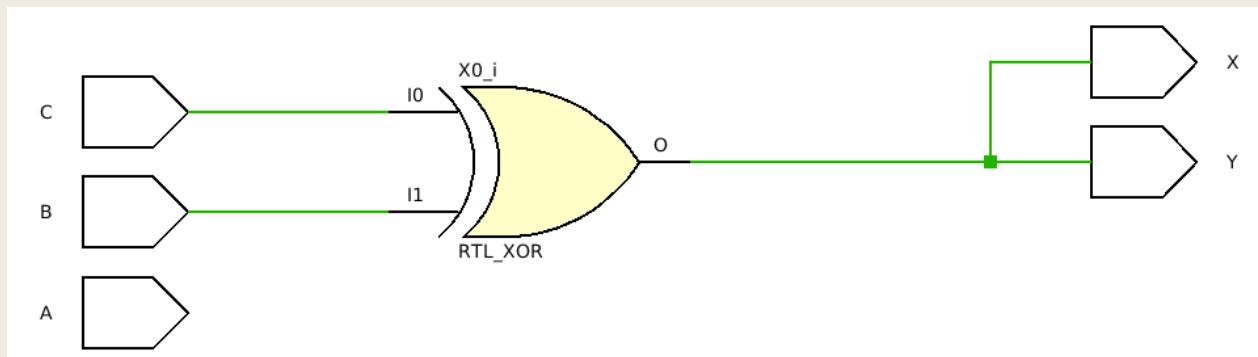
Ανάθεση τιμών στο
εσωτερικό σήμα D
με δύο εντολές

Το εσωτερικό σήμα D δηλώνεται
στη λίστα ευαισθησίας

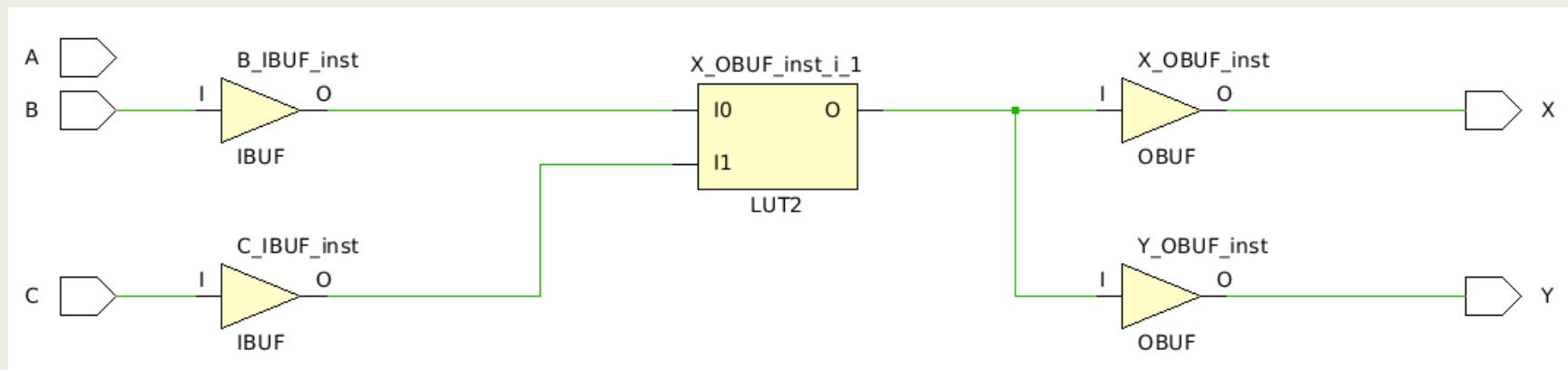
- Κατά τη σύνθεση υλοποιείται η εξίσωση Boole:
 - $X = Y = C \text{ xor } B$

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση εσωτερικών σημάτων

- Σχηματικό διάγραμμα RTL

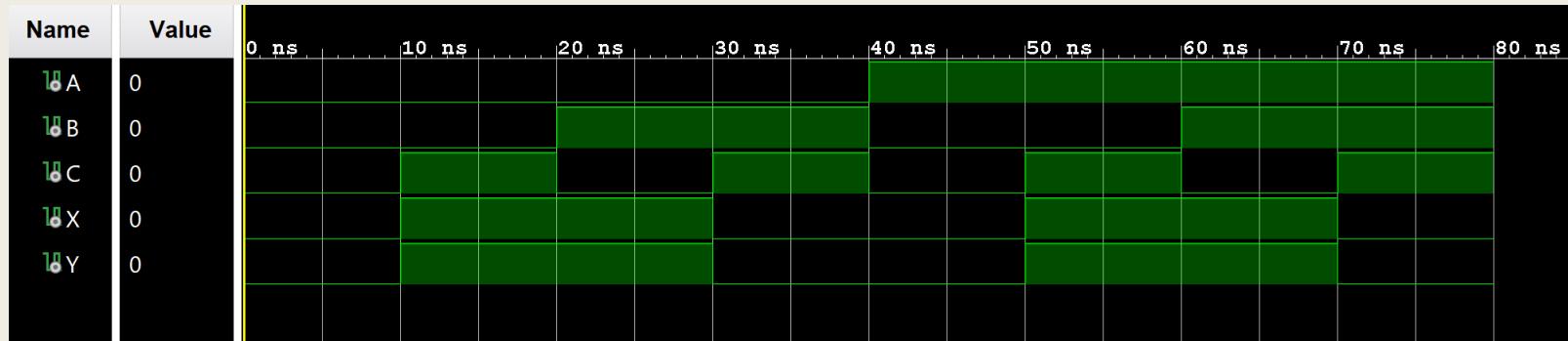


- Σχηματικό διάγραμμα σε τεχνολογία FPGA

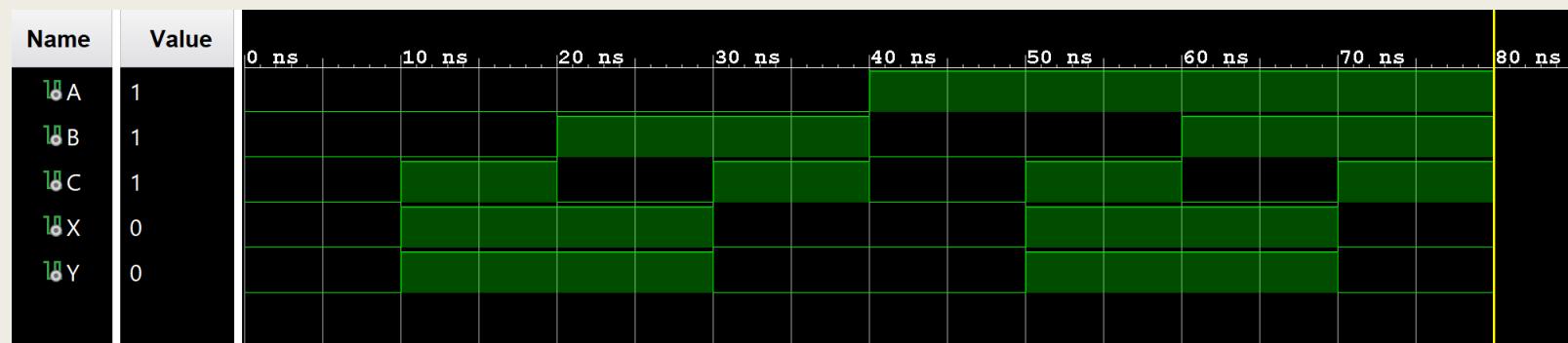


Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση εσωτερικών σημάτων

- Behavioral simulation (του VHDL κώδικα, πριν τη σύνθεση)



- Post-implementation functional simulation (μετά τη σύνθεση και την υλοποίηση)



Προσοχή: Συμφωνία στην προσομοίωση! Λόγω ορθής δήλωσης
του εσωτερικού σήματος D στη λίστα ευαισθησίας

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση εσωτερικών σημάτων

```
entity XOR_single is port (
    A, B, C : in STD_LOGIC;
    X, Y : out STD_LOGIC);
end XOR_single;
architecture beh of XOR_sig is
signal D : STD_LOGIC;
begin
process (A,B,C)
begin
    D <= A; -- ignored
    X <= C xor D;
    D <= B; -- considered
    Y <= C xor D;
end process;
end beh;
```

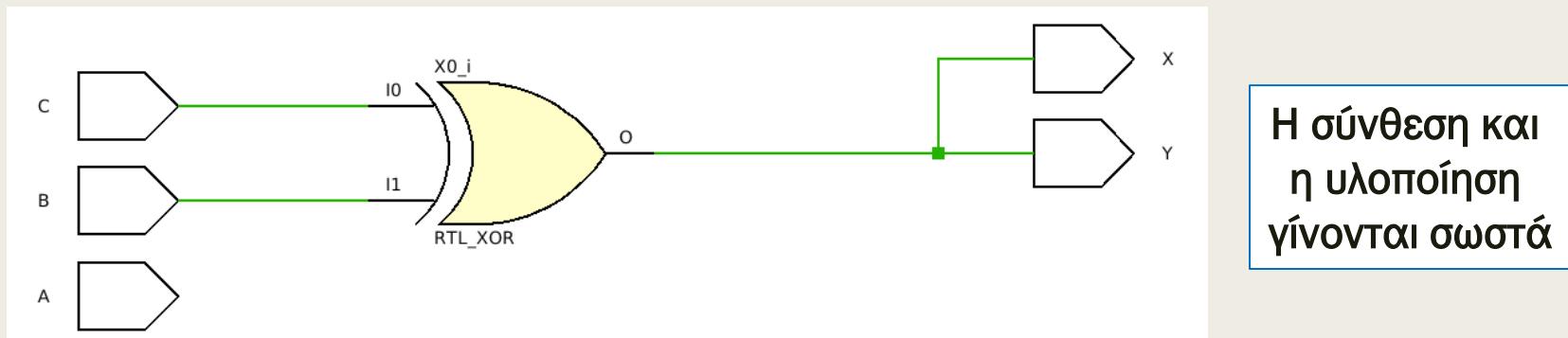
Ανάθεση τιμών στο
εσωτερικό σήμα D
με δύο εντολές

Το εσωτερικό σήμα D **δεν** δηλώνεται
στη λίστα ευαισθησίας

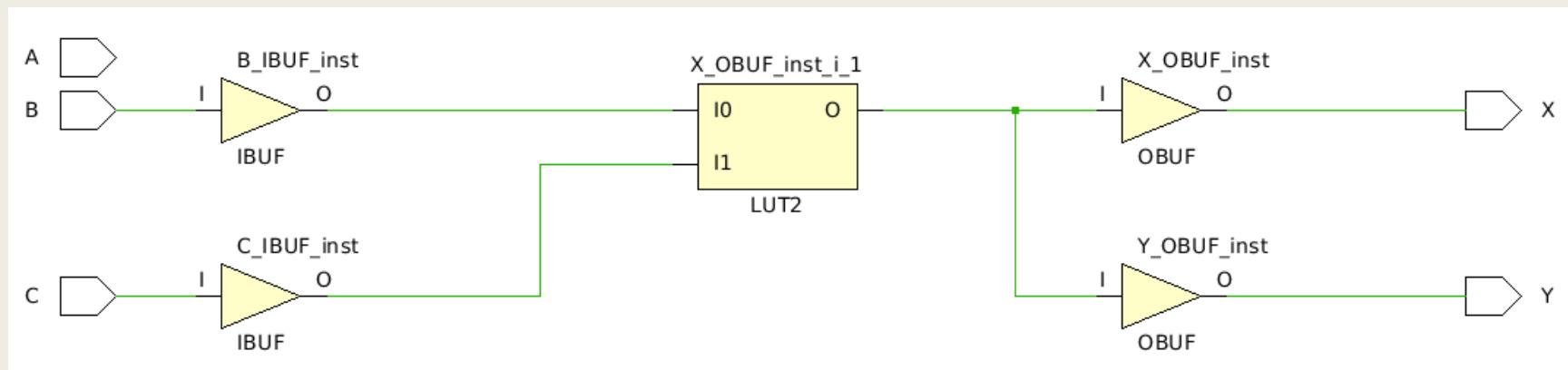
- Κατά τη σύνθεση υλοποιείται η εξίσωση Boole:
 - $X = Y = C \text{ xor } B$

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση εσωτερικών σημάτων

- Σχηματικό διάγραμμα RTL

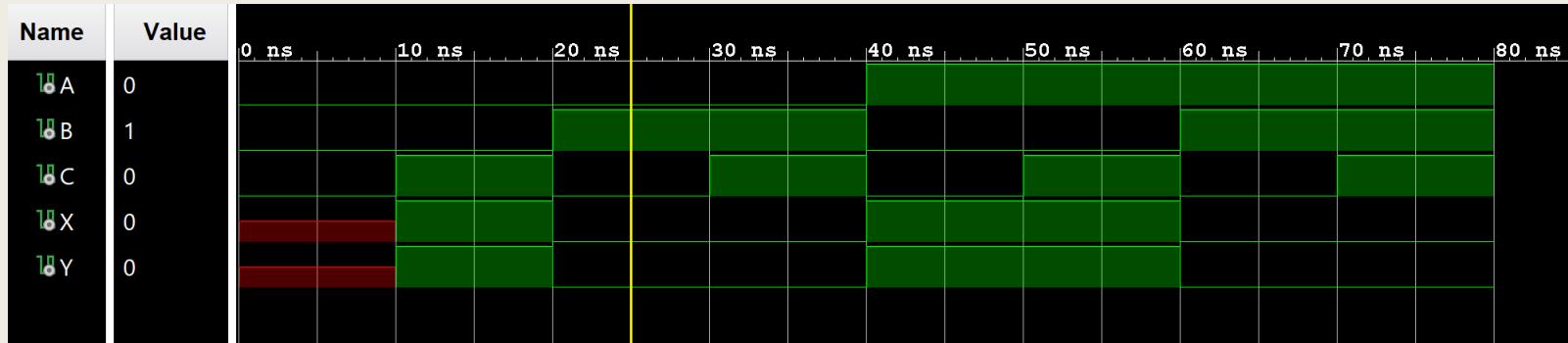


- Σχηματικό διάγραμμα σε τεχνολογία FPGA

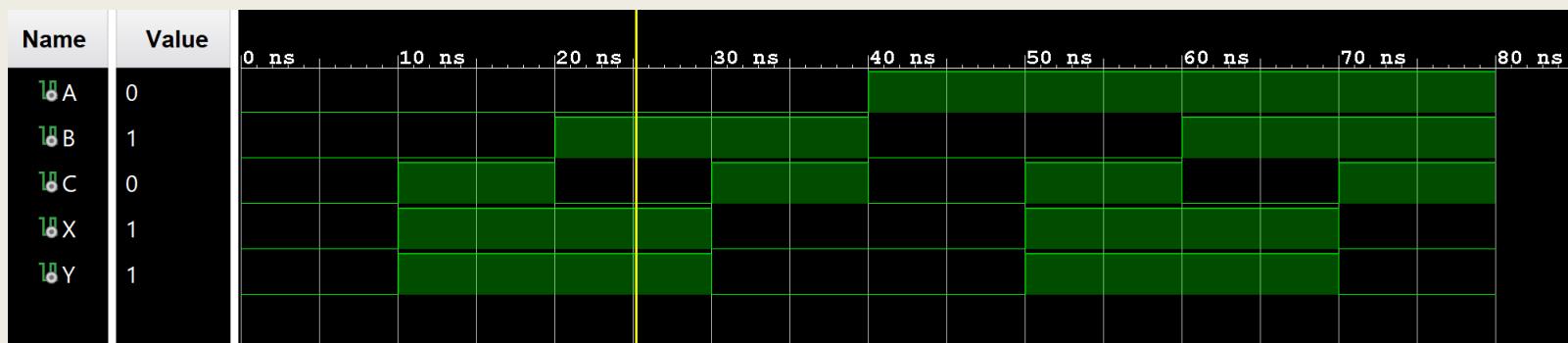


Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση εσωτερικών σημάτων

- Behavioral simulation (του VHDL κώδικα, πριν τη σύνθεση) – **Λάθος!**



- Post-implementation functional simulation (μετά τη σύνθεση και την υλοποίηση) – Σωστό



Προσοχή: Ασυμφωνία στην προσομοίωση! Λόγω **μη δήλωσης** του εσωτερικού σήματος D στη λίστα ευαισθησίας

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση μεταβλητών

- Η ενημέρωση των μεταβλητών γίνεται αμέσως μόλις εκτελεσθεί η αντίστοιχη εντολή ανάθεσης τιμής στη μεταβλητή εντός του process
- Η μεταβλητή διατηρεί την τιμή της μέχρι να προσδιορισθεί άλλη τιμή σε μία επόμενη εντολή ανάθεσης τιμής στην ίδια μεταβλητή εντός του process

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση μεταβλητών

```
entity XOR_var is port (
    A, B, C: in STD_LOGIC;
    X, Y: out STD_LOGIC);
end XOR_var;
architecture beh of XOR_var is
begin
    process (A,B,C)
        variable D: STD_LOGIC;
    begin
        D := A; -- considered
        X <= C xor D;
        D := B; -- considered
        Y <= C xor D;
    end process;
end beh;
```

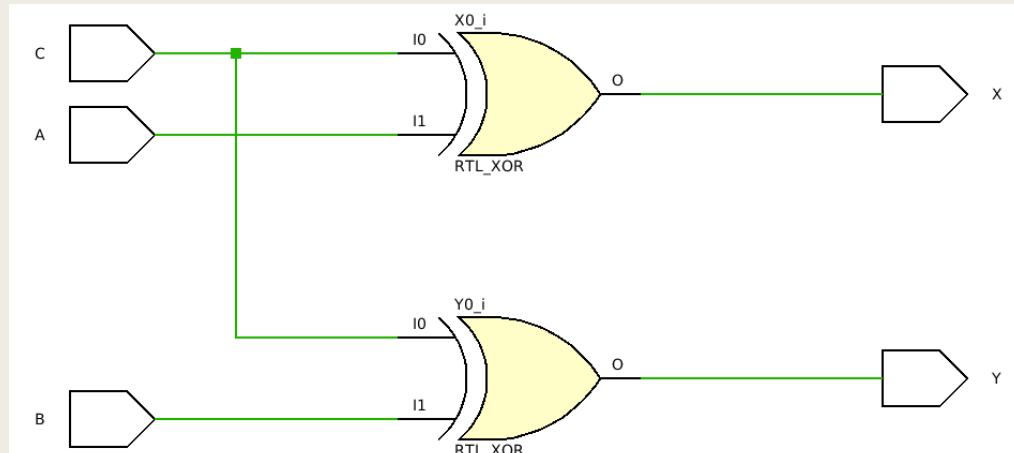
Η μεταβλητή D λαμβάνει άμεσα την τιμή της εισόδου A

Η μεταβλητή D λαμβάνει άμεσα την τιμή της εισόδου B

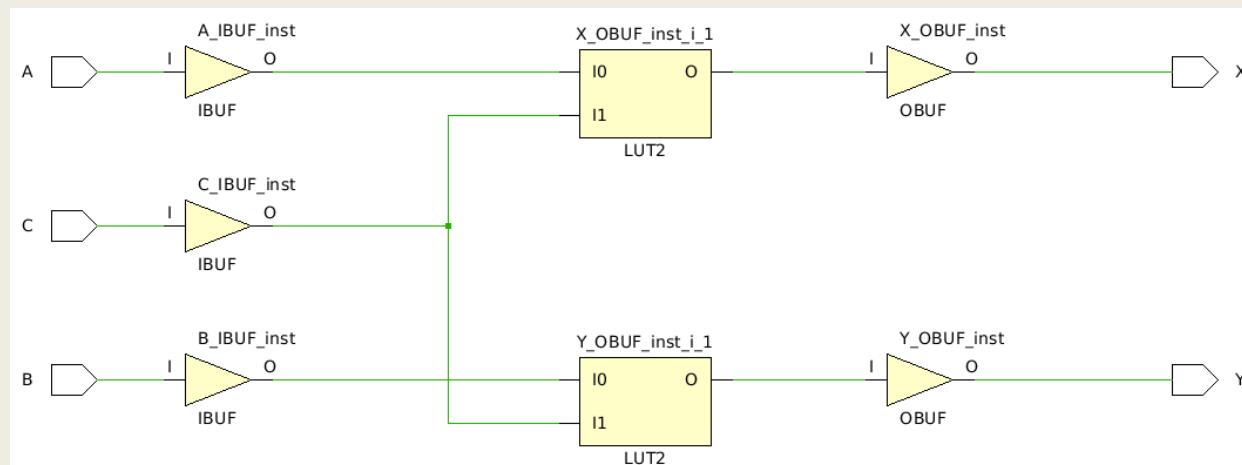
- Κατά τη σύνθεση υλοποιούνται οι εξισώσεις Boole:
 - $X = C \text{ xor } A$ και $Y = C \text{ xor } B$

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση μεταβλητών

■ Σχηματικό διάγραμμα RTL



■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση σημάτων

```
entity LAST is
  port (
    A, B, selA, selB : in STD_LOGIC;
    C : out STD_LOGIC);
end LAST;
architecture LAST_BEH of LAST is
begin
  process (A, B, selA, selB) begin
    if (selA = '1') then
      C <= A;
    else
      C <= '0';
    end if;
    if (selB = '1') then
      C <= B;
    else
      C <= '0';
    end if;
  end process;
end LAST_BEH;
```

Ανάθεση τιμών στο
σήμα εξόδου C
με δύο εντολές IF

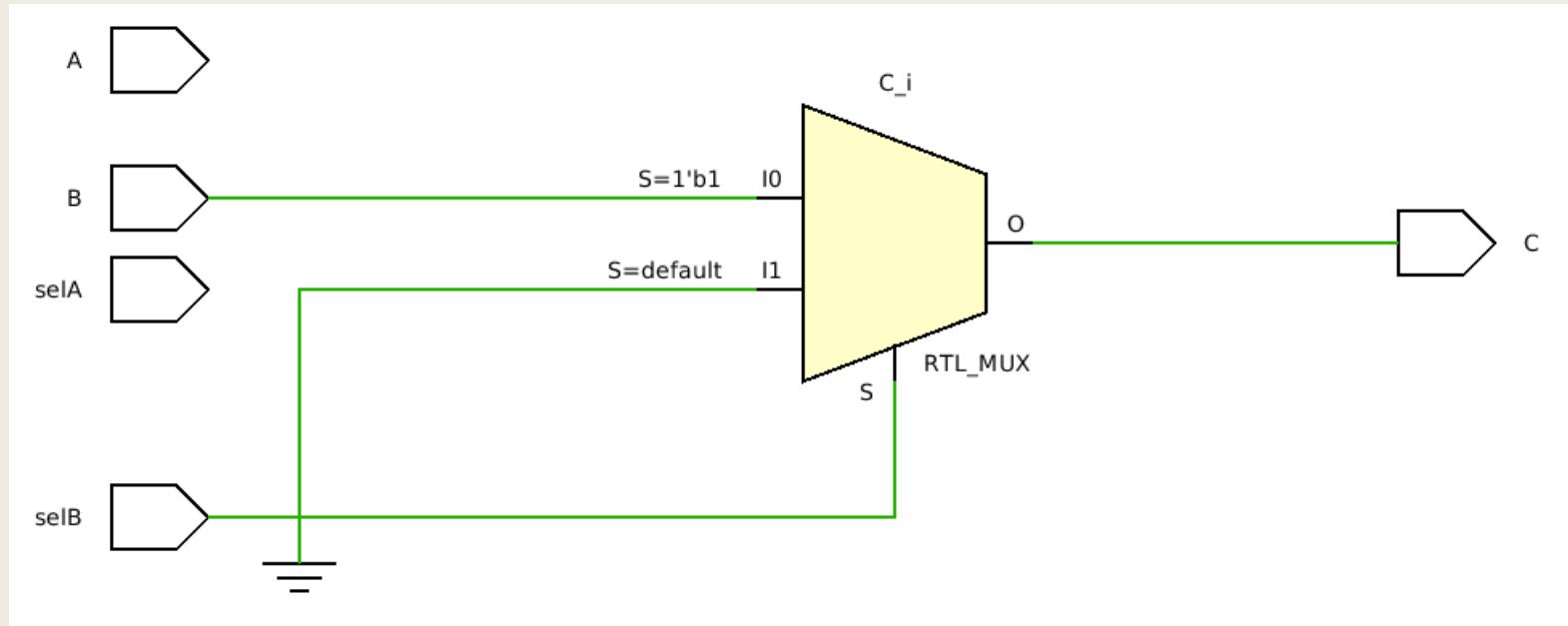
Δεν λαμβάνεται υπόψη

Λαμβάνεται υπόψη

- Ποιο είναι το αποτέλεσμα της σύνθεσης;

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση σημάτων

- Σχηματικό διάγραμμα RTL



- Εξίσωση Boole που υλοποιείται

$$C = B \text{ and } selB$$

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση σημάτων

- Τροποποιούμε τον κώδικα με τη χρήση του **elsif**

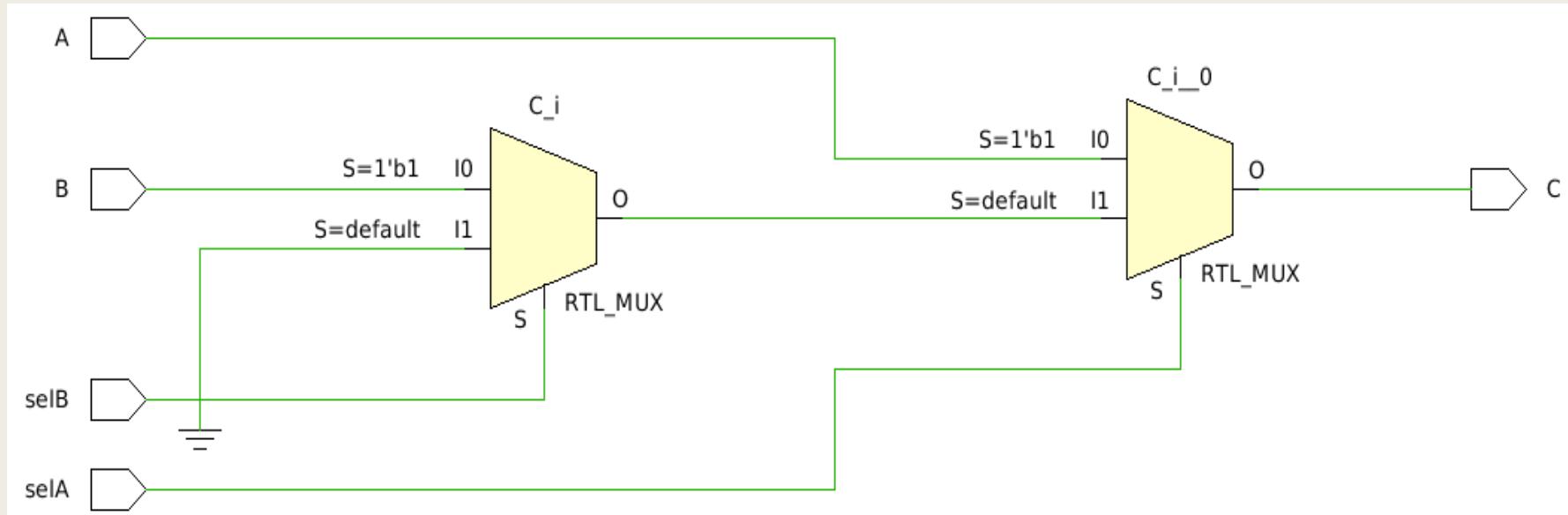
```
entity LAST is
  port (
    A, B, selA, selB: in STD_LOGIC;
    C: out STD_LOGIC);
end LAST;
architecture LAST_BEH of LAST is
begin
  process (A, B, selA, selB)
  begin
    if (selA = '1') then
      C <= A;
    elsif (selB = '1') then
      C <= B;
    else
      C <= '0';
    end if;
  end process;
end LAST_BEH;
```

Η χρήση του **elsif** επιτρέπει
την ανάθεση τιμών
στο σήμα εξόδου C
με μία μόνο εντολή IF

- Ποιο είναι το αποτέλεσμα της σύνθεσης μετά την τροποποίηση;

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Χρήση σημάτων

- Σχηματικό διάγραμμα RTL



- Εξίσωση Boole που υλοποιείται

$$C = (B \text{ and } \overline{\text{selB}} \text{ and } \overline{\text{selA}}) \text{ or } (A \text{ and } \text{selA})$$

Περιγραφή συμπεριφοράς (behavioral) στη VHDL – Συμπεράσματα στη χρήση σημάτων και μεταβλητών

- **Προσοχή!** Να μην γίνεται ανάθεση τιμών με πολλές εντολές στο ίδιο σήμα εξόδου (ή εσωτερικό σήμα) μέσα σε ένα process
- **Προσοχή!** Στην ανάθεση τιμής εσωτερικού σήματος μέσα σε ένα process, όταν αυτό επαναχρησιμοποιείται στην έκφραση μίας εντολής ανάθεσης τιμής στο ίδιο process
 - Το εσωτερικό σήμα πρέπει να συμπεριλαμβάνεται στη **λίστα ευαισθησίας**, ώστε το process να εκτελεσθεί ξανά για να διορθωθεί το λανθασμένο αποτέλεσμα του behavioral simulation
- Προτιμούμε τη χρήση μεταβλητής, αντί για εσωτερικού σήματος, όταν αυτή επαναχρησιμοποιείται στην έκφραση μίας εντολής ανάθεσης τιμής στο ίδιο process
 - Δεν απαιτείται η διόρθωση στη λίστα ευαισθησίας
 - Ο χρόνος προσομοίωσης μειώνεται σημαντικά γιατί το process εκτελείται μόνο μια φορά

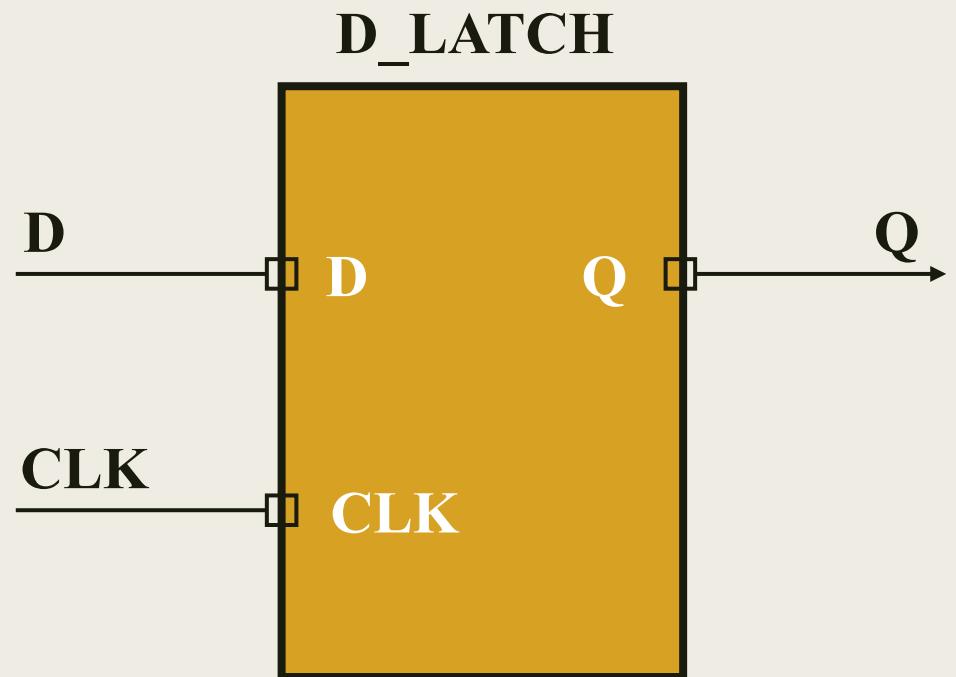
Περιγραφή ακολουθιακής λογικής στη VHDL

- Στην περιγραφή συμπεριφοράς οι ακολουθιακές εντολές ανάθεσης τιμής
 - ενημερώνουν τα σήματα εξόδου (ή τα εσωτερικά σήματα) με τη νέα τιμή στο **τέλος του process** με **καθυστέρηση δέλτα δ_{delay}**
 - μέχρι το τέλος της διεργασίας τα σήματα «θυμούνται» την τρέχουσα τιμή τους, δηλαδή τα **σήματα έχουν μνήμη μέσα στο process**
- Η δυνατότητα των σημάτων να «θυμούνται» την τρέχουσα τιμή τους μας επιτρέπει να περιγράψουμε **ακολουθιακή λογική** με τη χρήση μίας **εντολής IF** στην οποία υπάρχει **ελλιπής ανάθεση τιμής** σε ένα σήμα εξόδου (ή εσωτερικό σήμα)
 - στην εντολή IF δεν ορίζεται η τιμή του σήματος με εντολή ανάθεσης τιμής, όταν δεν ικανοποιείται η συνθήκη

```
if boolean_expression (condition) then
    sequential_statement_1;
end if;
```

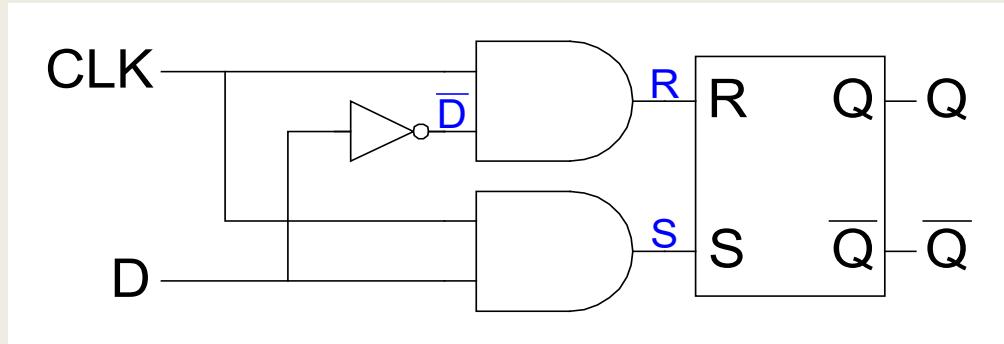
Η οντότητα του D Latch στη VHDL

```
entity D_LATCH is
port (
    CLK, D: in STD_LOGIC;
    Q: out STD_LOGIC);
end D_LATCH;
```



Η αρχιτεκτονική του D Latch στη VHDL

Περιγραφή συμπεριφοράς



CLK	D	Q	\bar{Q}
0	X	Q_{prev}	\bar{Q}_{prev}
1	0	0	1
1	1	1	0

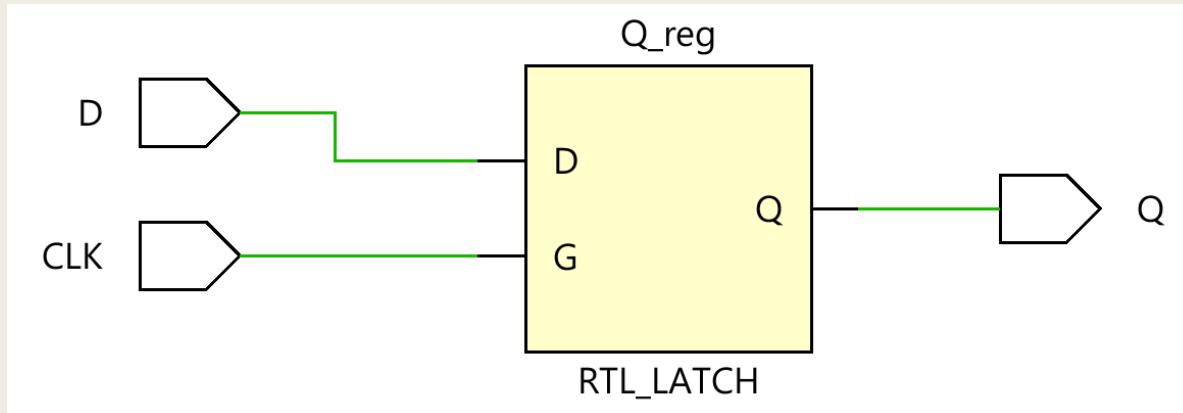
```
architecture BEHABIORAL of D_LATCH is
begin
  process (CLK, D)
  begin
    if (CLK = '1') then
      Q <= D;
    end if;           -- Το Q δεν ορίζεται για όλες τις τιμές του CLK
    end process;      -- Ελλιπής ανάθεση του Q (incomplete assignment)
  end BEHABIORAL;
```

Εκμεταλλευόμαστε την εσωτερική μνήμη των σημάτων

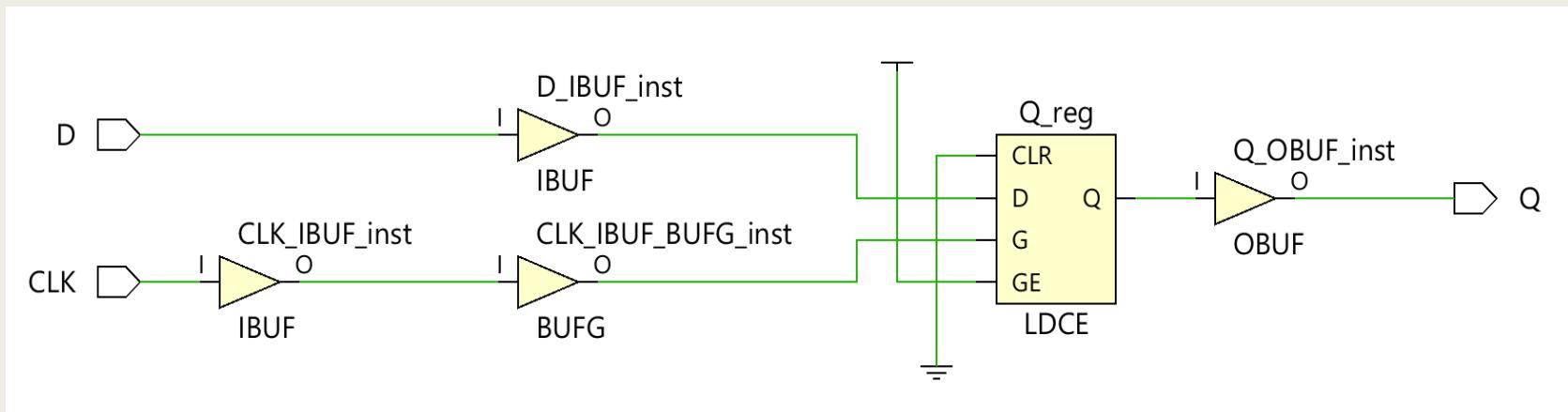
Η αρχιτεκτονική του D Latch στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA



Το πρόβλημα της ελλιπούς ανάθεσης τιμής στη VHDL

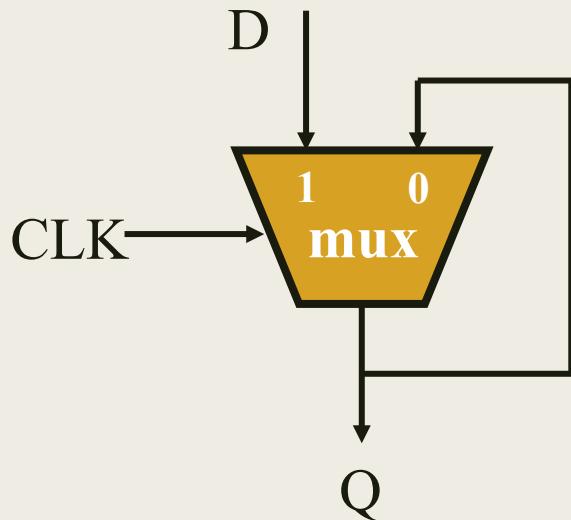
- Ανεπιθύμητη εμφάνιση ενός D-Latch λόγω ελλιπούς ανάθεσης τιμής
 - όταν για ένα σήμα δεν ορίζεται μία εντολή ανάθεσης τιμής σε όλες τις διακλαδώσεις μίας εντολής IF, υπάρχει το ενδεχόμενο να εμφανισθεί μετά τη σύνθεση ένα ανεπιθύμητο D-Latch για το συγκεκριμένο σήμα
 - η ελλιπής ανάθεση οδηγεί στην υλοποίηση επιπλέον λογικής που συνήθως είναι πλεονάζουσα
- Για να αποφευχθεί η ελλιπής ανάθεση τιμής κατά τη σύνθεση συνδυαστικής λογικής, για κάθε σήμα:
 - βάζουμε μία αρχική τιμή, και
 - ορίζουμε μία εντολή ανάθεσης τιμής ή την εντολή null, (που σημαίνει «μην κάνεις τίποτα – διατήρησε την τρέχουσα τιμή των σημάτων»), σε όλες τις διακλαδώσεις μίας εντολής IF

Υλοποίηση ελλιπούς ανάθεσης τιμής

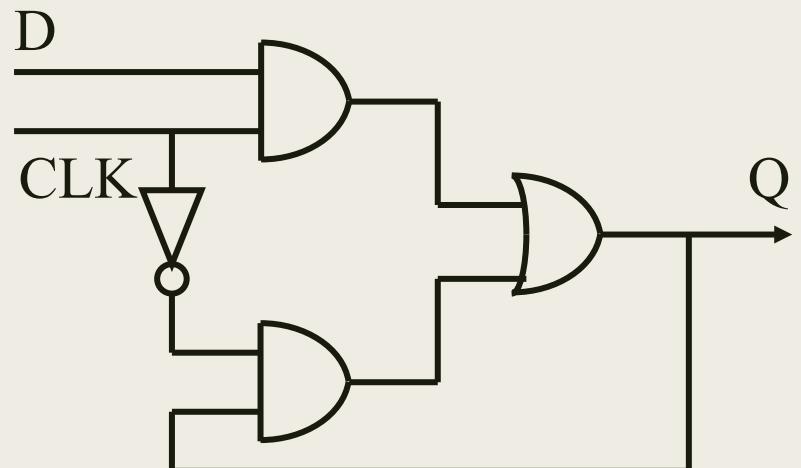
Ακολουθιακή λογική

D-Latch

```
process (CLK, D)
begin
    if (CLK = '1') then
        Q <= D;
    end if;
end process;
```



Σε επίπεδο πολυπλέκτη

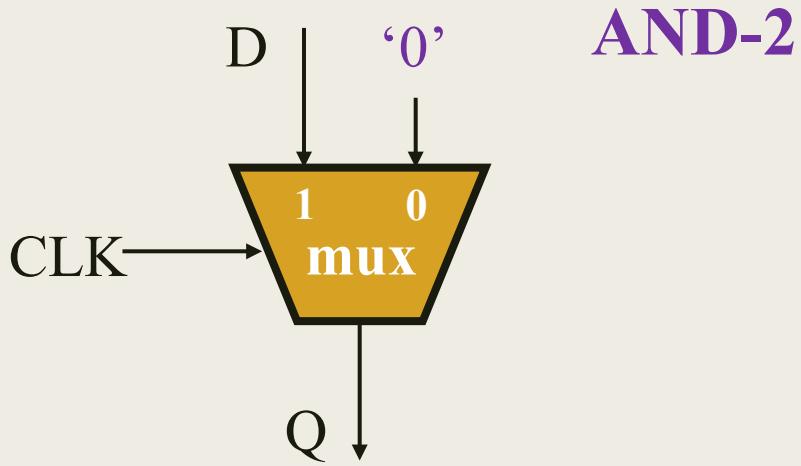


Σε επίπεδο πύλης

Υλοποίηση μη ελλιπούς ανάθεσης τιμής

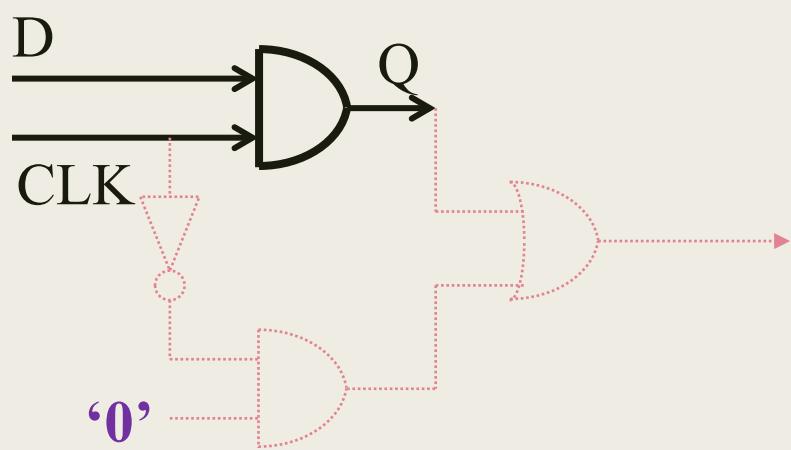
Συνδυαστική λογική

```
process (CLK, D)
begin
    if (CLK = '1') then
        Q <= D;
    else
        Q <= '0';
    end if;
end process;
```



Σε επίπεδο πολυπλέκτη

```
process (CLK, D)
begin
    Q <= '0';
    if (CLK = '1') then
        Q <= D;
    else
        null;
    end if;
end process;
```



Σε επίπεδο πύλης

Υλοποίηση μη ελλιπούς ανάθεσης τιμής

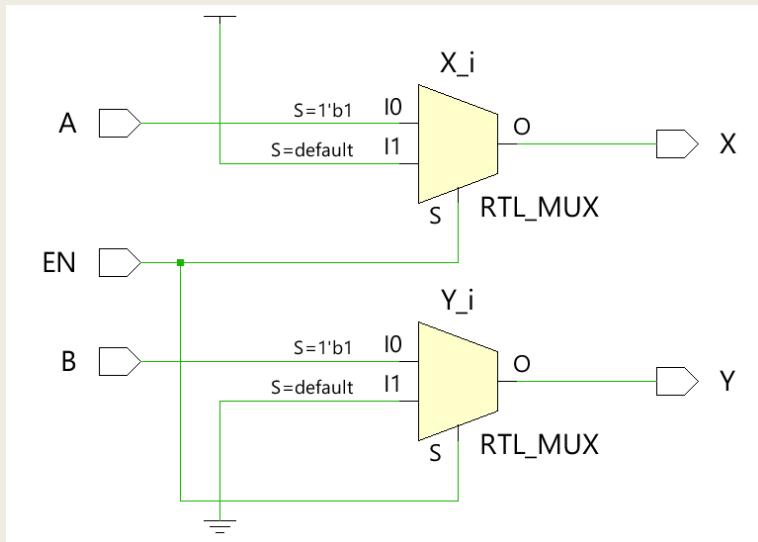
- Παράδειγμα αποφυγής εμφάνισης ελλιπούς ανάθεσης τιμής

```
architecture EXAMPLE_BEH of EXAMPLE is
begin
  process (EN, A, B)
    begin
      X <= '0'; -- αρχικές τιμές
      Y <= '0';
      if (EN = '1') then
        X <= A;
        Y <= B;
      else
        X <= '1';
      end if;
    end process;
  end EXAMPLE_BEH;
```

Στην περίπτωση που δεν ικανοποιείται η συνθήκη $EN = 1$, η έξοδος X λαμβάνει την τιμή 1, ενώ η έξοδος Y διατηρεί την τιμή 0 που έλαβε κατά την αρχικοποίηση των εξόδων. Λόγω της αρχικοποίησης των εξόδων, δεν υλοποιείται ακολουθιακή, αλλά συνδυαστική λογική. Εάν δεν υπήρχε η αρχικοποίηση των εξόδων, θα υλοποιείτο ένα D Latch για την έξοδο Y .

Υλοποίηση μη ελλιπούς ανάθεσης τιμής

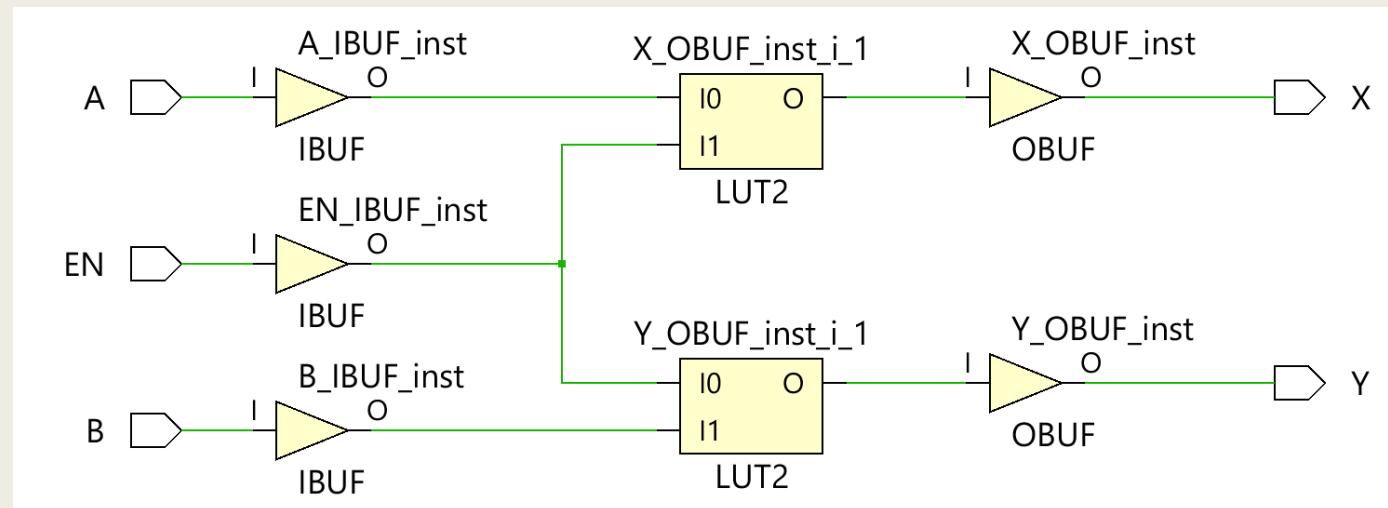
■ Σχηματικό διάγραμμα RTL



$$X = ENA + \overline{EN} \quad 1 = A + \overline{EN}$$

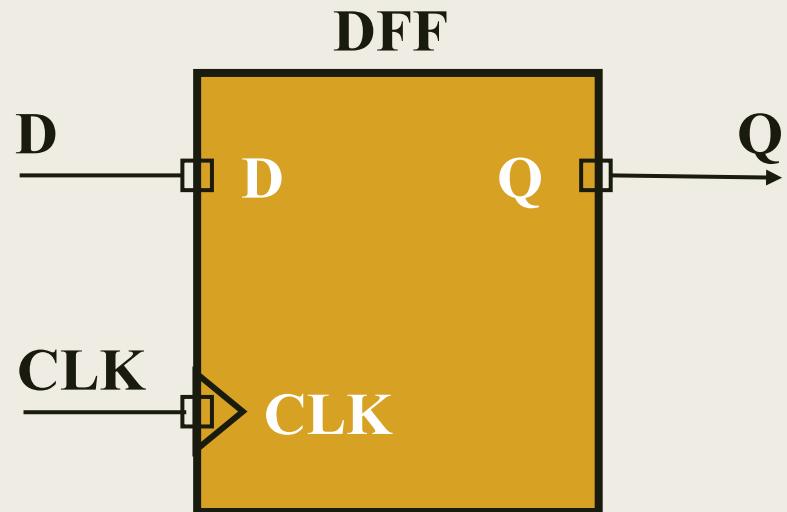
$$Y = ENB + \overline{EN} \quad 0 = ENB$$

■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



Η οντότητα του D Flip-Flop στη VHDL

```
entity DFF is
port (
    CLK, D: in STD_LOGIC;
    Q: out STD_LOGIC);
end DFF;
```



- Ο κώδικας του D Flip-Flop είναι σχεδόν ίδιος με τον κώδικα του D Latch
- Διαφοροποιούνται μόνο στη συνθήκη του CLK
 - Στον κώδικα του D Flip-Flop χρησιμοποιείται επιπλέον το **event attribute (CLK'event)**, που λαμβάνει την τιμή TRUE όταν το σήμα CLK αλλάζει τιμή ($0 \rightarrow 1$ ή $1 \rightarrow 0$)
 - Η ανερχόμενη ακμή του CLK περιγράφεται ως
 - **CLK = '1' and CLK'event**
 - Η κατερχόμενη ακμή του CLK περιγράφεται ως
 - **CLK = '0' and CLK'event**

Η αρχιτεκτονική του D Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

```
architecture BEHAVIORAL of DFF is
begin
  process (CLK)
    begin
      if (CLK = '1' and CLK'event) then
        Q <= D;
      end if;
    end process;
end DFF_BEH;
```

Το D δεν
τοποθετείται
στη λίστα
ευαισθησίας

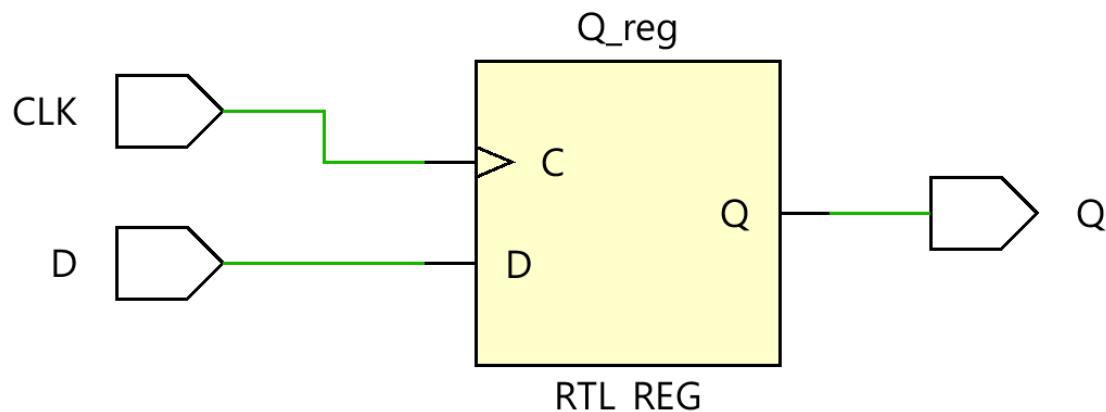
Η συνθήκη του
σύγχρονου D
εξετάζεται μετά
τη συνθήκη του
CLK

Προσοχή! Η απουσία του CLK'event οδηγεί στην υλοποίηση ενός D-Latch

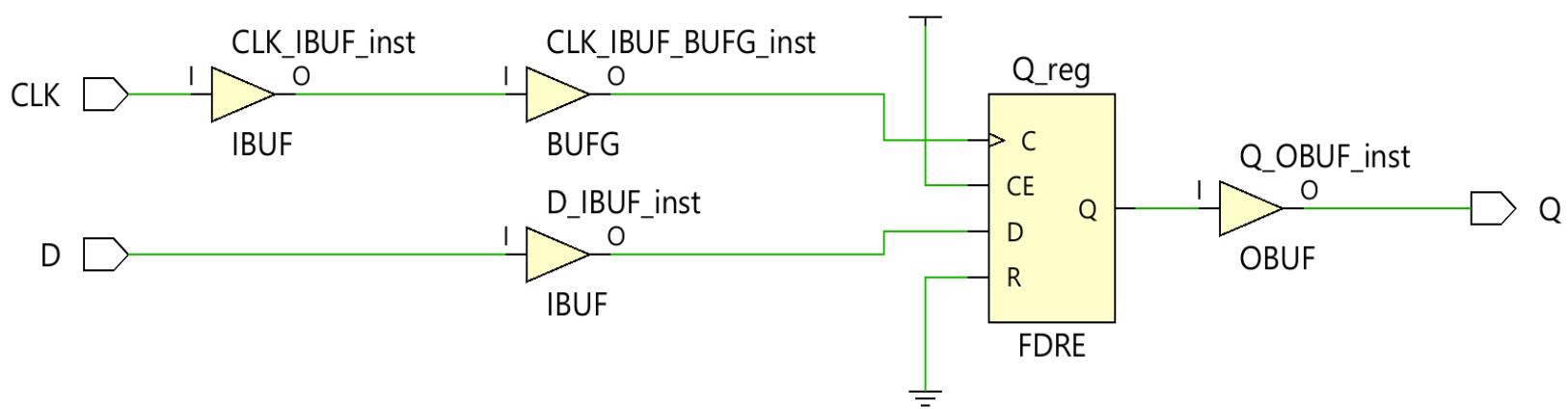
Η αρχιτεκτονική του D Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA



D Latch vs D Flip-Flop στη VHDL

```
architecture BEHAVIORAL of D_LATCH is
begin
  process (CLK, D)
  begin
    if (CLK = '1') then
      Q <= D;
    end if;
  end process;
end BEHAVIORAL;
```

```
architecture BEHAVIORAL of DFF is
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      Q <= D;
    end if;
  end process;
end DFF_BEH;
```

D Flip-Flop με 2 εξόδους στη VHDL

Περιγραφή συμπεριφοράς

- Υλοποίηση δύο εξόδων (Q και QN) με δύο D Flip-Flop

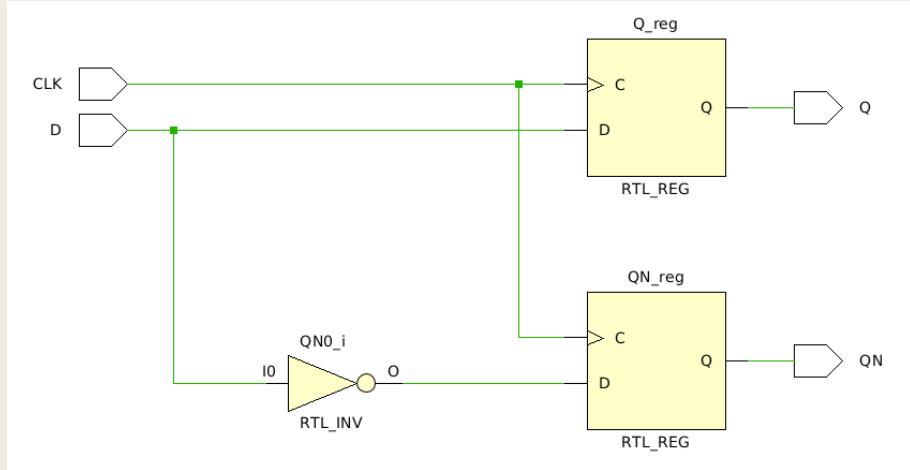
```
entity DFF is
port (
    CLK, D: in STD_LOGIC;
    Q, QN: out STD_LOGIC);
end DFF;
architecture DFF_BEH of DFF is
begin
process (CLK)
begin
    if (CLK = '1' and CLK'event) then
        Q <= D; QN <= not D;
    end if;
end process;
end DFF_BEH;
```

Ανάθεση τιμής
στα Q και QN
εντός process

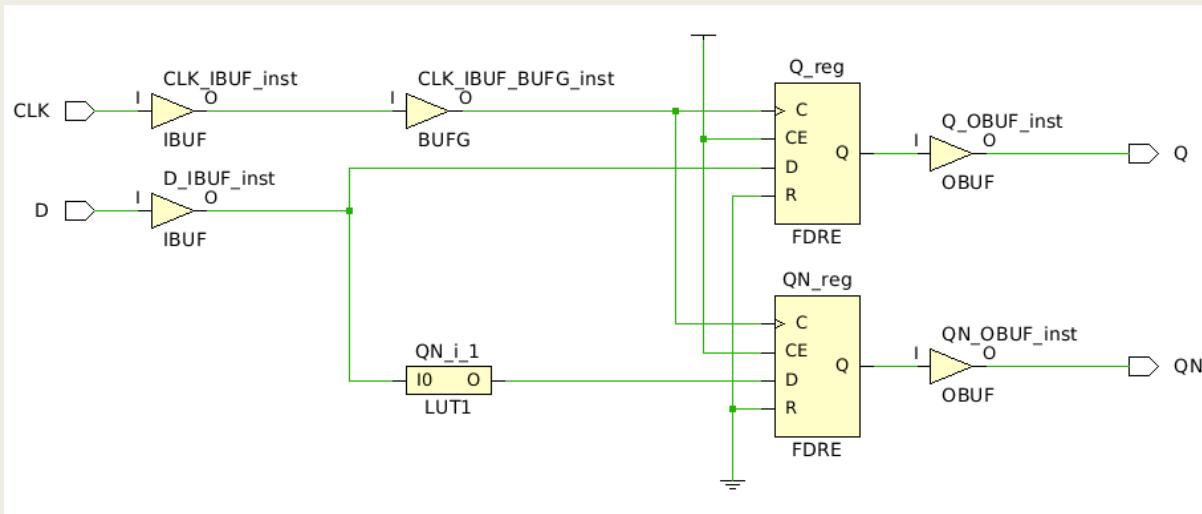
D Flip-Flop με 2 εξόδους στη VHDL

Περιγραφή συμπεριφοράς με 2 D F/F

■ Σχηματικό διάγραμμα RTL



■ Σχηματικό διάγραμμα σε τεχνολογία FPGA



D Flip-Flop με 2 εξόδους στη VHDL

Περιγραφή συμπεριφοράς

- Υλοποίηση δύο εξόδων (Q και QN) με ένα D Flip-Flop

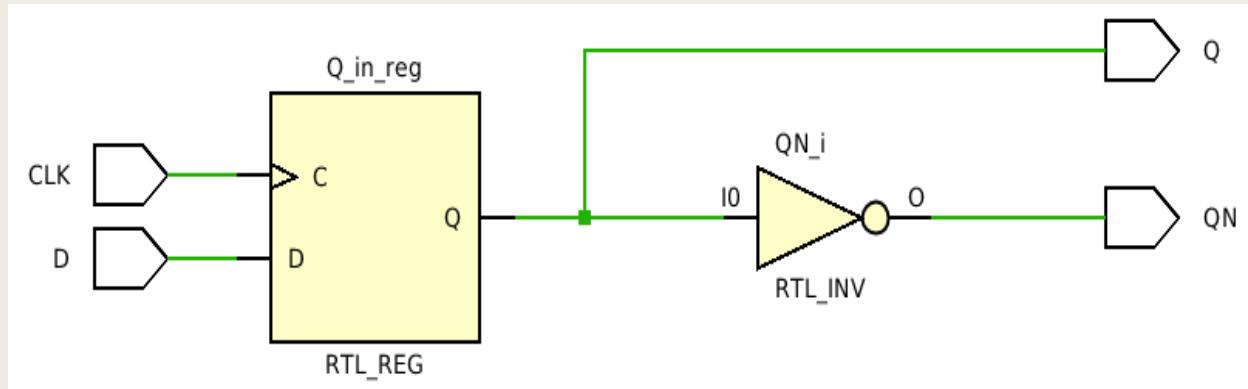
```
entity DFFwQN is
  port (
    CLK, D: in STD_LOGIC;
    Q, QN: out STD_LOGIC);
end DFFwQN;
architecture DFFwQN_BEH of DFFwQN is
  signal Q_in: STD_LOGIC;
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      Q_in <= D;
    end if;
  end process;
  Q <= Q_in; QN <= not Q_in;
end DFFwQN_BEH;
```

Ανάθεση τιμής
στα Q και QN
εκτός process

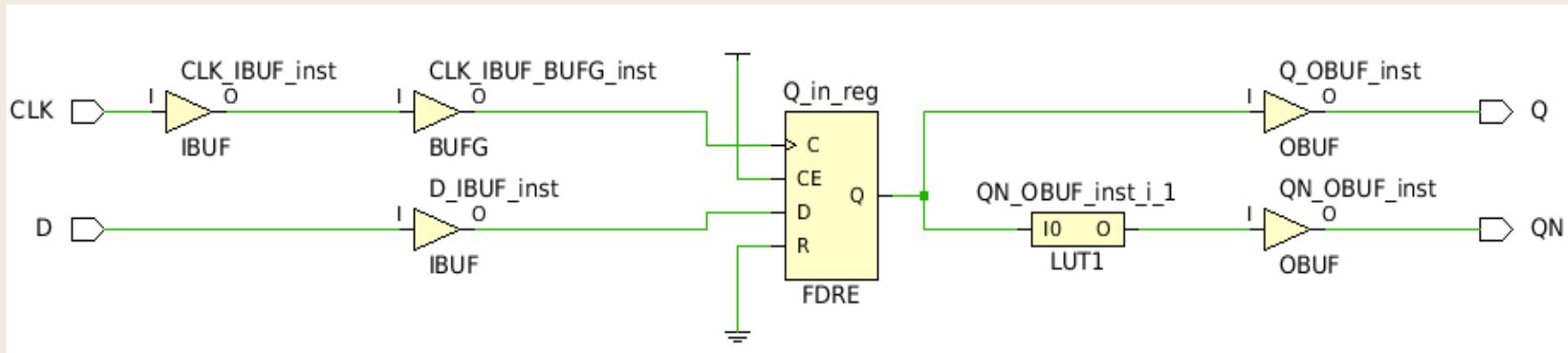
D Flip-Flop με 2 εξόδους στη VHDL

Περιγραφή συμπεριφοράς με 1 D F/F

- Σχηματικό διάγραμμα RTL

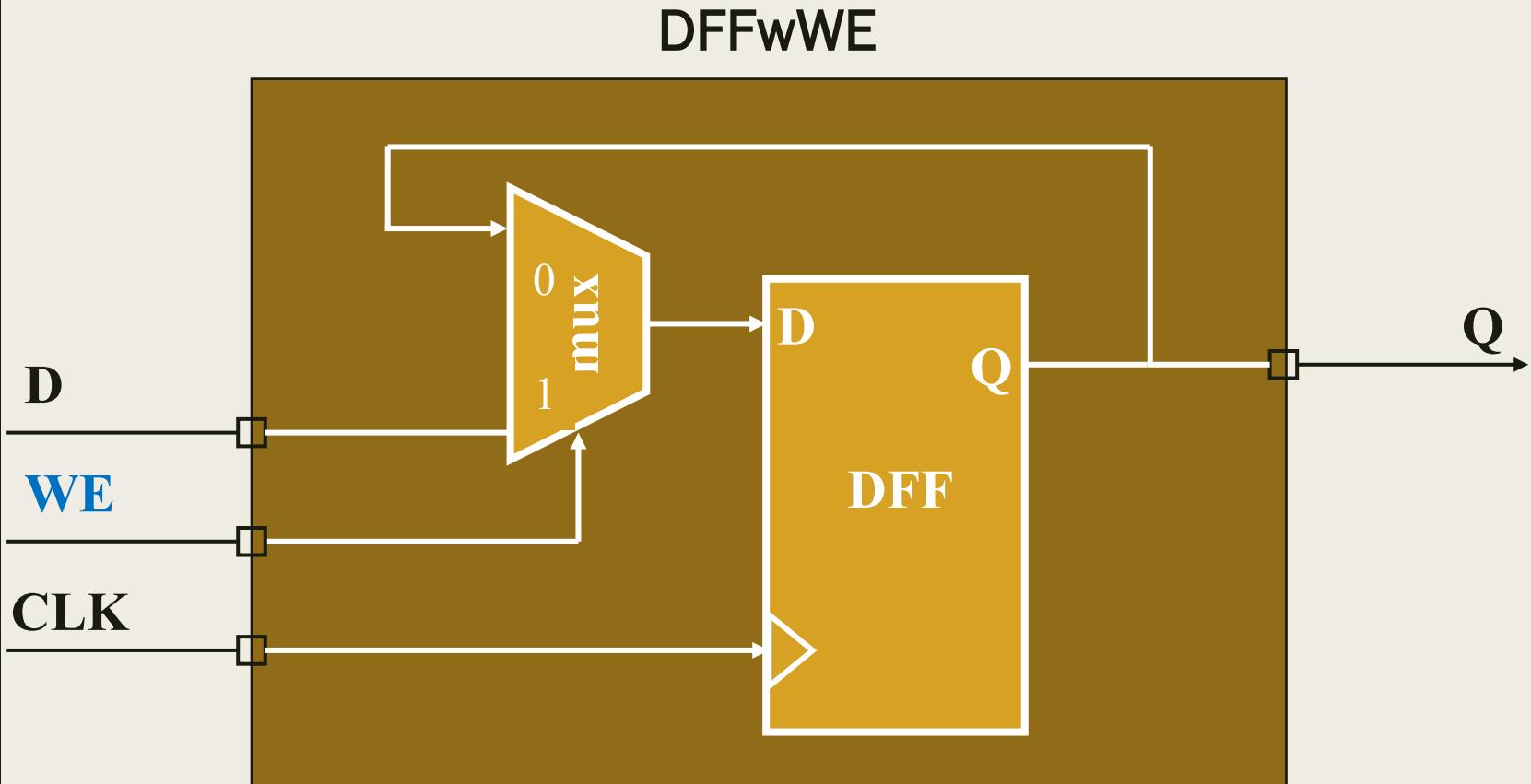


- Σχηματικό διάγραμμα σε τεχνολογία FPGA



D Flip-Flop with Write Enable στη VHDL

Περιγραφή συμπεριφοράς



Το σήμα Write Enable (WE = 1) είναι σύγχρονο και εγκρίνει την εγγραφή του D F/F στην επόμενη ακμή του CLK

D Flip-Flop with Write Enable στη VHDL

Περιγραφή συμπεριφοράς

```
entity DFFwWE is
  port (
    CLK, D, WE: in STD_LOGIC;
    Q: out STD_LOGIC);
end DFFwWE;
architecture DFFwWE_BEH of DFFwWE is
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      if (WE = '1') then
        Q <= D;
      end if;
    end if;
  end process;
end DFFwWE_BEH;
```

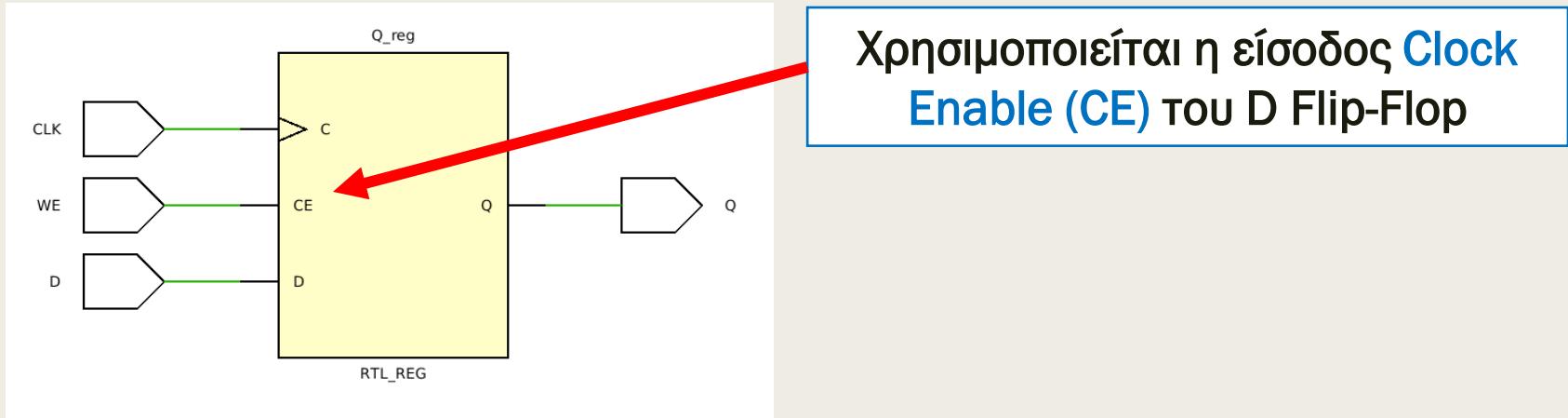
To WE δεν τοποθετείται στη λίστα ευαισθησίας

Η συνθήκη του σύγχρονου WE εξετάζεται μετά τη συνθήκη του CLK

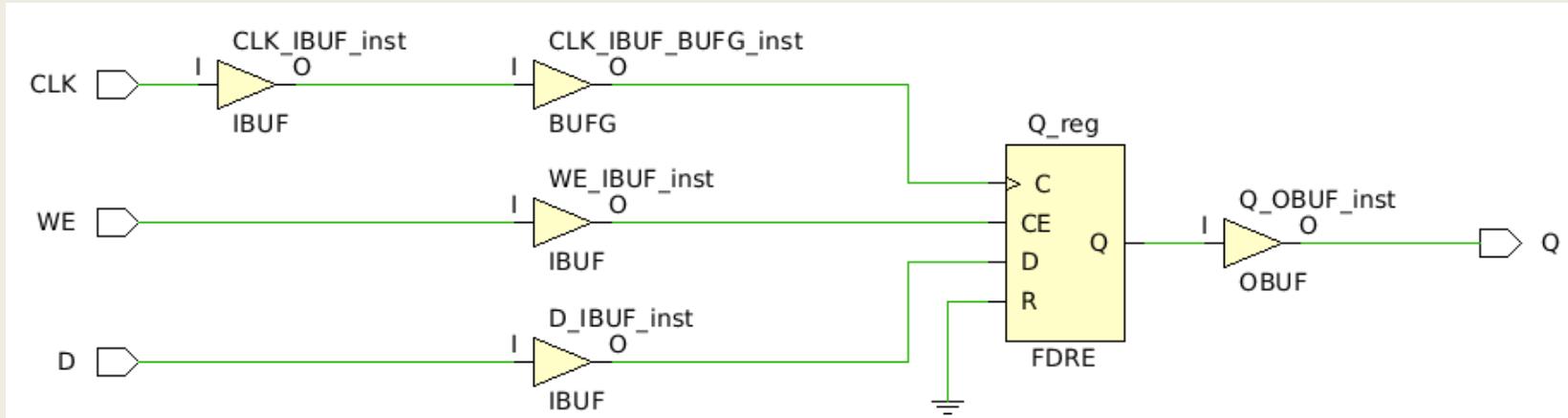
D Flip-Flop with Write Enable στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL

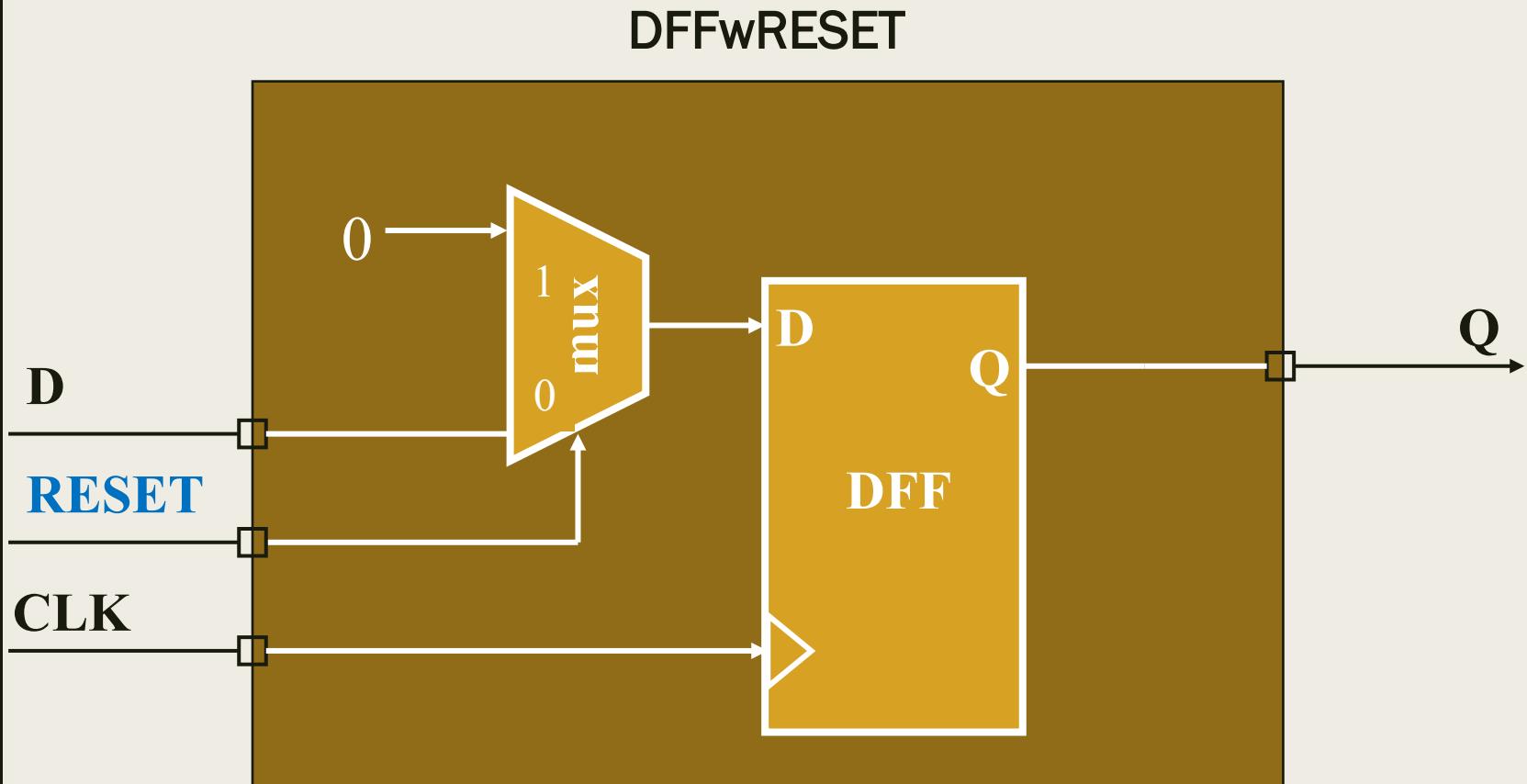


- Σχηματικό διάγραμμα σε τεχνολογία FPGA



D Flip-Flop with Reset στη VHDL

Περιγραφή συμπεριφοράς



Το σήμα **Reset Active High (RESET = 1)** είναι **σύγχρονο** και επαναφέρει το D F/F στην κατάσταση στο 0 στην επόμενη ακμή του CLK

D Flip-Flop with Reset στη VHDL

Περιγραφή συμπεριφοράς

```
entity DFFwRESET is
  port (
    CLK, D, RESET: in STD_LOGIC;
    Q: out STD_LOGIC);
end DFFwRESET;
architecture DFFwRESET_BEH of DFFwRESET is
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      if (RESET = '1') then
        Q <= '0';
      else
        Q <= D;
      end if;
    end if;
  end process;
end DFFwRESET_BEH;
```

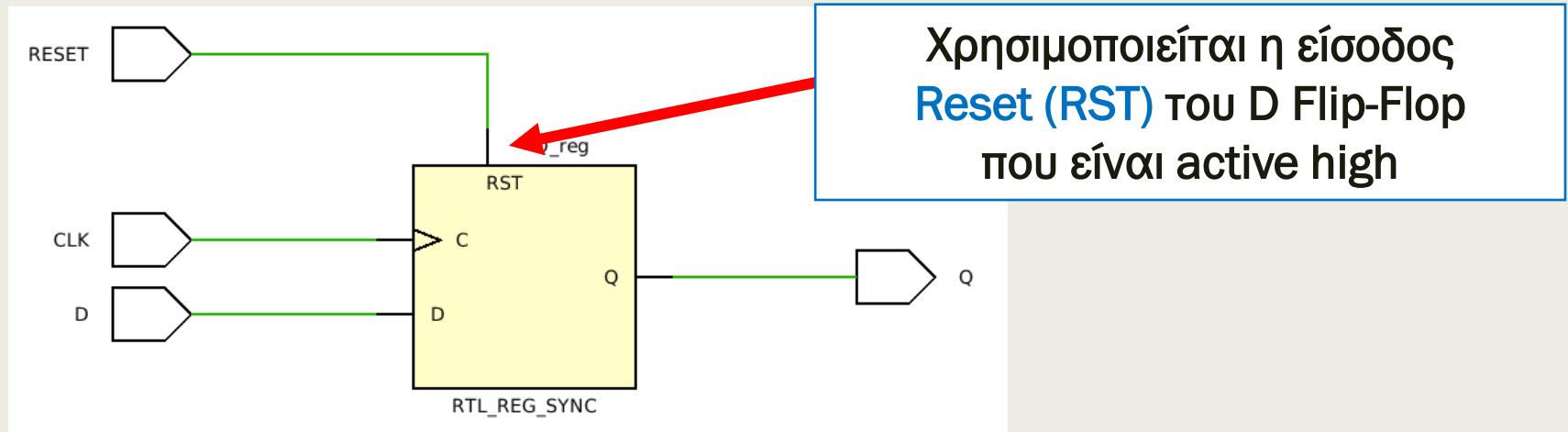
To RESET δεν τοποθετείται στη λίστα ευαισθησίας

Η συνθήκη του σύγχρονου RESET εξετάζεται μετά τη συνθήκη του CLK

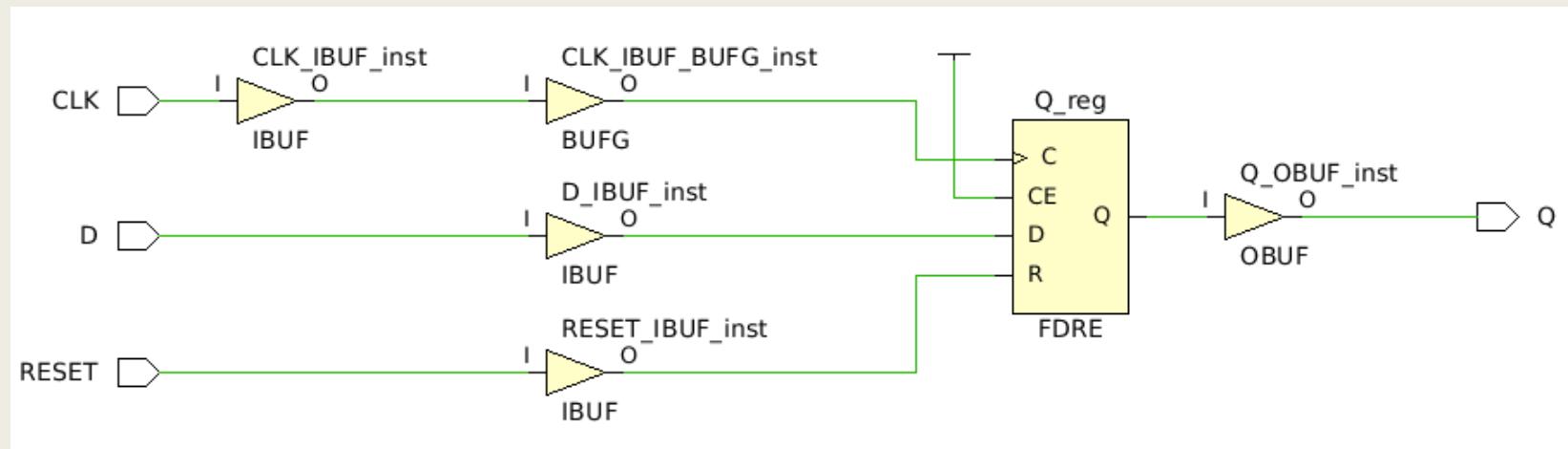
D Flip-Flop with Reset στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



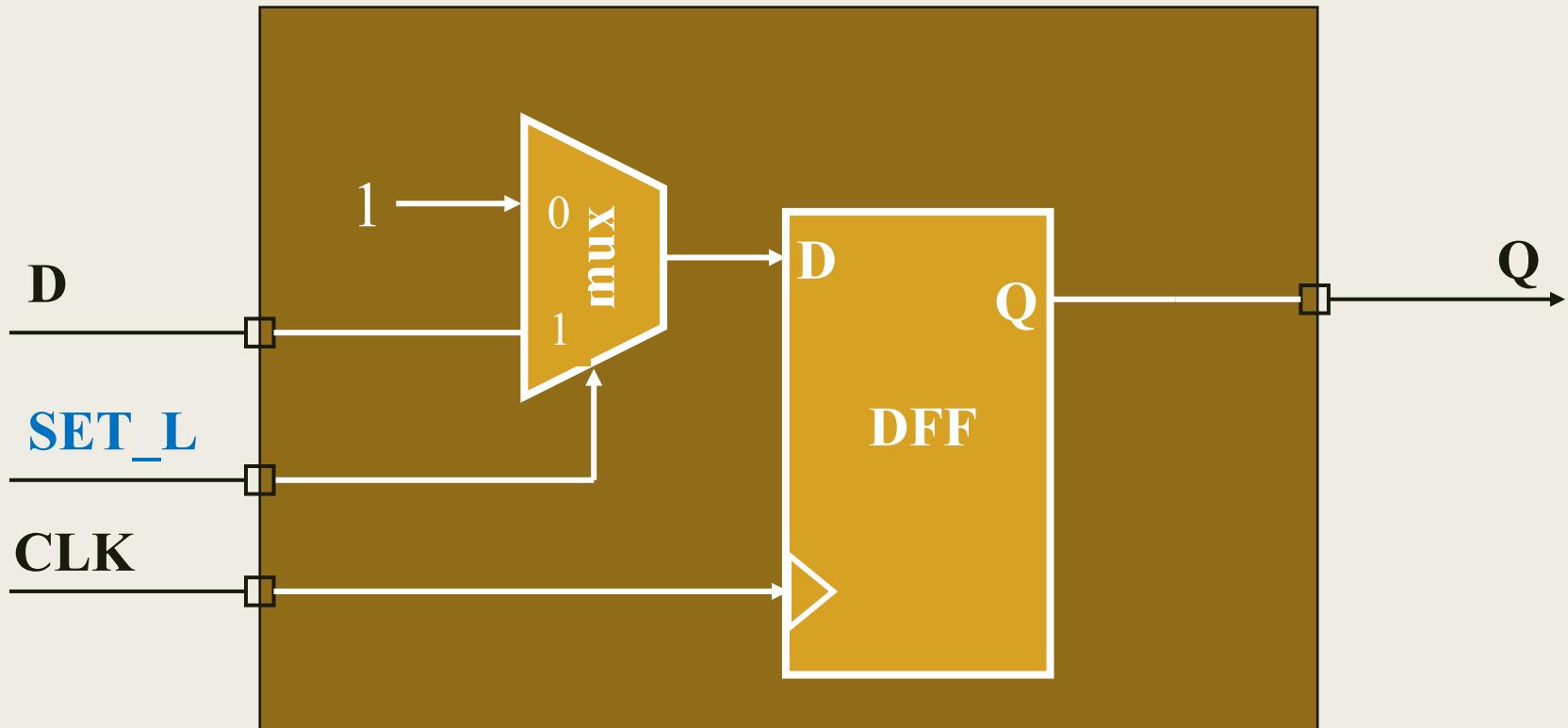
- Σχηματικό διάγραμμα σε τεχνολογία FPGA



D Flip-Flop with Set στη VHDL

Περιγραφή συμπεριφοράς

DFFwSET



Το σήμα **Set Active Low (SET_L = 0)** είναι **σύγχρονο** και τοποθετεί το D F/F στην κατάσταση 1 στην επόμενη ακμή του CLK

D Flip-Flop with Set στη VHDL

Περιγραφή συμπεριφοράς

```
entity DFFwSET is
  port (
    CLK, D, SET_L: in STD_LOGIC;
    Q: out STD_LOGIC);
end DFFwSET;
architecture DFFwSET_BEH of DFFwSET is
begin
  process (CLK) ←
begin
  if (CLK = '1' and CLK'event) then ←
    if (SET_L = '0') then ←
      Q <= '1';
    else
      Q <= D;
    end if;
  end if;
end process;
end DFFwSET_BEH;
```

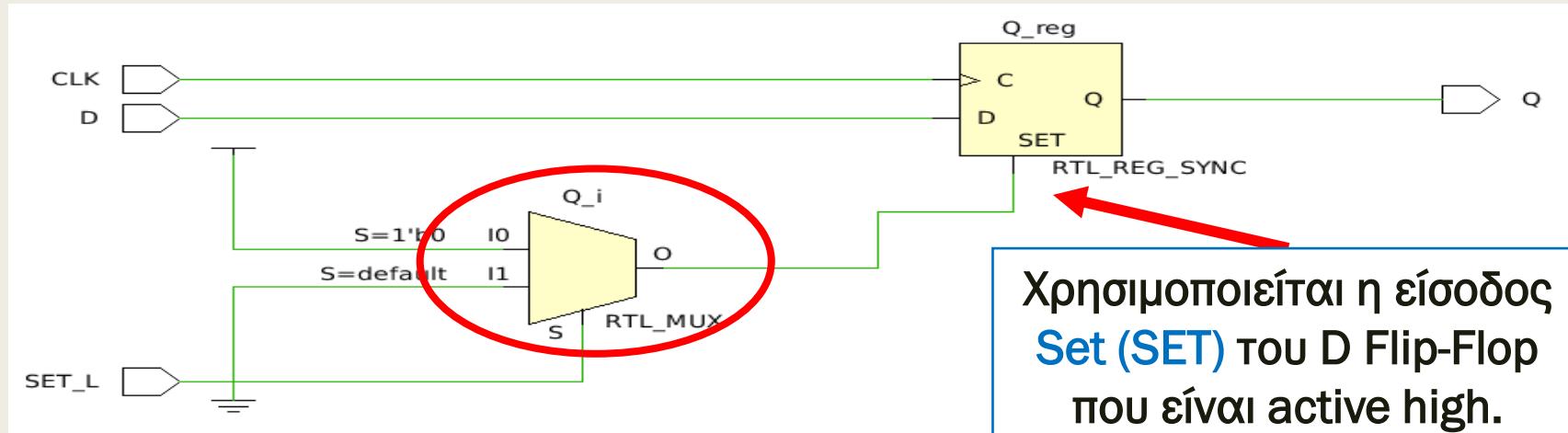
To SET_L δεν τοποθετείται στη λίστα ευαισθησίας

Η συνθήκη του σύγχρονου SET εξετάζεται μετά τη συνθήκη του CLK

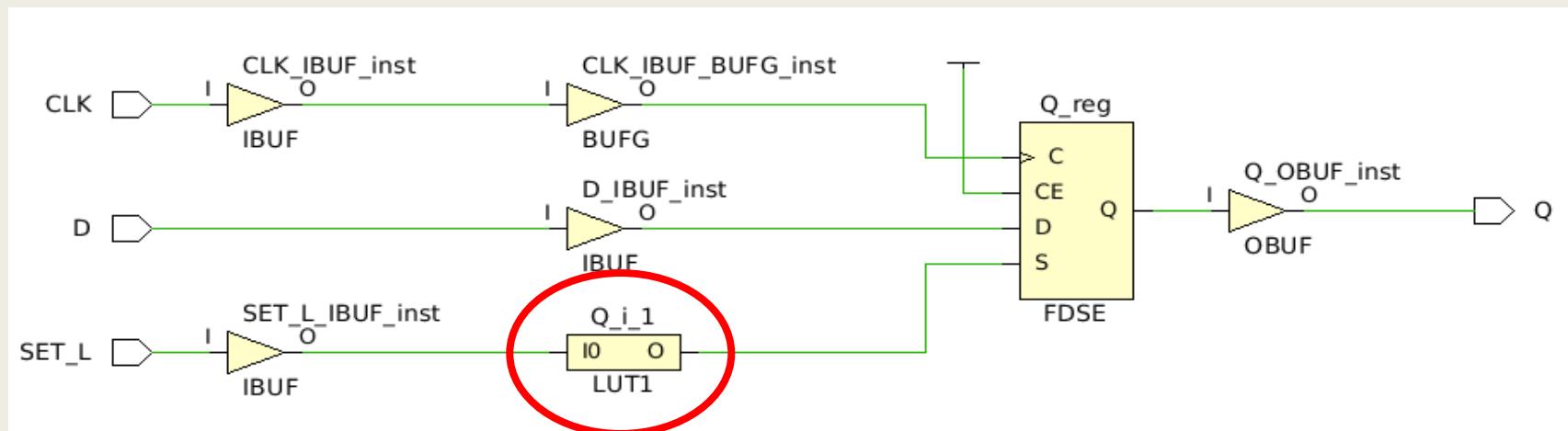
D Flip-Flop with Set στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL

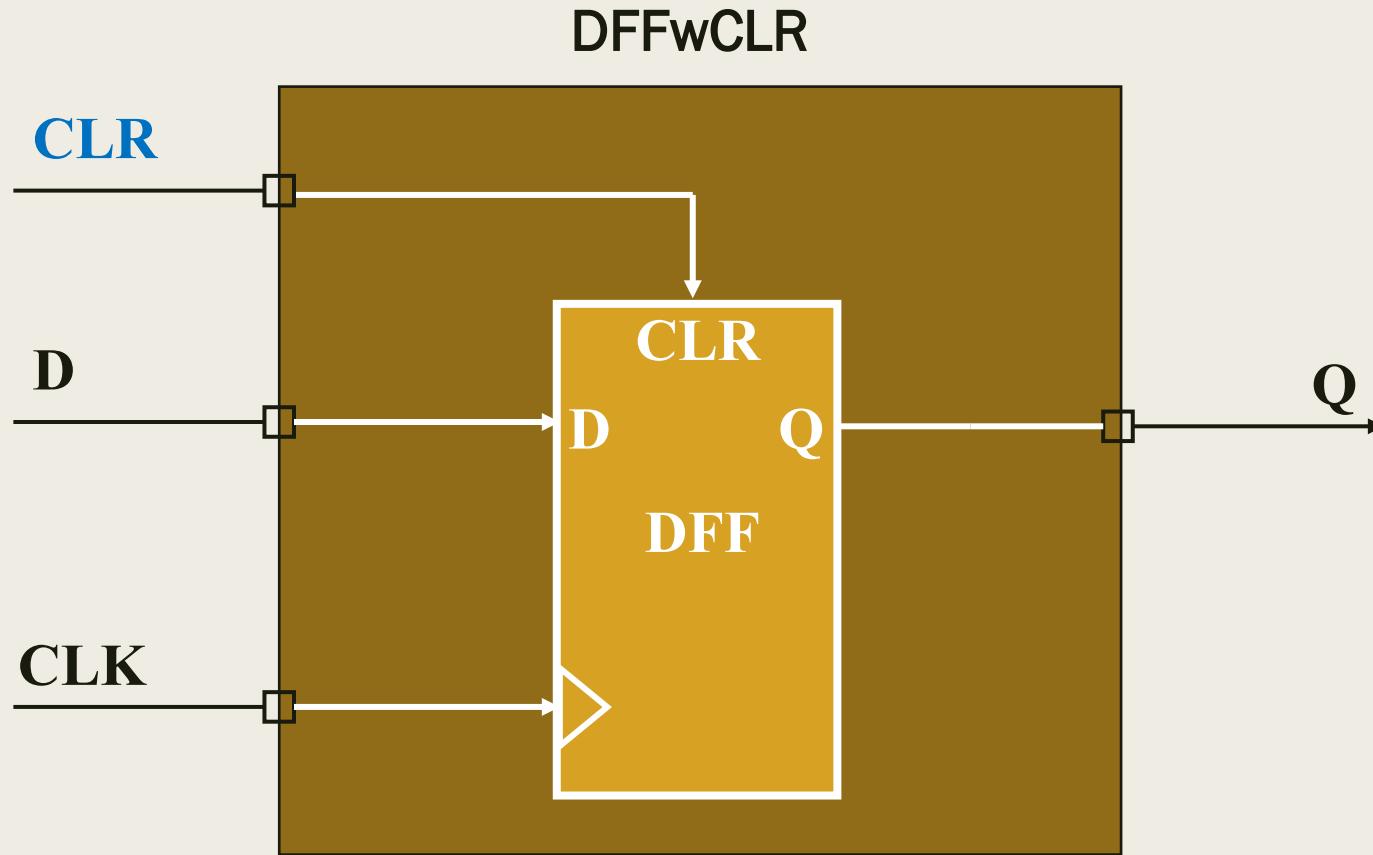


- Σχηματικό διάγραμμα σε τεχνολογία FPGA



D Flip-Flop with Clear στη VHDL

Περιγραφή συμπεριφοράς



Το σήμα **Clear Active High (CLR = 1)** είναι **ασύγχρονο** και επαναφέρει το D F/F στην κατάσταση 0 άμεσα, ανεξάρτητα από το CLK

D Flip-Flop with Clear στη VHDL

Περιγραφή συμπεριφοράς

```
entity DFFwCLR is
  port (
    D, CLK, CLR: in STD_LOGIC;
    Q: out STD_LOGIC);
end DFFwCLR;
architecture DFFwCLR_BEH of DFFwCLR is
begin
  process (CLK, CLR)
  begin
    if (CLR = '1') then
      Q <= '0';
    elsif (CLK = '1' and CLK'event) then
      Q <= D;
    end if;
  end process;
end DFFwCLR_BEH;
```

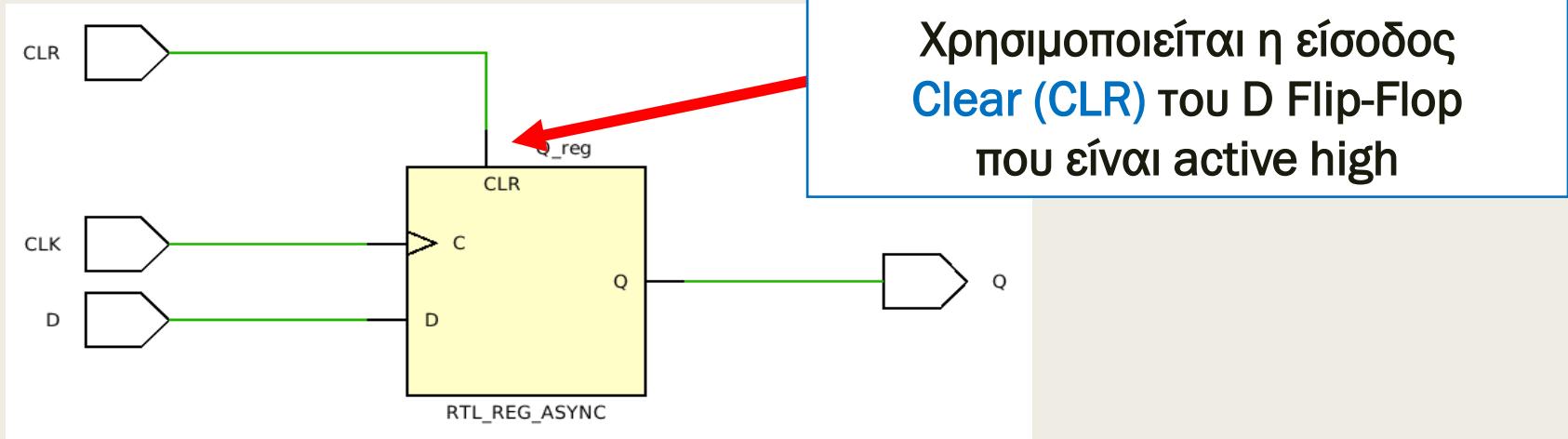
To CLR
ΤΟΠΟΘΕΤΕΙΤΑΙ
στη λίστα
ευαισθησίας

Η συνθήκη του
ασύγχρονου CLR
εξετάζεται πριν
τη συνθήκη του
CLK

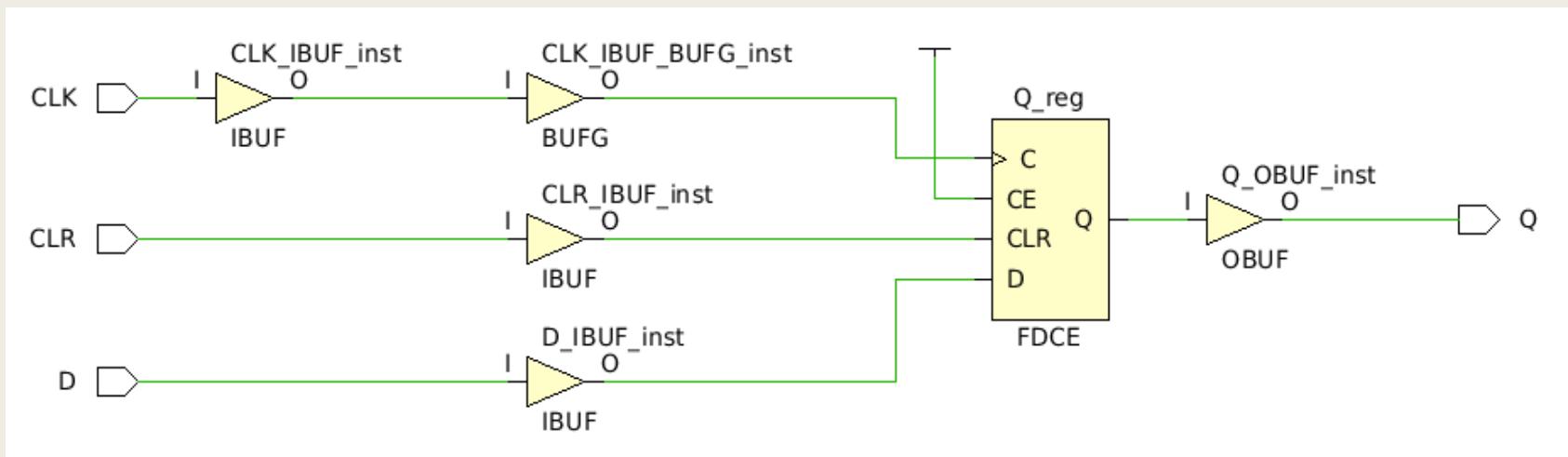
D Flip-Flop with Clear στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA



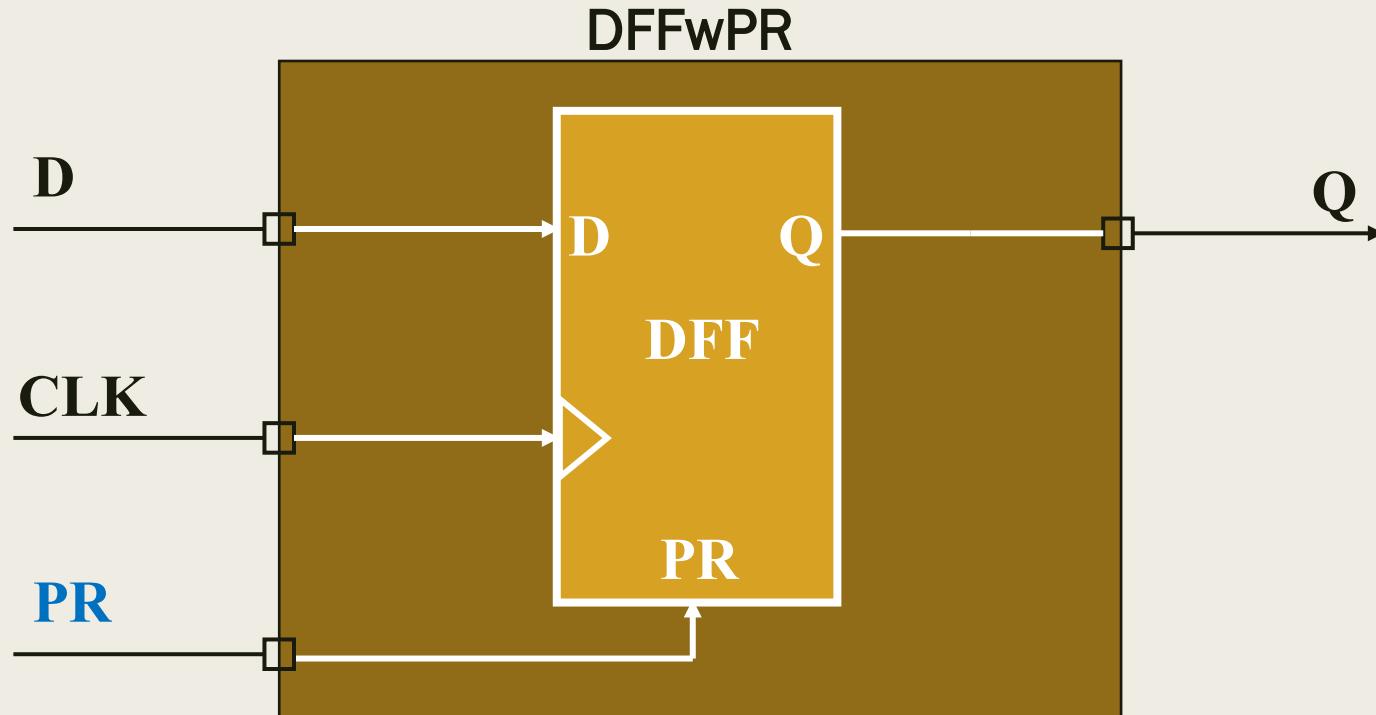
Σύγχρονο RESET vs ασύγχρονου CLEAR στη VHDL

```
architecture DFFwRESET_BEH of DFFwRESET is
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      if (RESET = '1') then
        Q <= '0';
      else
        Q <= D;
      end if;
    end if;
  end process;
end DFFwRESET_BEH;
```

```
architecture DFFwCLR_BEH of DFFwCLR is
begin
  process (CLK, CLR)
  begin
    if (CLR = '1') then
      Q <= '0';
    elsif (CLK = '1' and CLK'event) then
      Q <= D;
    end if;
  end process;
end DFFwCLR_BEH;
```

D Flip-Flop with Preset στη VHDL

Περιγραφή συμπεριφοράς



Το σήμα **Preset Active High (PR = 1)** είναι **ασύγχρονο** και θέτει το D F/F στην κατάσταση 1 άμεσα, ανεξάρτητα από το CLK

D Flip-Flop with Preset στη VHDL

Περιγραφή συμπεριφοράς

```
entity DFFwCLR is
  port (
    D, CLK, CLR: in STD_LOGIC;
    Q: out STD_LOGIC);
end DFFwCLR;
architecture DFFwPR_BEH of DFFwPR is
begin
  process (CLK, PR)
  begin
    if (PR = '1') then
      Q <= '1';
    elsif (CLK = '1' and CLK'event) then
      Q <= D;
    end if;
  end process;
end DFFwPR_BEH;
```

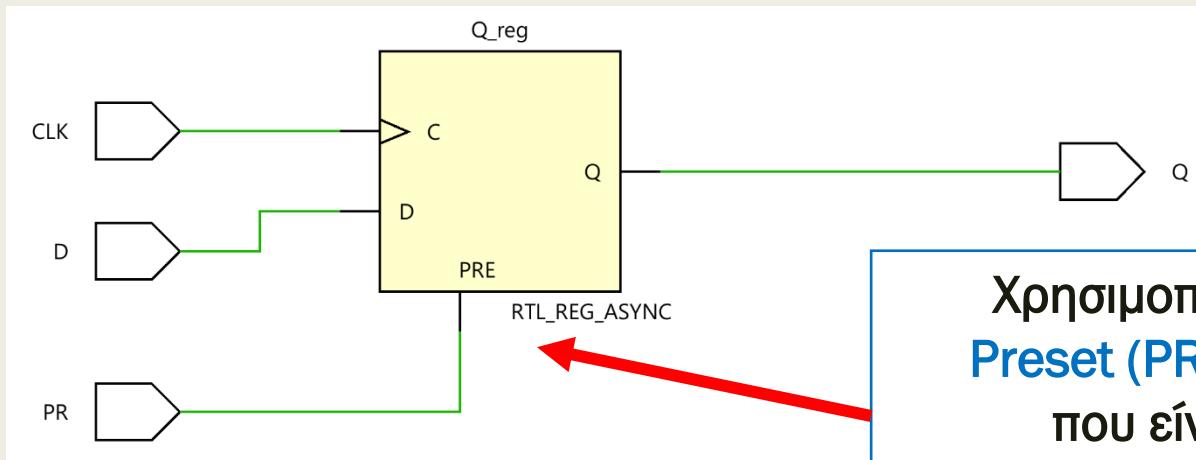
To PR
ΤΟΠΟΘΕΤΕΙΤΑΙ
στη λίστα
ευαισθησίας

Η συνθήκη του
ασύγχρονου PR
εξετάζεται πριν
τη συνθήκη του
CLK

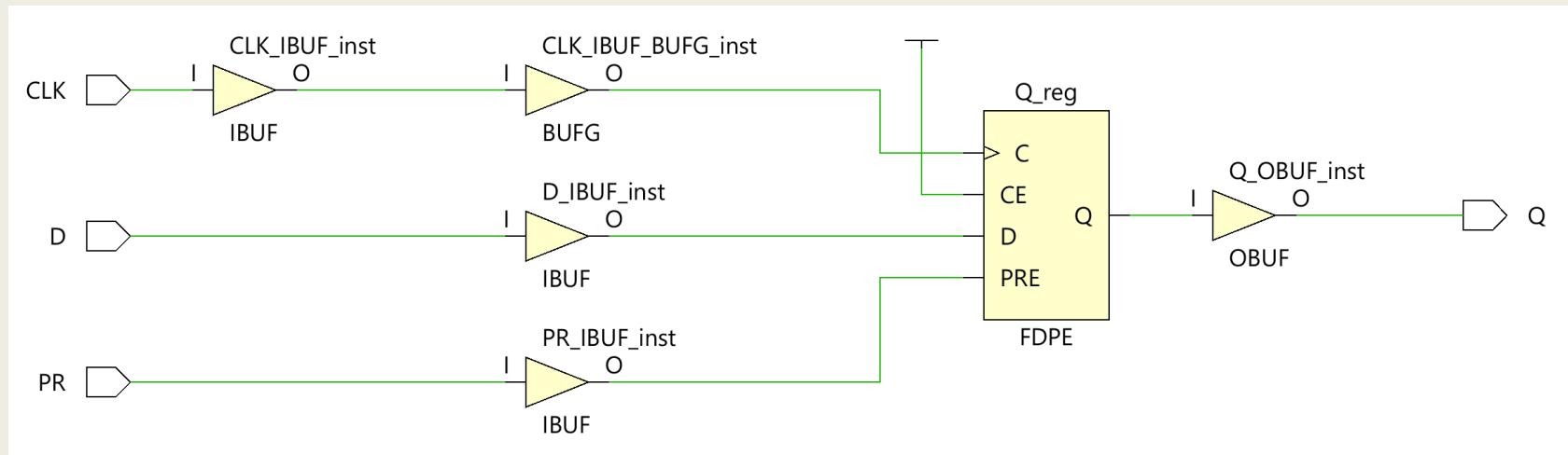
D Flip-Flop with Preset στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



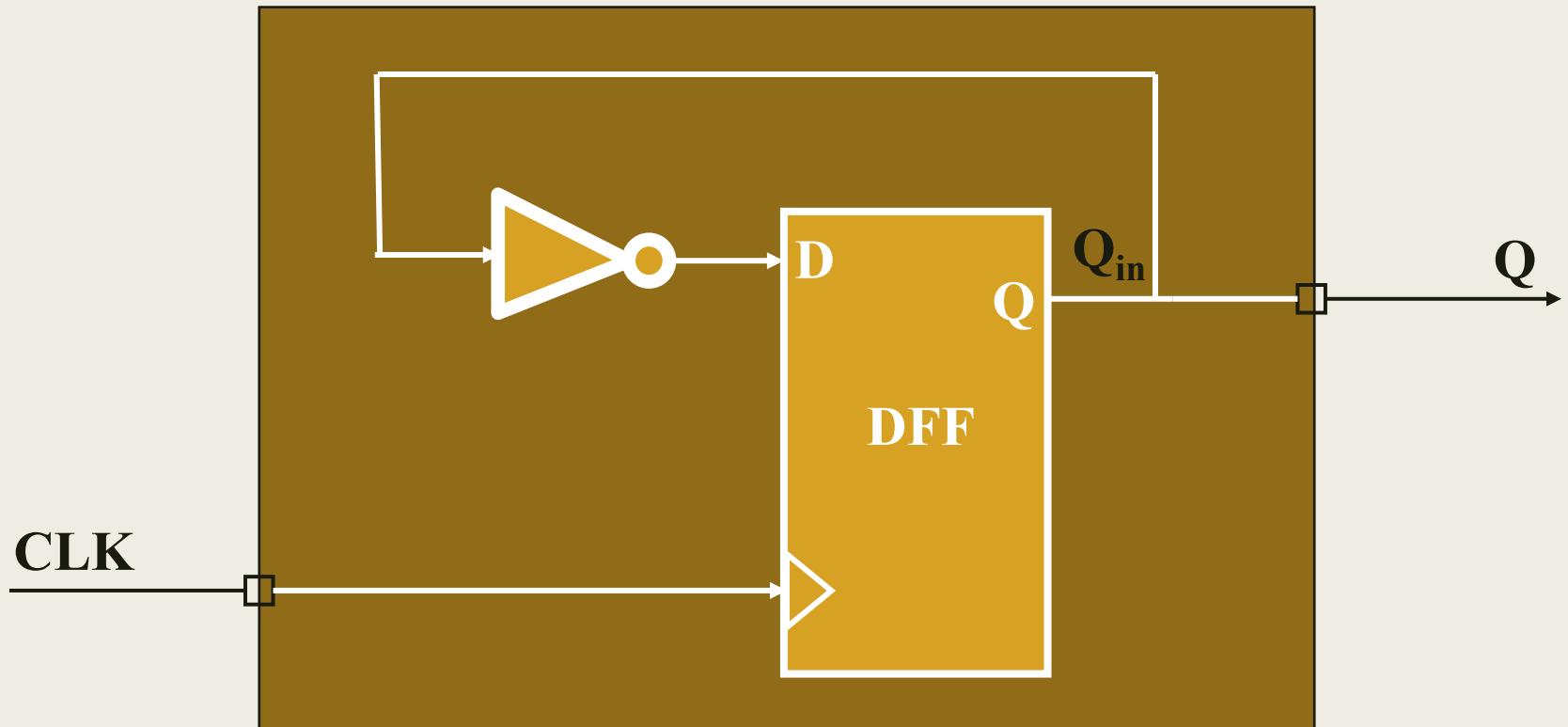
- Σχηματικό διάγραμμα σε τεχνολογία FPGA



T (Toggle) Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

TFF



Σε κάθε ανερχόμενη ακμή του CLK αλλάζει κατάσταση ($0 \rightarrow 1 \rightarrow 0 \dots$)

T Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

```
entity TFF is
  port (
    CLK : in STD_LOGIC;
    Q  : out STD_LOGIC);
end TFF;
architecture BEHAVIORAL of TFF is
  signal Q_in: STD_LOGIC;
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      Q_in <= not Q_in;
    end if;
  end process;
  Q <= Q_in;
end BEHAVIORAL;
```

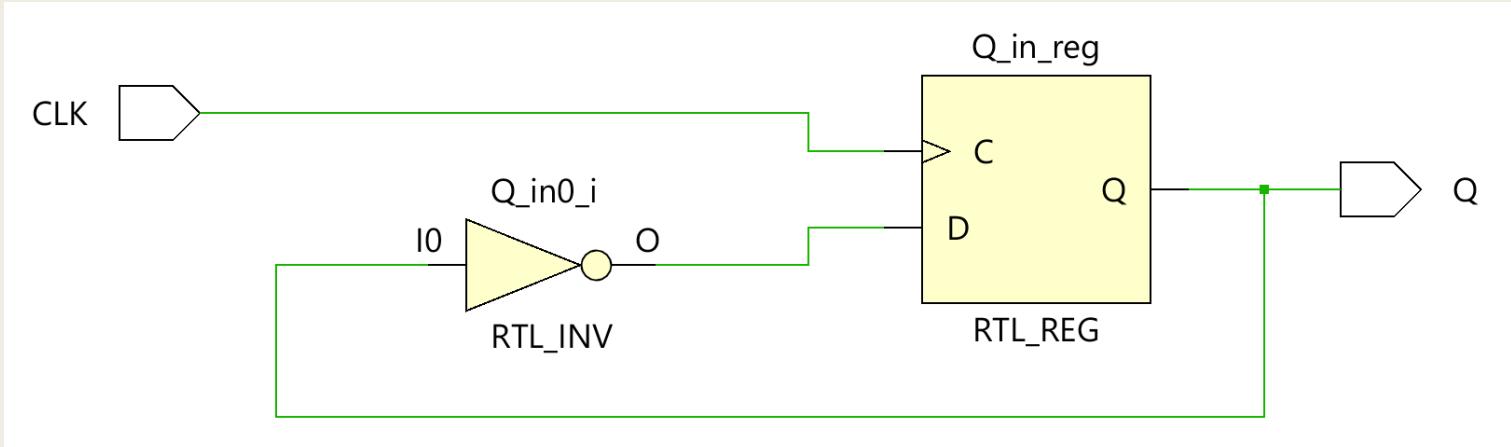
Απαίτηση για εσωτερικό σήμα Q_in

Ανάθεση τιμής στο Q
εκτός process

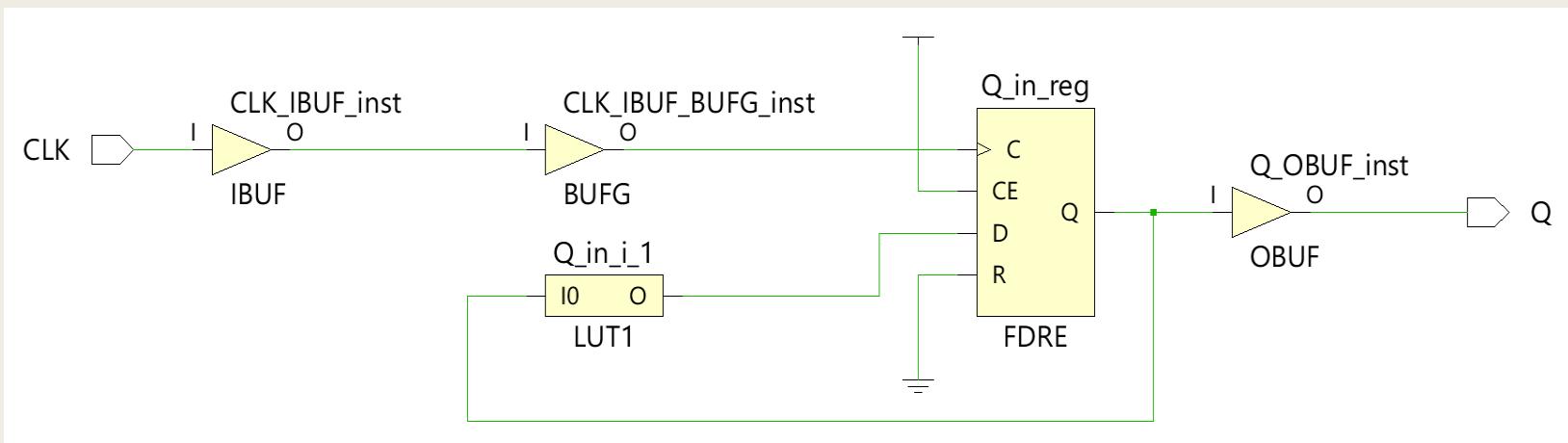
T Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



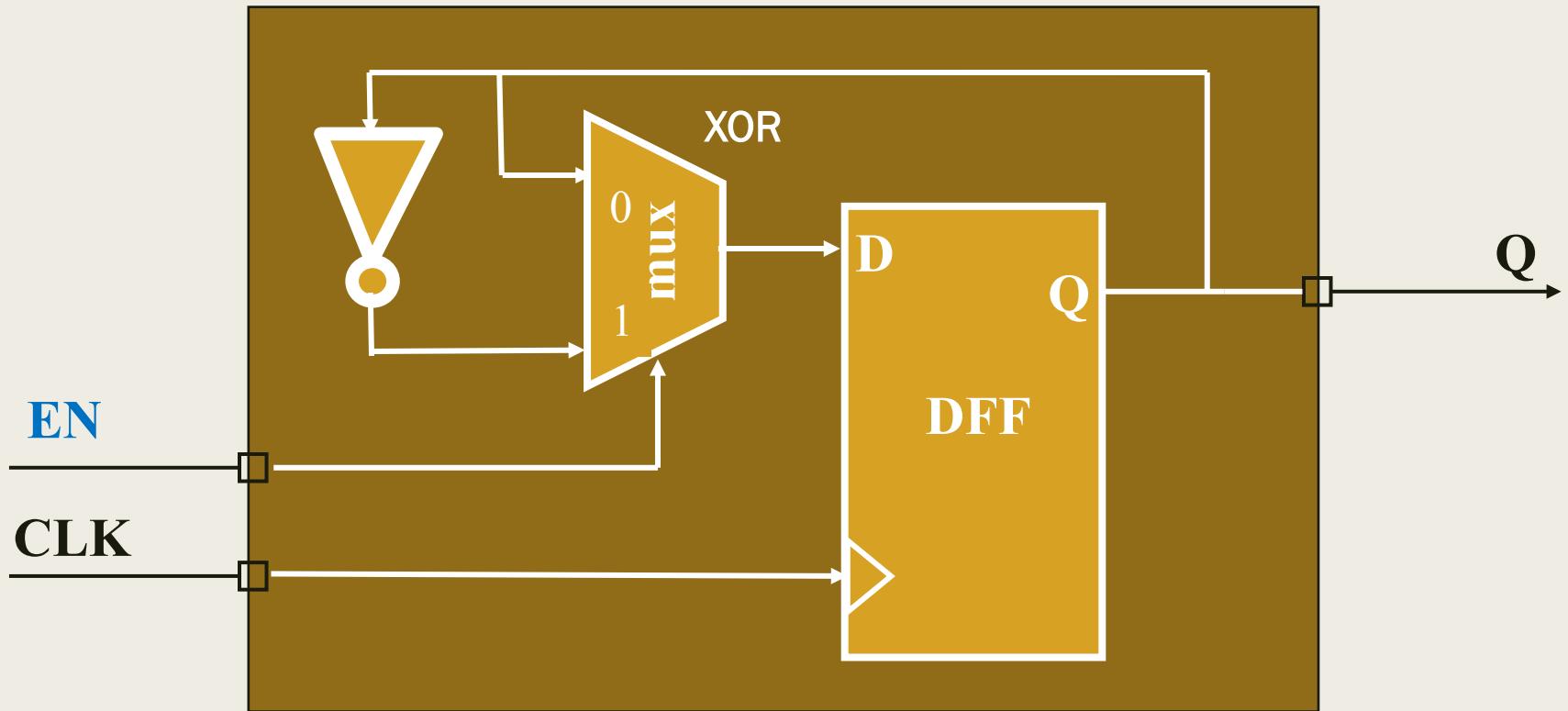
- Σχηματικό διάγραμμα σε τεχνολογία FPGA



T Flip-Flop with Enable στη VHDL

Περιγραφή συμπεριφοράς

TFFwEN



Το σήμα **Enable** ($EN = 1$) είναι **σύγχρονο** και εγκρίνει την αλλαγή κατάστασης του T F/F στην επόμενη ακμή του CLK

T Flip-Flop with Enable στη VHDL

Περιγραφή συμπεριφοράς

```
entity TFFwEN is
  port (
    CLK, EN : in STD_LOGIC;
    Q : out STD_LOGIC);
end TFFwEN;
architecture BEHAVIORAL of TFFwEN is
  signal Q_in: STD_LOGIC;
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      if (EN = '1') then
        Q_in <= not Q_in;
      end if;
    end if;
  end process;
  Q <= Q_in;
end BEHAVIORAL;
```

Απαίτηση για εσωτερικό σήμα Q_in

To EN δεν τοποθετείται στη λίστα ευαισθησίας

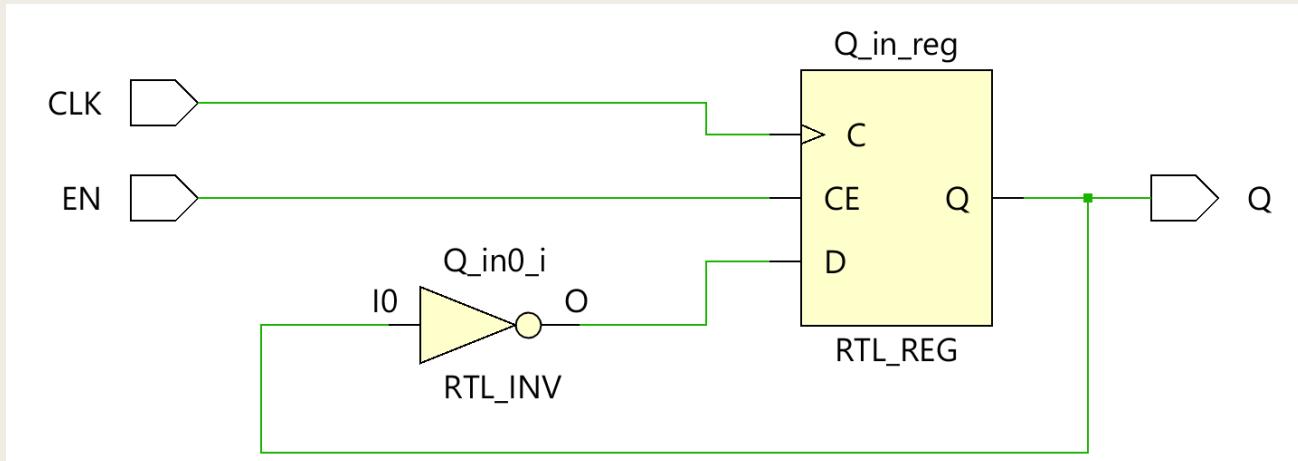
Η συνθήκη του σύγχρονου EN εξετάζεται μετά τη συνθήκη του CLK

Ανάθεση τιμής στο Q εκτός process

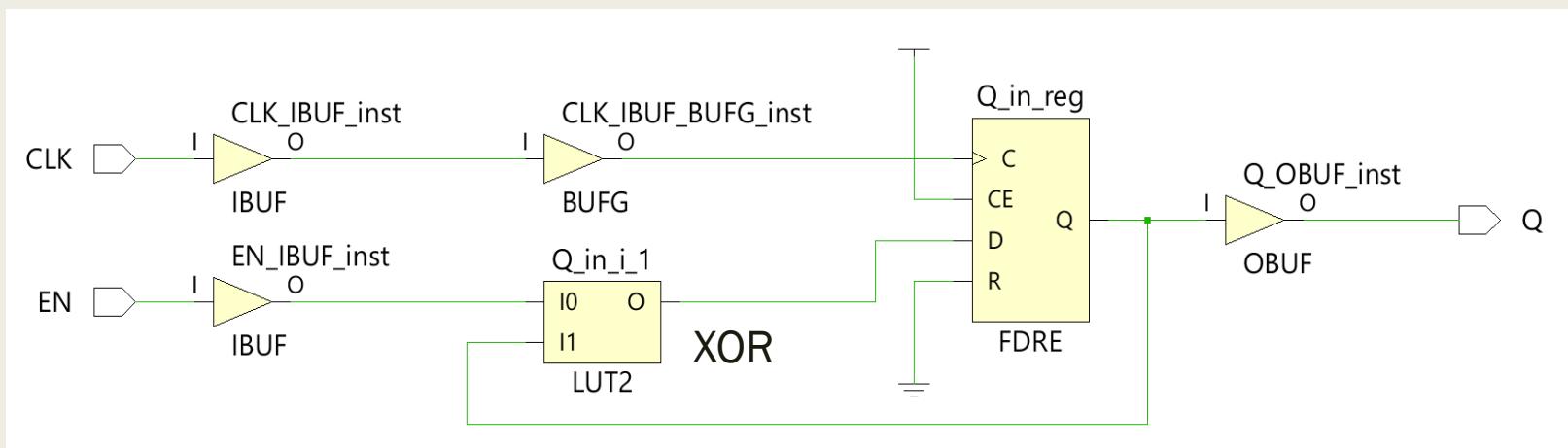
T Flip-Flop with Enable στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL



- Σχηματικό διάγραμμα σε τεχνολογία FPGA



JK Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

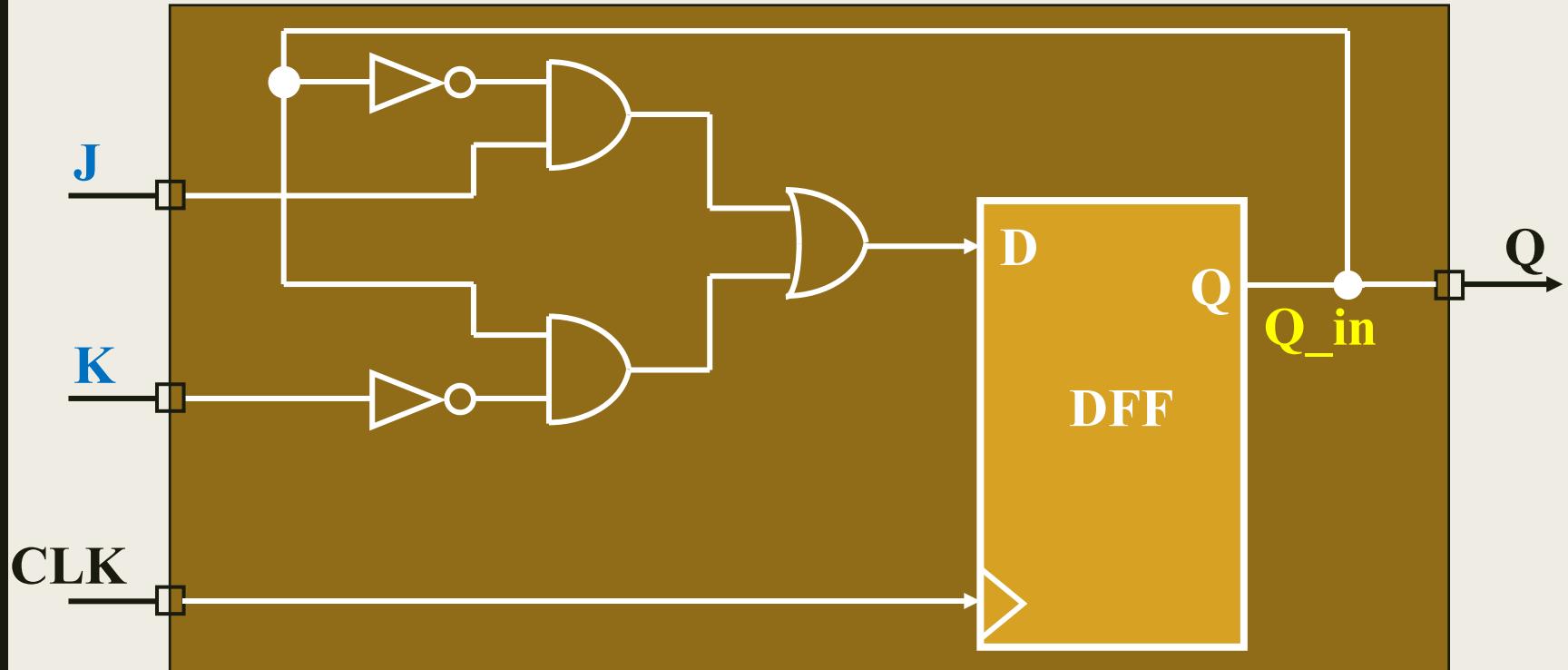
- Το **J-K Flip-Flop** δέχεται ένα σήμα CLK και δύο **σύγχρονες** εισόδους (το *J* και το *K*).
- Κατά την ανερχόμενη ακμή του CLK ενημερώνει την έξοδο *Q*, σύμφωνα με τις τιμές που έχουν οι είσοδοι *J* και *K*, ως εξής:
 - Όταν οι είσοδοι *J* και *K* έχουν και οι δύο την τιμή 0, η έξοδος *Q* διατηρεί την προηγούμενη τιμή της (*hold*)
 - Όταν η είσοδος *J* έχει την τιμή 0 και η είσοδος *K* έχει την τιμή 1, η έξοδος *Q* παίρνει την τιμή 0 (*reset*)
 - Όταν η είσοδος *J* έχει την τιμή 1 και η είσοδος *K* έχει την τιμή 0, η έξοδος *Q* παίρνει την τιμή 1 (*set*)
 - Όταν οι είσοδοι *J* και *K* έχουν και οι δύο την τιμή 1, η έξοδος *Q* εναλλάσσει την τιμή της με το συμπλήρωμα της προηγούμενης τιμής της (*toggle*)
- Το **J-K Flip-Flop** υλοποιεί **την εξίσωση Boole**:

$$Q(t + 1) = \overline{Q(t)} J + Q(t) \bar{K}$$

JK Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

JKFF



Τα σήματα **J** και **K** είναι **σύγχρονα**

Υλοποιείται η εξίσωση Boole:

$$Q(t+1) = \overline{Q(t)} J + Q(t) \bar{K}$$

JK Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

```
entity JKFF is
  port (
    CLK, J, K: in STD_LOGIC;
    Q: out STD_LOGIC);
end JKFF;
architecture JKFF_BEH of JKFF is
  signal Q_in: STD_LOGIC;
begin
  process (CLK)
    begin
      if (CLK = '1' and CLK'event) then
        Q_in <= (J and (not Q_in)) or ((not K) and Q_in);
      end if;
    end process;
    Q <= Q_in;
  end JKFF_BEH;
```

Απαίτηση για εσωτερικό σήμα Q_in

Τα J, K δεν
τοποθετούνται
στη λίστα
ευαισθησίας

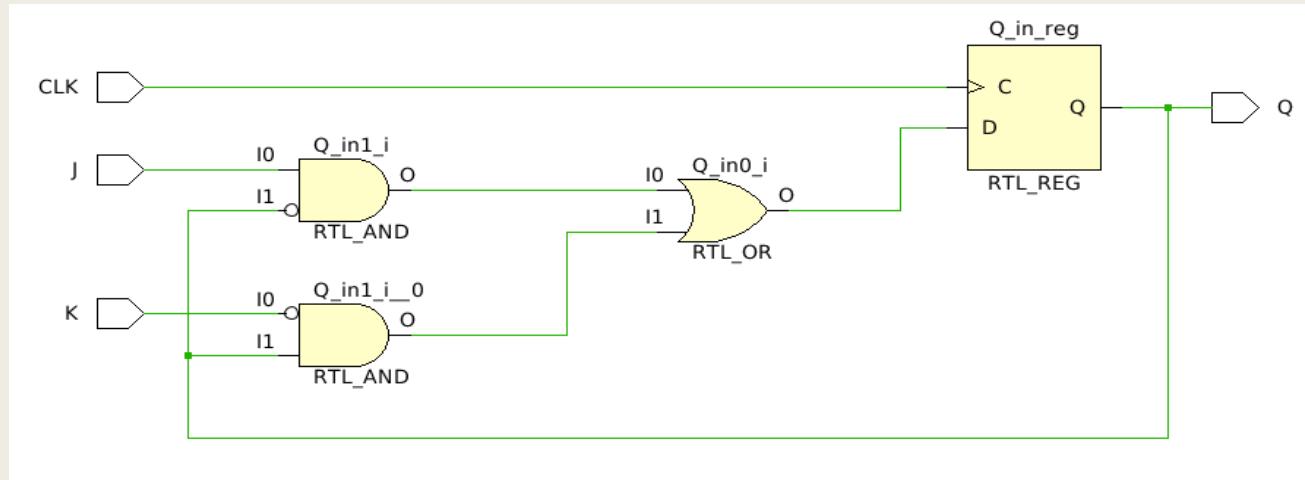
Ανάθεση τιμής στο Q
ΕΚΤΟΣ process

Η εξίσωση Boole
του Flip-Flop
περιγράφεται μετά
τη συνθήκη του CLK

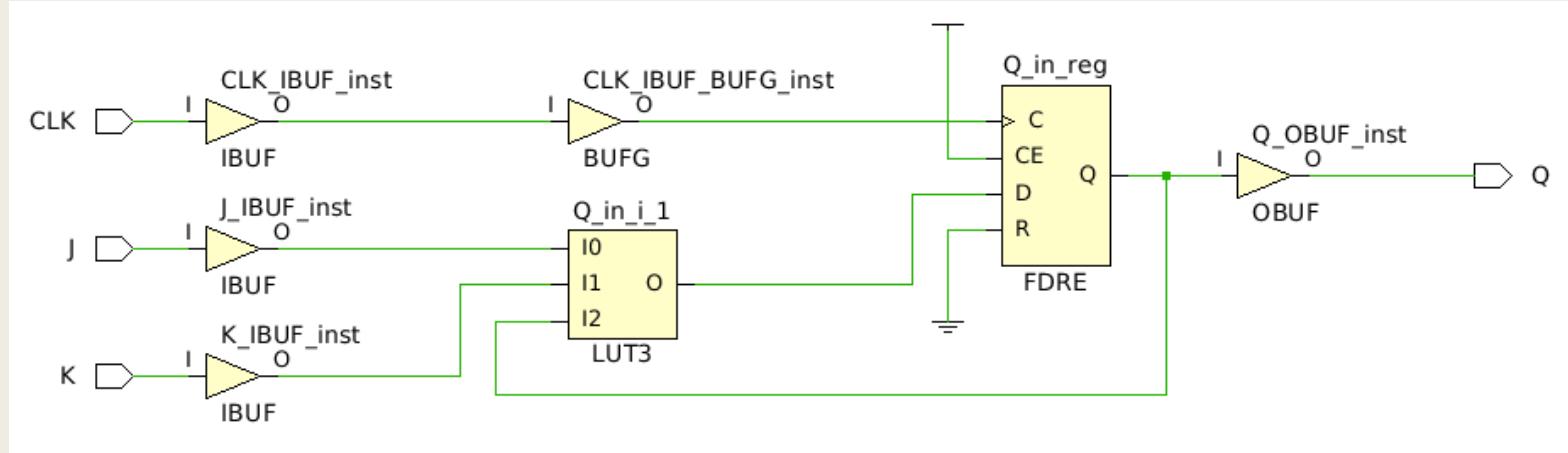
JK Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

- Σχηματικό διάγραμμα RTL – Υλοποιείται η εξίσωση Boole



- Σχηματικό διάγραμμα σε τεχνολογία FPGA – Το LUT3 υλοποιεί την εξίσωση Boole



Επιλεγμένη άσκηση: το AB Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

- Να περιγράψετε στη γλώσσα VHDL την αρχιτεκτονική του **AB Flip-Flop**

AB	Λειτουργία
00	HOLD
01	TOGGLE
10	RESET
11	LOAD

Επιλεγμένη άσκηση: το AB Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

- Να περιγράψετε στη γλώσσα VHDL την αρχιτεκτονική του **AB Flip-Flop**

AB	Λειτουργία
00	HOLD
01	TOGGLE
10	RESET
11	LOAD

AB	Λειτουργία	Q(t+1)
00	HOLD	Q(t)
01	TOGGLE	$\overline{Q(t)}$
10	RESET	0
11	LOAD	D

Επιλεγμένη άσκηση: το AB Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

- Να περιγράψετε στη γλώσσα VHDL την αρχιτεκτονική του **AB Flip-Flop**

AB	Λειτουργία
00	HOLD
01	TOGGLE
10	RESET
11	LOAD

AB	Λειτουργία	Q(t+1)
00	HOLD	Q(t)
01	TOGGLE	$\overline{Q(t)}$
10	RESET	0
11	LOAD	D

$$\begin{aligned} Q(t+1) &= \overline{A}\overline{B}Q(t) + \overline{A}B\overline{Q(t)} + ABD \\ &= \overline{A}(\overline{B}Q(t) + B\overline{Q(t)}) + ABD \\ &= \overline{A}(Q(t) \text{ xor } B) + ABD \end{aligned}$$

Επιλεγμένη άσκηση: το AB Flip-Flop στη VHDL

Περιγραφή συμπεριφοράς

- Να περιγράψετε στη γλώσσα VHDL την αρχιτεκτονική του AB Flip-Flop με εξίσωση Boole:

$$Q(t+1) = \bar{A}(B \oplus Q(t)) + A(BD)$$

AB	Λειτουργία
00	HOLD
01	TOGGLE
10	RESET
11	LOAD

```

architecture BEHAVIORAL of ABFF is
    signal Q_in: STD_LOGIC;
begin
    process (CLK)
        begin
            if (CLK = '1' and CLK'event) then
                Q_in <= ((not A) and (B xor Q_in)) or ((A and (B and D)));
            end if;
        end process;
        Q <= Q_in;
    end BEHAVIORAL;

```

Απαίτηση για εσωτερικό σήμα Q_in

Τα A, B δεν τοποθετούνται στη λίστα ευαισθησίας

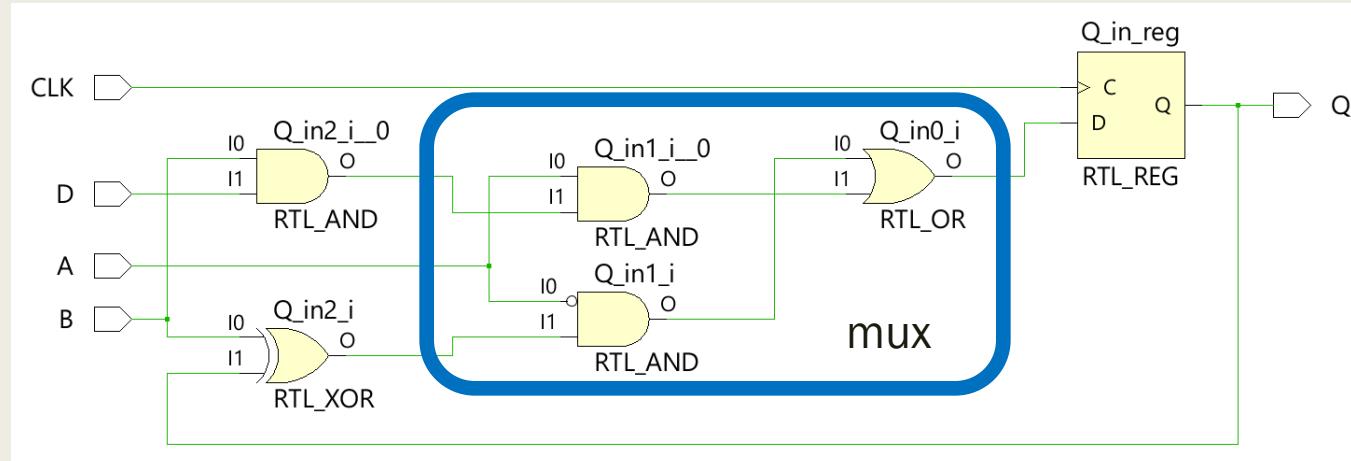
Η εξίσωση Boole του Flip-Flop περιγράφεται μετά τη συνθήκη του CLK

Ανάθεση τιμής στο Q εκτός process

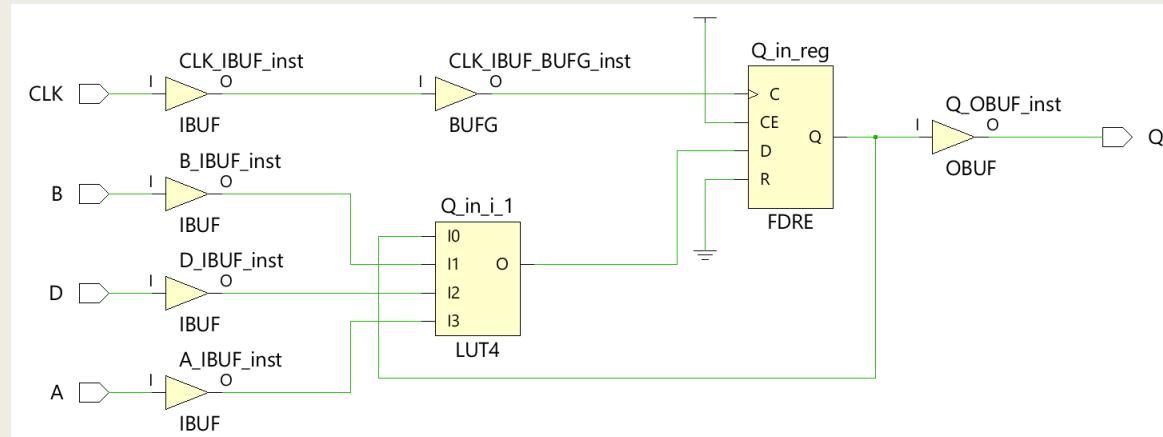
AB Flip-Flop στη VHDL

Περιγραφή Συμπεριφοράς

- Σχηματικό διάγραμμα RTL – Υλοποιείται η εξίσωση Boole



- Σχηματικό διάγραμμα σε τεχνολογία FPGA – Το LUT4 υλοποιεί την εξίσωση Boole



Ασκήσεις Επανάληψης -1

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y:   out STD_LOGIC);
end LD_VHDL;

architecture Behavioral of LD_VHDL is
  signal Z: std_logic;

begin
  B<=A;
  Z:=A+Y;

end Behavioral;
```

Εντοπίστε τα λάθη
(3)

Ασκήσεις Επανάληψης -1

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y:   out STD_LOGIC);
end LD_VHDL;
```

```
architecture Behavioral of LD_VHDL is
```

```
  signal Z: std_logic;
```

```
  begin
```

```
    B<=A;
```

```
    Z:=A+Y;
```

```
  end Behavioral;
```

Εντοπίστε τα λάθη
(3)

Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;
```

architecture Behavioral of LD_VHDL is

```
signal C: std_logic;
```

```
begin
```

```
process (A) is
```

```
begin
```

```
Y<=A AND B;
```

```
C<=B;
```

```
Z<=A AND C;
```

```
end process;
```

```
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη των A, B ανά 20ns

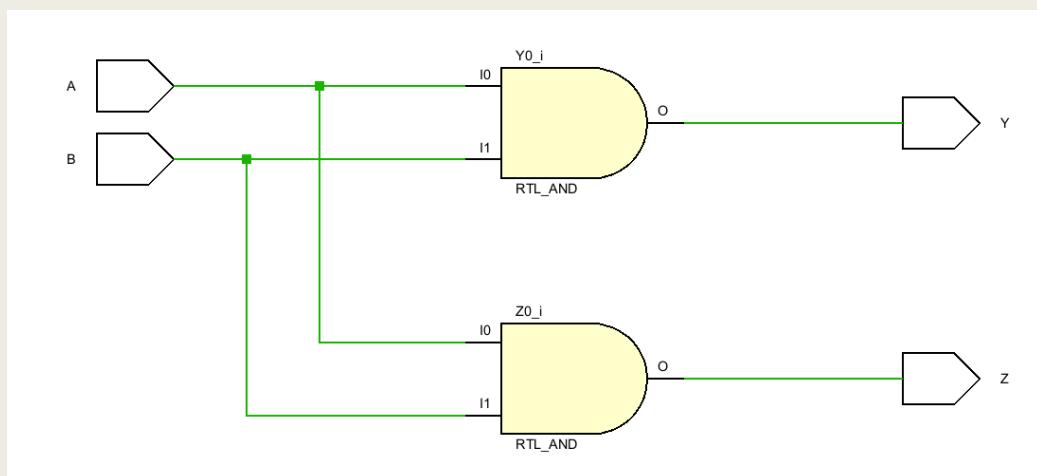
A='0', B='0'

A='1', B='1'

A='1', B='0'

Βρείτε πως μεταβάλλονται οι έξοδοι Y, Z και το σήμα C

Θεωρείστε ότι αρχικά ΟΛΑ τα σήματα είναι στην τιμή '0'



Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z: out STD_LOGIC);
end LD_VHDL;
```

architecture Behavioral of LD_VHDL is

```
  signal C: std_logic;
```

```
begin
```

```
  process (A) is
  begin
```

```
    Y<=A AND B;
```

```
    C<=B;
```

```
    Z<=A AND C;
```

```
  end process;
```

```
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη
των A, B ανά 20ns

A='0', B='0'

A='1', B='1'

A='1', B='0'

Βρείτε πως μεταβάλλονται οι έξοδοι
Y, Z και το σήμα C

A='0', B='0' => C='0', Y='0', Z='0'

Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
```

```
Port (
```

```
    A, B: in STD_LOGIC;
```

```
    Y, Z: out STD_LOGIC);
```

```
end LD_VHDL;
```

```
architecture Behavioral of LD_VHDL is
```

```
signal C: std_logic;
```

```
begin
```

```
process (A) is
```

```
begin
```

```
    Y<=A AND B;
```

```
    C<=B;
```

```
    Z<=A AND C;
```

```
end process;
```

```
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη

των A, B ανά 20ns

A='0', B='0'

A='1', B='1'

A='1', B='0'

Βρείτε πως μεταβάλλονται οι έξοδοι
Y, Z και το σήμα C

A='0', B='0' => C='0', Y='0', Z='0'

A='1', B='1' => C='1', Y='1', Z='0'

Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
```

```
Port (
```

```
    A, B: in STD_LOGIC;
```

```
    Y, Z: out STD_LOGIC);
```

```
end LD_VHDL;
```

```
architecture Behavioral of LD_VHDL is
```

```
signal C: std_logic;
```

```
begin
```

```
process (A) is
```

```
begin
```

```
    Y<=A AND B;
```

```
    C<=B;
```

```
    Z<=A AND C;
```

```
end process;
```

```
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη

των A, B ανά 20ns

A='0', B='0'

A='1', B='1'

A='1', B='0'

Βρείτε πως μεταβάλλονται οι έξοδοι
Y, Z και το σήμα C

A='0', B='0' => C='0', Y='0', Z='0'

A='1', B='1' => C='1', Y='1', Z='0'

A='1', B='0' => C='1', Y='1', Z='0'

Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;
```

```
architecture Behavioral of LD_VHDL is
```

```
  signal C: std_logic;
```

```
begin
```

```
  process (A) is
  begin
```

```
    Y<=A AND B;
    C<=B;
    Z<=A AND C;
  end process;
```

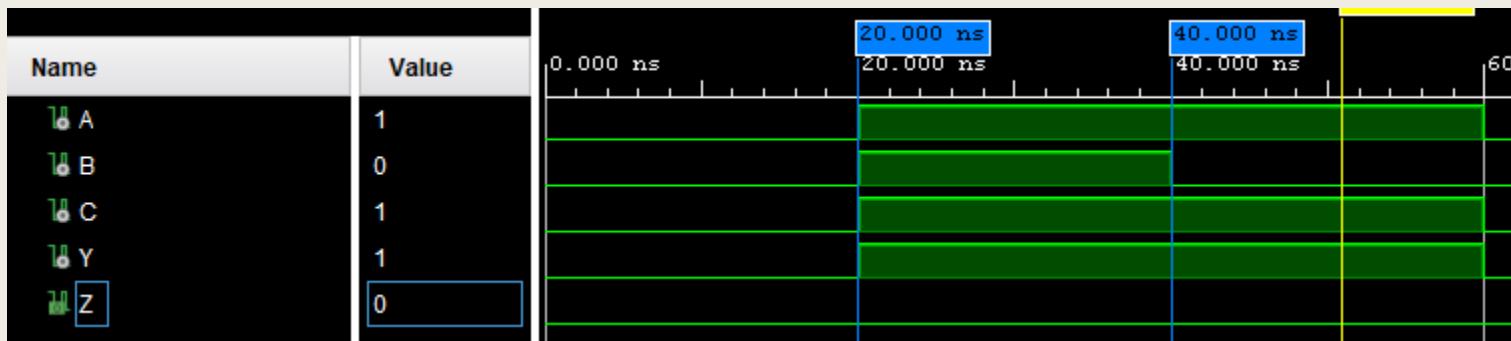
```
end Behavioral;
```

Ήθελα αυτές τις τιμές?

Μάλλον όχι

A='0', B='0' => C='0', Y='0', Z='0'
A='1', B='1' => C='1', Y='1', Z='0'
A='1', B='0' => C='1', Y='1', Z='0'

Behavioral Simulation



Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;

architecture Behavioral of LD_VHDL is

  signal C: std_logic;

begin

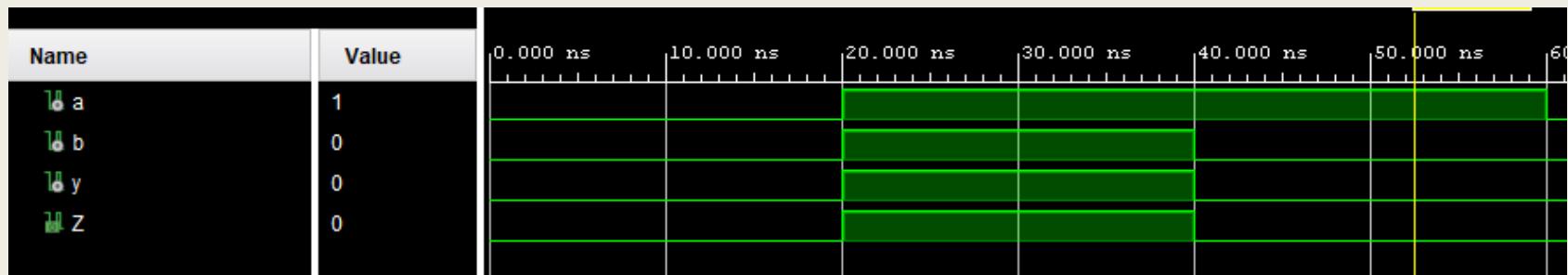
  process (A) is
  begin
    Y<=A AND B;
    C<=B;
    Z<=A AND C;
  end process;

end Behavioral;
```

Τιμές που μάλλον ήθελα

A='0', B='0' => C='0', Y='0', Z='0'
A='1', B='1' => C='1', Y='1', Z='1'
A='1', B='0' => C='0', Y='0', Z='0'

Post Synthesis Functional Simulation



Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;
```

architecture Behavioral of LD_VHDL is

```
signal C: std_logic;
```

```
begin
```

```
process (A,B,C) is
```

```
begin
```

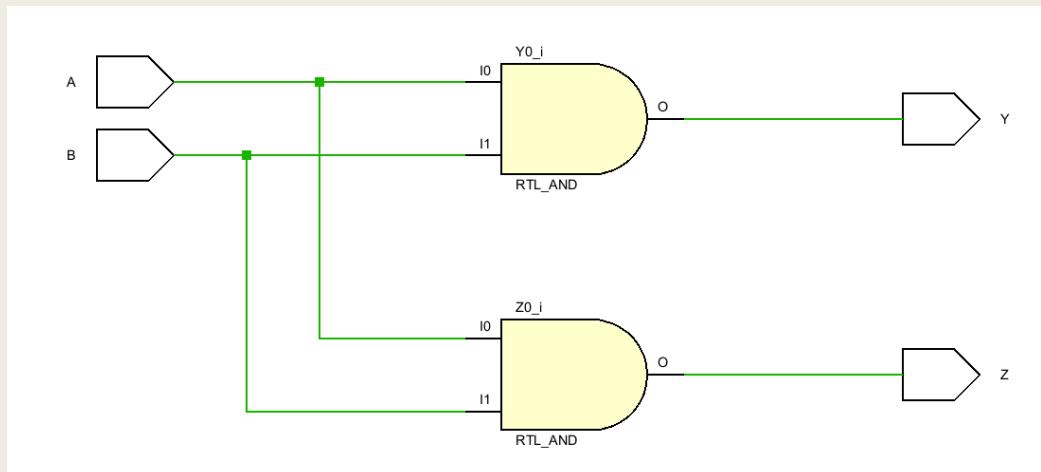
```
Y<=A AND B;
```

```
C<=B;
```

```
Z<=A AND C;
```

```
end process;
```

```
end Behavioral;
```



Ασκήσεις Επανάληψης -2

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;

architecture Behavioral of LD_VHDL is

  signal C: std_logic;

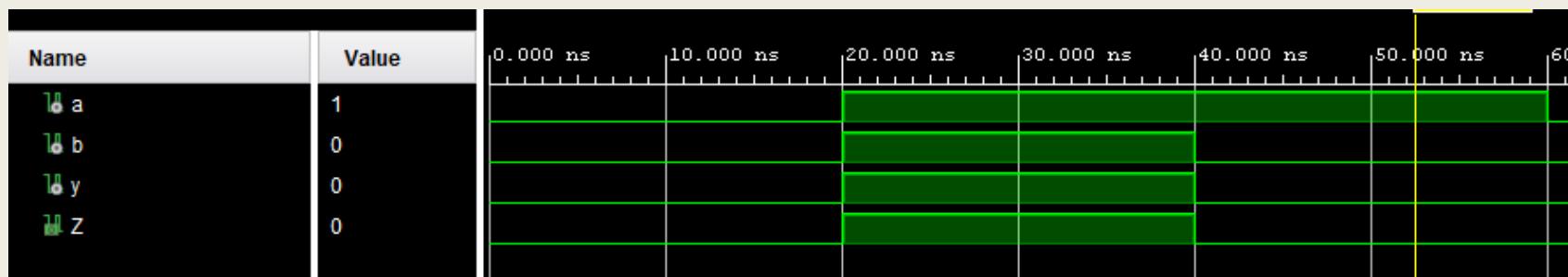
begin

  process (A, B, C) is
  begin

    Y<=A AND B;
    C<=B;
    Z<=A AND C;
  end process;

end Behavioral;
```

All Simulations



Ασκήσεις Επανάληψης -3

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;
```

architecture Behavioral of LD_VHDL is

```
signal C: std_logic;
```

Εντοπίστε τα λάθη

```
begin
```

```
process (A, B, C) is
```

```
variable D: std_logic;
begin
```

```
C:=B;
Y<=A AND C;
D<=A;
Z<=A AND C;
end process;
```

```
end Behavioral;
```

Ασκήσεις Επανάληψης -3

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;
```

architecture Behavioral of LD_VHDL is

```
  signal C: std_logic;
```

```
begin
```

```
  process (A, B, C) is
```

```
    variable D: std_logic;
```

```
    begin
```

```
      C:=B;
```

```
      Y<=A AND C;
```

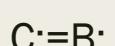
```
      D<=B;
```

```
      Z<=A AND D;
```

```
    end process;
```

```
  end Behavioral;
```

Εντοπίστε τα λάθη
(2)



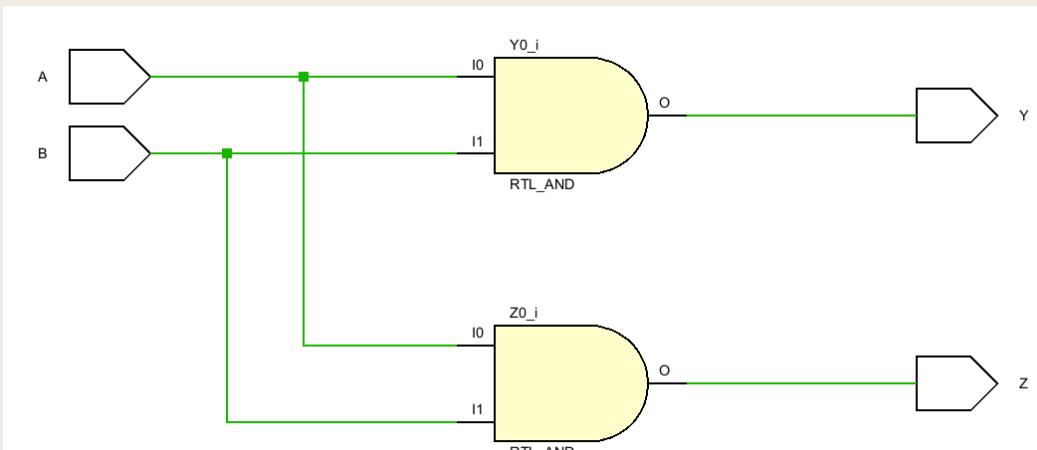
Ασκήσεις Επανάληψης -3

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;

architecture Behavioral of LD_VHDL is
  signal C: std_logic;
begin
  process (A, B, C) is
    variable D: std_logic;
  begin
    C<=B;
    Y<=A AND C;
    D:=B;
    Z<=A AND D;
  end process;
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη των A, B ανά 20ns
A='1', B='1'
A='0', B='0'
A='1', B='0'

Βρείτε πως μεταβάλλονται οι έξοδοι Y, Z, το σήμα C και η μεταβλητή D
Θεωρείστε ότι αρχικά ΟΛΑ τα σήματα είναι στην τιμή '0'



Ασκήσεις Επανάληψης -3

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z: out STD_LOGIC);
end LD_VHDL;

architecture Behavioral of LD_VHDL is
  signal C: std_logic;
begin
  process (A, B, C) is
    variable D: std_logic;
  begin
    C<=B;
    Y<=A AND C;
    D:=B;
    Z<=A AND D;
  end process;
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη των A, B ανά 20ns
A='1', B='1'
A='0', B='0'
A='1', B='0'

**Βρείτε πως μεταβάλλονται οι έξοδοι Y, Z, το σήμα C και η μεταβλητή D
Αρχικά A=B=C=D='0'**

A='1', B='1' => C='1', D='1', Y='0', Z='1'
Επειδή όμως άλλαξε τιμή το C (από '0'->'1')
το process εκτελείται ξανά και έχουμε:
A='1', B='1' => C='1', D='1', Y='1', Z='1'

Ασκήσεις Επανάληψης -3

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z:  out STD_LOGIC);
end LD_VHDL;

architecture Behavioral of LD_VHDL is
  signal C: std_logic;
begin
  process (A, B, C) is
    variable D: std_logic;
  begin
    C<=B;
    Y<=A AND C;
    D:=B;
    Z<=A AND D;
  end process;
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη των A, B ανά 20ns
A='1', B='1'
A='0', B='0'
A='1', B='0'

**Βρείτε πως μεταβάλλονται οι έξοδοι Y, Z, το σήμα C και η μεταβλητή D
Αρχικά A=B=C=D='0'**

A='1', B='1' => C='1', D='1', Y='1', Z='1'
A='0', B='0' => C='0', D='0', Y='0', Z='0'
(διαδικασία όπως προηγουμένως)

Ασκήσεις Επανάληψης -3

```
entity LD_VHDL is
  Port (
    A, B: in STD_LOGIC;
    Y, Z: out STD_LOGIC);
end LD_VHDL;

architecture Behavioral of LD_VHDL is
  signal C: std_logic;
begin
  process (A, B, C) is
    variable D: std_logic;
  begin
    C<=B;
    Y<=A AND C;
    D:=B;
    Z<=A AND D;
  end process;
end Behavioral;
```

Στο κύκλωμα εισέρχονται 3 ζεύγη των A, B ανά 20ns
A='1', B='1'
A='0', B='0'
A='1', B='0'

**Βρείτε πως μεταβάλλονται οι έξοδοι Y, Z, το σήμα C και η μεταβλητή D
Αρχικά A=B=C=D='0'**

A='1', B='1' => C='1', D='1', Y='1', Z='1'
A='0', B='0' => C='0', D='0', Y='0', Z='0'
A='1', B='0' => C='0', D='0', Y='0', Z='0'



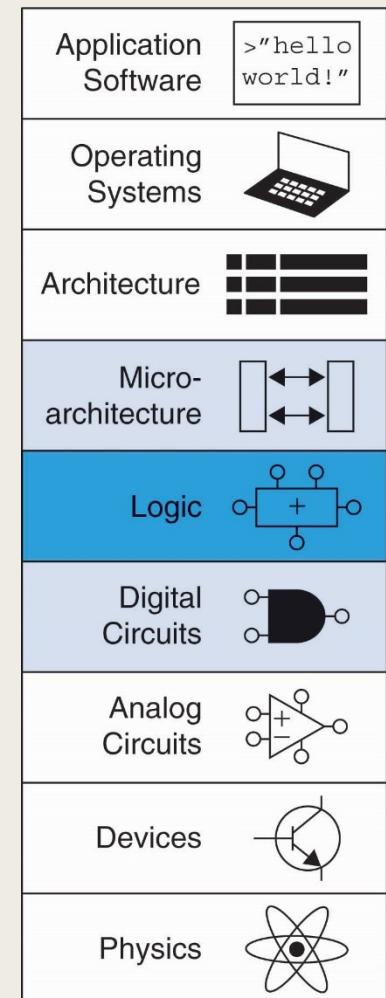
Κεφάλαιο 5

Ψηφιακά δομικά στοιχεία

Γιώργος Παπαδημητρίου, Αντώνης Πασχάλης,
Βασίλης Καρακώστας

Περιεχόμενα κεφαλαίου

- Αριθμητικά κυκλώματα
 - Αθροιστές – Αφαιρέτες
 - Συγκριτές
 - Αριθμητική/Λογική Μονάδα (ALU)
 - Ολισθητές και περιστροφείς
- Μετρητές
- Καταχωρητές ολίσθησης
- Αρχεία καταχωρητών
- Διατάξεις μνήμης (DRAM, SRAM, ROM)
- Διατάξεις λογικής (PLA, FPGA)



Αριθμητικά κυκλώματα

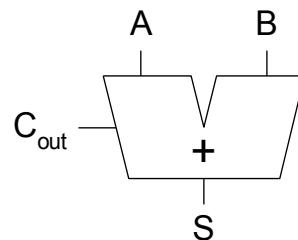
- Τα αριθμητικά κυκλώματα αποτελούν τα **κεντρικά δομικά στοιχεία** των υπολογιστών και των ψηφιακών συστημάτων γενικότερα
- Οι υπολογιστές και τα στοιχεία ψηφιακής λογικής εκτελούν πολλές **αριθμητικές λειτουργίες** ή **πράξεις**:
 - *πρόσθεση*
 - *αφαίρεση*
 - *συγκρίσεις*
 - *ολισθήσεις*
 - *πολλαπλασιασμό*
 - *διαίρεση*

Αριθμητικά κυκλώματα

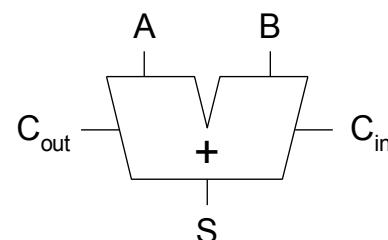
- Τα δομικά στοιχεία επιδεικνύουν τα τρια «A»: την **ιεραρχία**, την **τμηματικότητα** και την **κανονικότητα**
 - Συναρμολογούνται **ιεραρχικά από απλούστερα στοιχεία**, όπως οι λογικές πύλες, πολυπλέκτες και πλήρεις αθροιστές (FA)
 - Κάθε δομικό στοιχείο διαθέτει μια **καλώς ορισμένη διασύνδεση** (interface) και μπορεί να εκληφθεί ως «μαύρο κουτί»
 - Η κανονική δομή κάθε δομικού στοιχείου μπορεί **εύκολα να επεκταθεί** για διαφορετικά μεγέθη τελεστέων

Αθροιστής (adder) του ενός bit

Half
Adder



Full
Adder



A	B	C _{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\begin{aligned}S &= A \oplus B \\C_{\text{out}} &= AB\end{aligned}$$

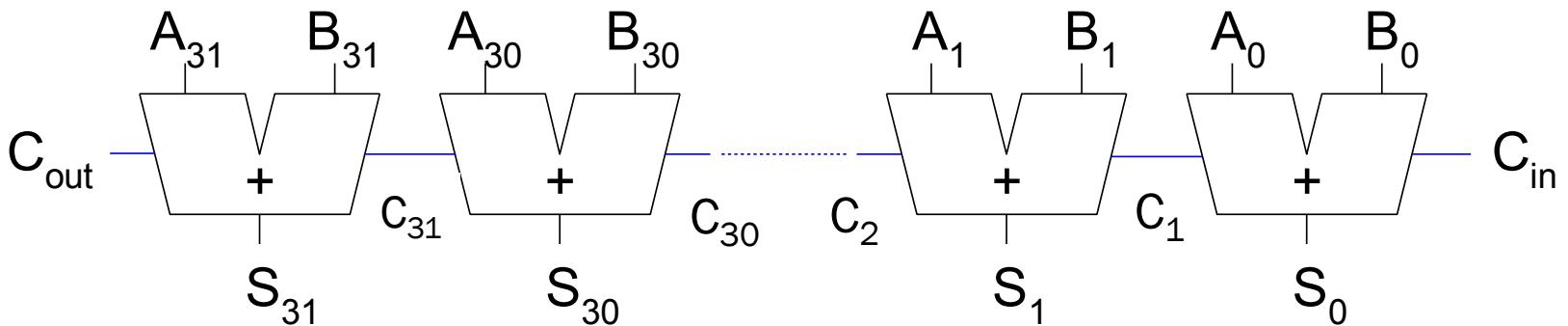
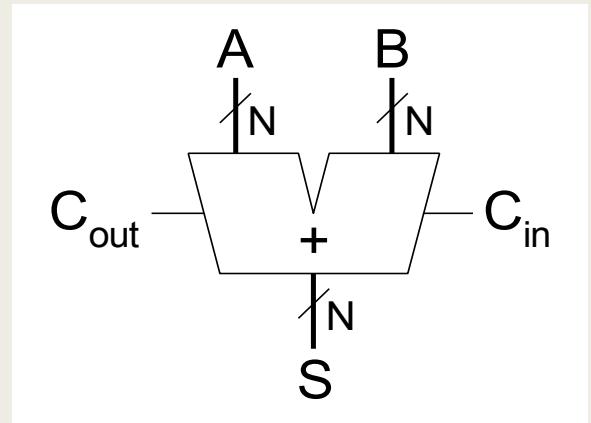
C _{in}	A	B	C _{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\begin{aligned}S &= A \oplus B \oplus C_{\text{in}} \\C_{\text{out}} &= AB + AC_{\text{in}} + BC_{\text{in}}\end{aligned}$$

Αθροιστής (adder) των N bit

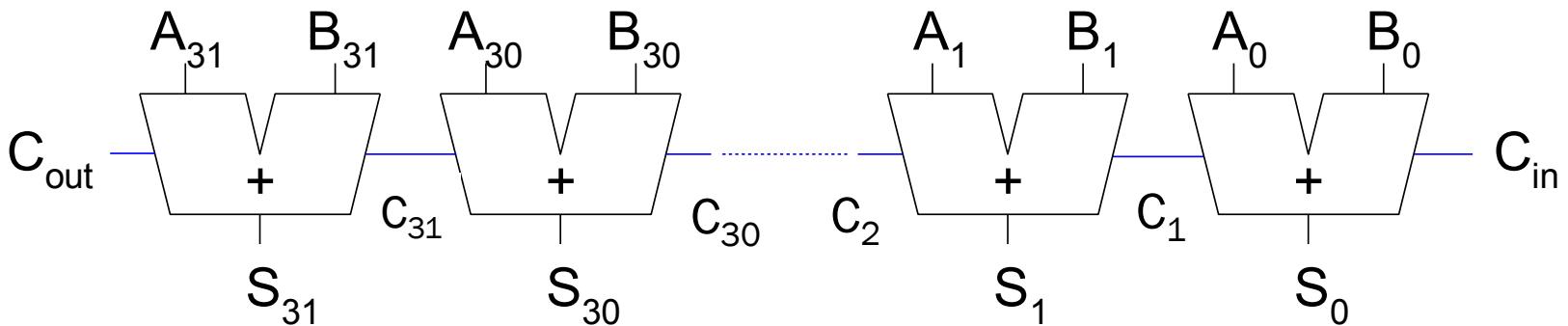
■ Αθροιστής κυμάτωσης κρατούμενου (ripple-carry adder - RCA)

- N πλήρεις αθροιστές του ενός bit συνδεδεμένοι αλυσιδωτά
- Το κρατούμενο διαδίδεται μέσω της αλυσίδας κρατουμένου
- Η έξοδος C_{out} του ενός βάρους χρησιμεύει ως είσοδος C_{in} του επόμενου βάρους
- Γενικά είναι αργός (χρησιμοποιείται μόνο στις διατάξεις FPGA)



Αθροιστής κυμάτωσης κρατουμένου

- Τέλεια εφαρμογή της τμηματικότητας και της κανονικότητας
 - η υπομονάδα του πλήρους αθροιστή επαναχρησιμοποιείται πολλές φορές για τον σχηματισμό ενός μεγαλύτερου συστήματος
- **Μειονέκτημα:** είναι αργός όταν το N έχει μεγάλη τιμή
 - Το S_{31} εξαρτάται από το A_0



Αθροιστής (adder) των N bit

- **Αθροιστής Πρόβλεψης Κρατουμένου (carry lookahead adder - CLA)**
 - Διαχειρίζει τον αθροιστή σε τμηματικούς αθροιστές (CLA block) μικρότερου μεγέθους (π.χ. των 4 bit)
 - Το κρατούμενο εξόδου του τμηματικού αθροιστή υπολογίζεται γρήγορα αμέσως μόλις γίνει γνωστή η τιμή του κρατουμένου εισόδου του
 - Βασίζεται στα **σήματα παραγωγής και διάδοσης κρατούμενου**, που παράγονται κατά την πρόσθεση ενός ζεύγους εισόδων A_i και B_i με το αντίστοιχο κρατούμενο εισόδου C_i στον **πλήρη αθροιστή**
 - Εάν $A_i = 1$ και $B_i = 1$, τότε πάντα $C_{i+1} = 1$, ανεξάρτητα του C_i
 - Ορίζεται το **σήμα παραγωγής κρατούμενου $G_i = A_i \cdot B_i$**
 - Εάν $A_i = 1$ ή $B_i = 1$, τότε $C_{i+1} = 1$ μόνο όταν $C_i = 1$
 - Ορίζεται το **σήμα διάδοσης κρατούμενου $P_i = A_i + B_i$**
 - Συνεπώς για το **κρατούμενο εξόδου C_{i+1}** στον πλήρη αθροιστή ισχύει:

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

Αθροιστής πρόβλεψης κρατουμένου

- Για έναν τμηματικό αθροιστή (CLA block) των 4 bit ισχύουν τα ακόλουθα:

$$C_4 = G_3 + P_3 C_3$$

$$C_3 = G_2 + P_2 C_2$$

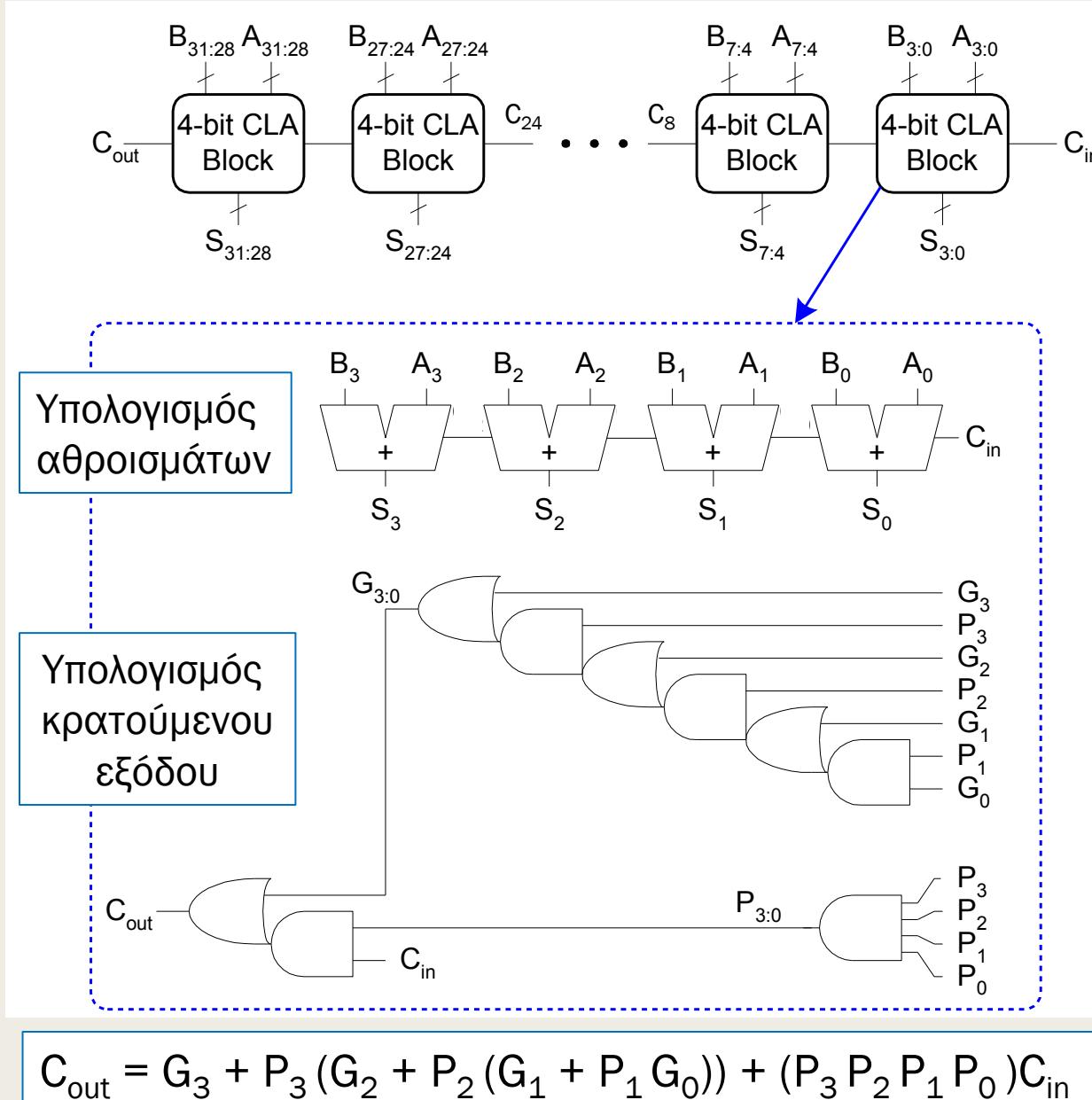
$$C_2 = G_1 + P_1 C_1$$

$$C_1 = G_0 + P_0 C_0$$

$$C_4 = G_3 + P_3 (G_2 + P_2 (G_1 + P_1 (G_0 + P_0 C_0)))$$

$$C_{out} = G_3 + P_3 (G_2 + P_2 (G_1 + P_1 G_0)) + (P_3 P_2 P_1 P_0) C_{in}$$

Αθροιστής πρόβλεψης κρατούμενου

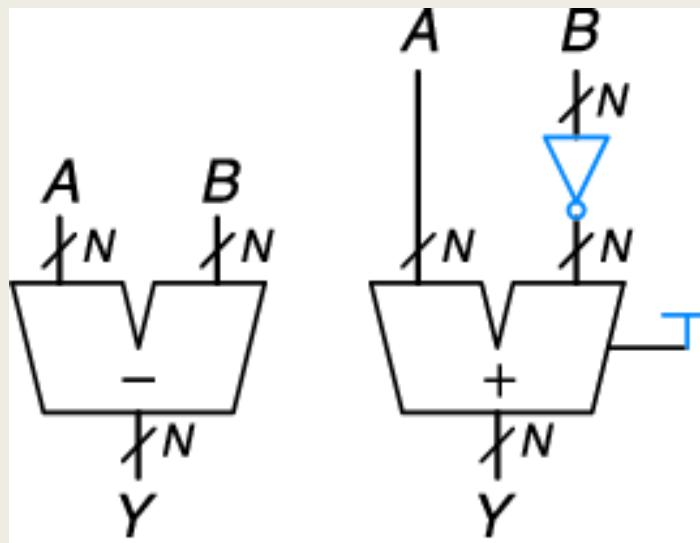


Αθροιστής πρόβλεψης κρατουμένου

- Σε όλους τους τμηματικούς αθροιστές όλα τα σήματα παραγωγής και διάδοσης κρατούμενου δημιουργούνται παράλληλα.
- Η κρίσιμη διαδρομή ξεκινάει με τον υπολογισμό των G_0 και $G_{3:0}$ στον πρώτο τμηματικό αθροιστή (CLA block)
- Κατόπιν το C_{in} ρέει απευθείας προς το C_{out} μέσω των πυλών AND-OR σε κάθε επόμενο τμηματικό αθροιστή μέχρι και τον τελευταίο
- Στην περίπτωση ενός μεγάλου αθροιστή, αυτό είναι πολύ πιο γρήγορο από ό,τι η αναμονή να κυματωθούν τα κρατούμενα σε κάθε διαδοχικό bit του αθροιστή
- Τέλος, η κρίσιμη διαδρομή μέσω του τελευταίου τμηματικού αθροιστή περιέχει έναν μικρό αθροιστή κυμάτωσης κρατουμένου των 4 bit που υπολογίζει τα αθροίσματα $S_{31:28}$ όταν εμφανισθεί το C_{28}

Αφαιρέτης (subtractor) των N bit

- Οι αθροιστές μπορούν να προσθέτουν θετικούς και αρνητικούς αριθμούς συμπληρώματος ως προς δύο
- Για να πραγματοποιήσουμε την αφαίρεση αντιστρέφουμε τα bit του αφαιρετέου και το προσθέτουμε στο μειωτέο ξεκινώντας με κρατούμενο εισόδου 1, αντί για 0



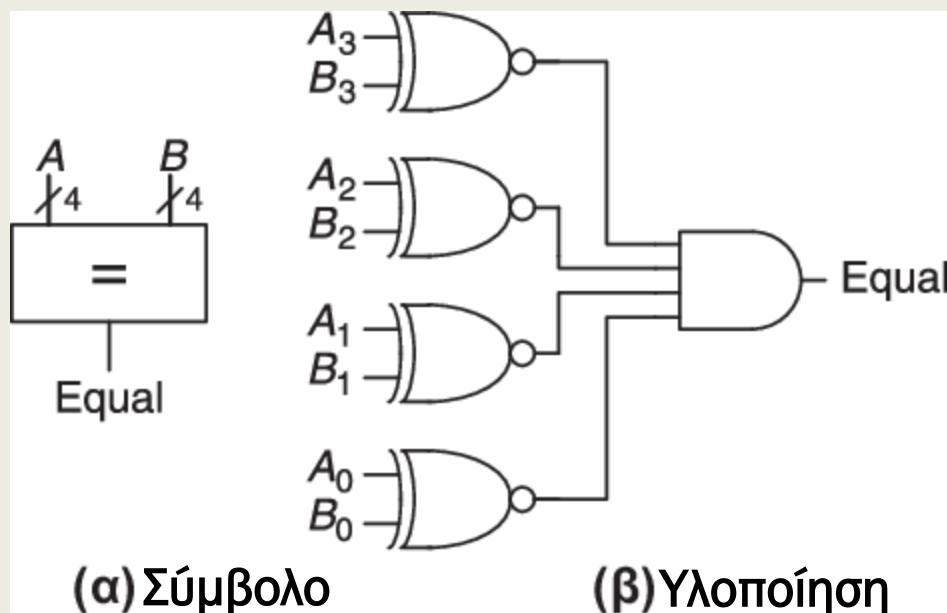
Στην υλοποίηση του αφαιρέτη χρησιμοποιούμε έναν αθροιστή

Συγκριτής (comparator) των N bit

- Ο **συγκριτής** (comparator) των N bit εξακριβώνει αν δύο δυαδικοί αριθμοί των N bit είναι ίσοι ή αν ο ένας είναι μεγαλύτερος ή μικρότερος από τον άλλο
- Υπάρχουν δύο διαδεδομένοι τύποι συγκριτών
 - Ο **συγκριτής ισότητας** (*equality comparator*) παράγει μία έξοδο που υποδεικνύει αν το A είναι ίσο με το B ($A = B$)
 - Ο **συγκριτής μεγέθους** (*magnitude comparator*) παράγει μία ή περισσότερες εξόδους που υποδεικνύουν τις σχετικές τιμές των A και B

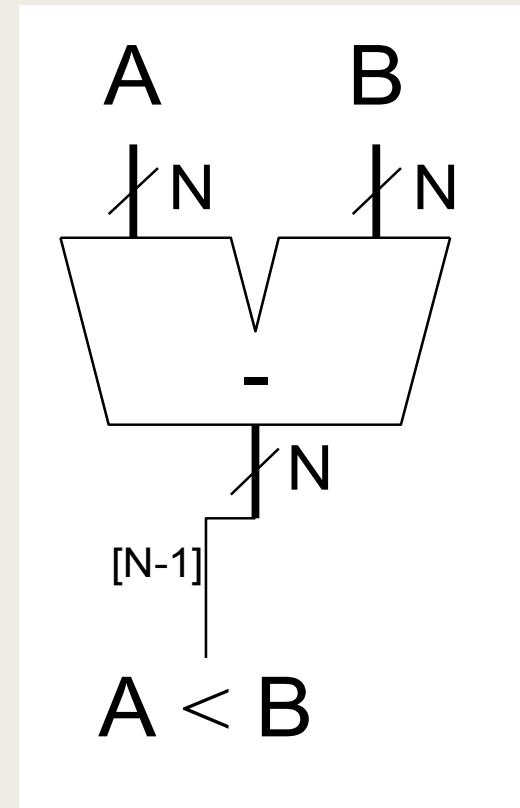
Συγκριτής ισότητας των 4 bit

- Ο συγκριτής πρώτα χρησιμοποιεί πύλες XNOR για να ελέγξει αν τα αντίστοιχα bit σε κάθε στήλη των A και B είναι ίσα
 - Οι αριθμοί είναι ίσοι αν τα αντίστοιχα bit σε όλες τις στήλες είναι ίσα
 - Έστω $A = (A_3, A_2, A_1, A_0)$ και $B = (B_3, B_2, B_1, B_0)$
 - **$A_3 = B_3$ και $A_2 = B_2$ και $A_1 = B_1$ και $A_0 = B_0$, τότε $A = B$**
 - Η έξοδος *Equal* του συγκριτή ισότητας παίρνει την τιμή 1 όταν δύο δυαδικοί αριθμοί A και B είναι μεταξύ τους ίσοι



Συγκριτής μεγέθους των N bit

- Η σύγκριση μεγέθους για προσημασμένους αριθμούς συνήθως πραγματοποιείται με τον υπολογισμό του $A - B$ και την εξέταση του προσήμου (δηλαδή του περισσότερο σημαντικού bit) του αποτελέσματος
 - Αν το αποτέλεσμα είναι αρνητικό, τότε το A είναι μικρότερο από το B
 - Διαφορετικά, το A είναι μεγαλύτερο από ή ίσο με B
 - *Η ισότητα προκύπτει όταν το αποτέλεσμα είναι μηδέν*
 - **Προσοχή!** Στην υπερχείλιση. Αντιστρέφει το πρόσημο του αποτελέσματος και απαιτείται διόρθωση μετά την ανίχνευσή της



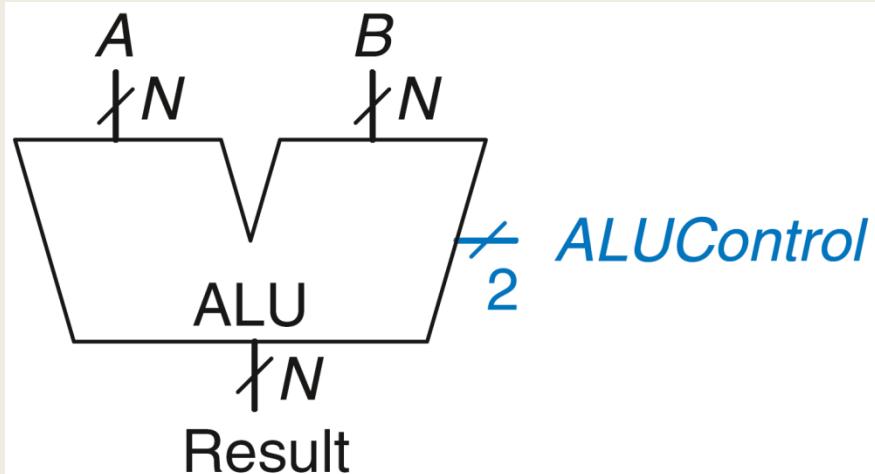
Συγκριτής Μεγέθους: Παράδειγμα

- Θεωρείστε ότι οι αριθμοί είναι μεγέθους 4 bit
- Έστω $A = 3$, $B = 5$. Τότε $3_{10} + (-5_{10}) = -2_{10}$
 - $3_{10} = 0011_2$ και $5_{10} = 0101_2$
 - $-5_{10} = 1010_2 + 1 = 1011_2$
 - $0011_2 + 1011_2 = \textcolor{blue}{1110}_2$
 - *To bit προσήμου είναι «1», άρα $A < B$ (Σωστό!)*
- Έστω $A = -3$, $B = 6$. Τότε $-3_{10} + (-6_{10}) = -9_{10}$ (**υπερχείλιση**)
 - $-3_{10} = 1101_2$ και $6_{10} = 0110_2$
 - $-6_{10} = 1001_2 + 1 = 1010_2$
 - $1101_2 + 1010_2 = \textcolor{red}{0111}_2$
 - *To bit προσήμου είναι «0», άρα $A > B$ (Λάθος! Λόγω υπερχείλισης)*

Αριθμητική/Λογική Μονάδα (ALU)

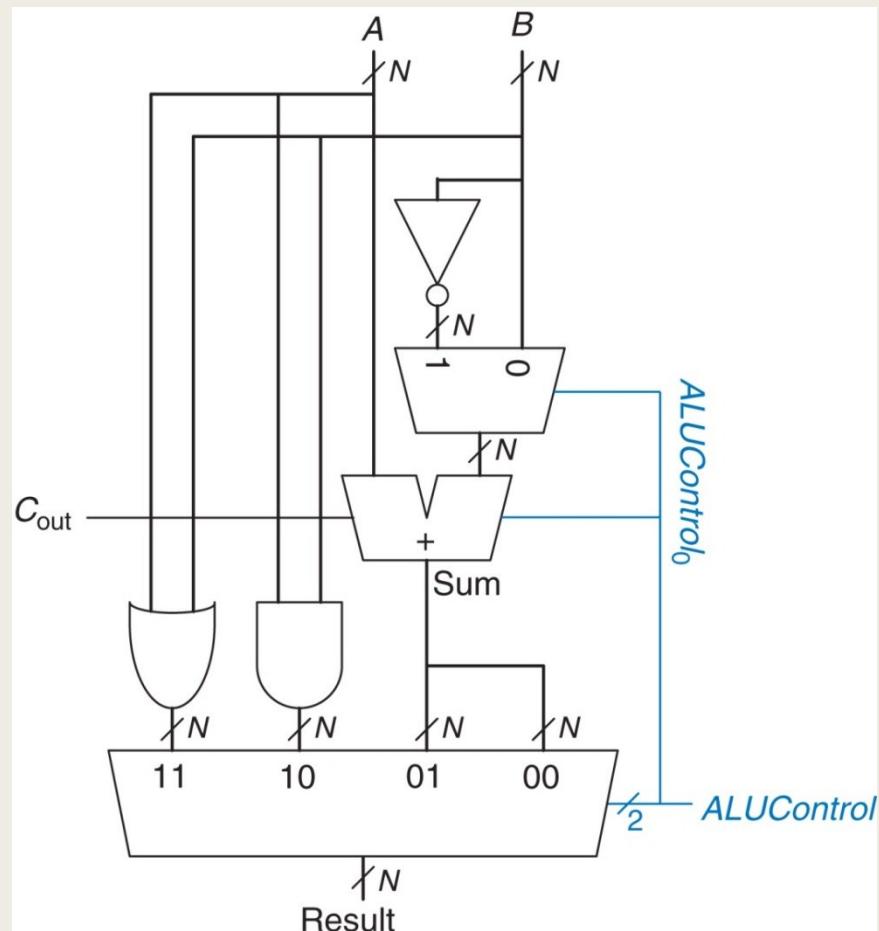
- Μια **Αριθμητική/Λογική Μονάδα** (Arithmetic/Logical Unit, ALU) υλοποιεί ποικίλες αριθμητικές και λογικές πράξεις σε μία μονάδα.
 - Πρόσθεσης, αφαίρεσης προσημασμένων ακεραίων αριθμών
 - Λογικές πράξεις ανά bit (π.χ. AND και OR)
- Η μονάδα ALU αποτελεί τον πυρήνα των περισσότερων υπολογιστικών συστημάτων
- Παράδειγμα μίας απλής μονάδας ALU που εκτελεί τις πράξεις που φαίνονται στον πίνακα σύμφωνα με την τιμή του σήματος ελέγχου **ALUControl** των 2 bit

ALUControl	Πράξη
00	Add
01	Subtract
10	AND
11	OR



Υλοποίηση μίας απλής μονάδας ALU

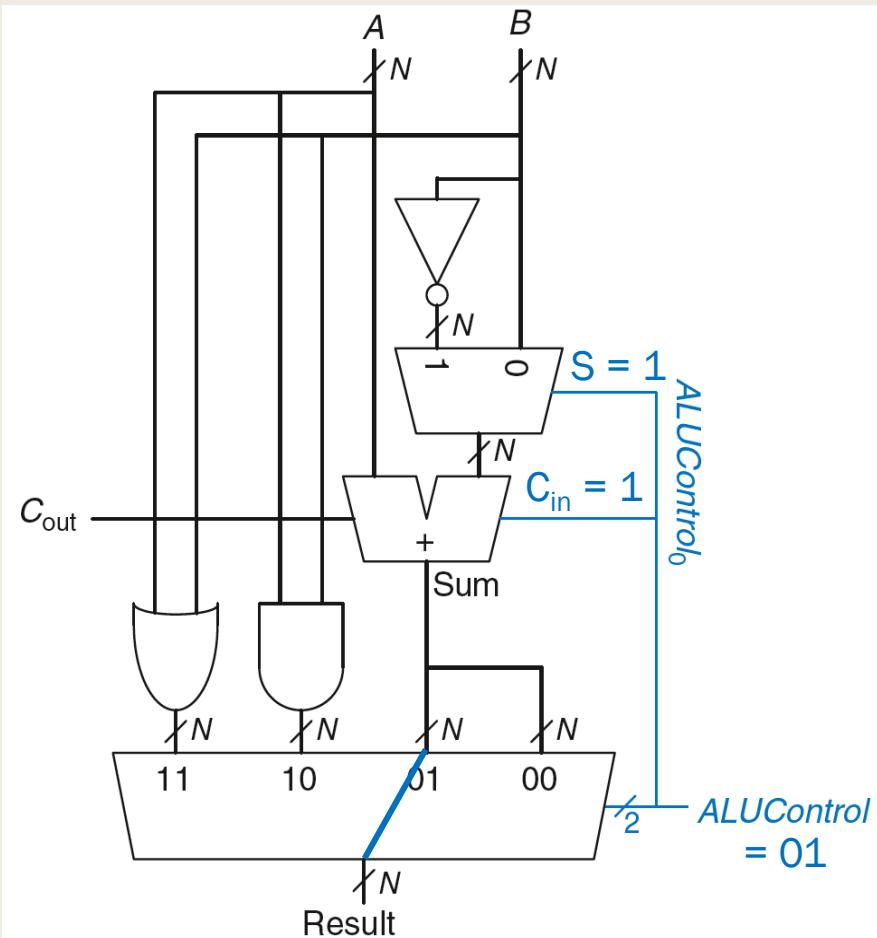
- Εσωτερικά η μονάδα ALU περιέχει:
 - Έναν **αθροιστή των N bit** με εισόδους A , B και έξοδο Sum των N bit
 - **N πύλες AND** με δύο εισόδους
 - **N πύλες OR** με δύο εισόδους
 - **N αντιστροφείς** και έναν **πολυπλέκτη 2 σε 1 των N bit**, ώστε η είσοδος B του αθροιστή να αντιστρέφεται όταν ενεργοποιείται το $ALUControl_0$
 - Έναν **πολυπλέκτη 4 σε 1 των N bit** που επιλέγει την επιθυμητή πράξη με βάση το σήμα $ALUControl$



Υλοποίηση μίας απλής μονάδας ALU

- Παράδειγμα: Εκτέλεση της αριθμητικής πράξης **Subtract** (Result = A - B)
 - $ALUControl = 01$ και $ALUControl_0 = S = Cin = 1$
 - Η είσοδος B του αθροιστή αντιστρέφεται
 - Ο αθροιστής εκτελεί την πράξη της αφαίρεσης
 - Ο πολυπλέκτης 4 σε 1 επιλέγει την είσοδο 01 ως έξοδο της ALU

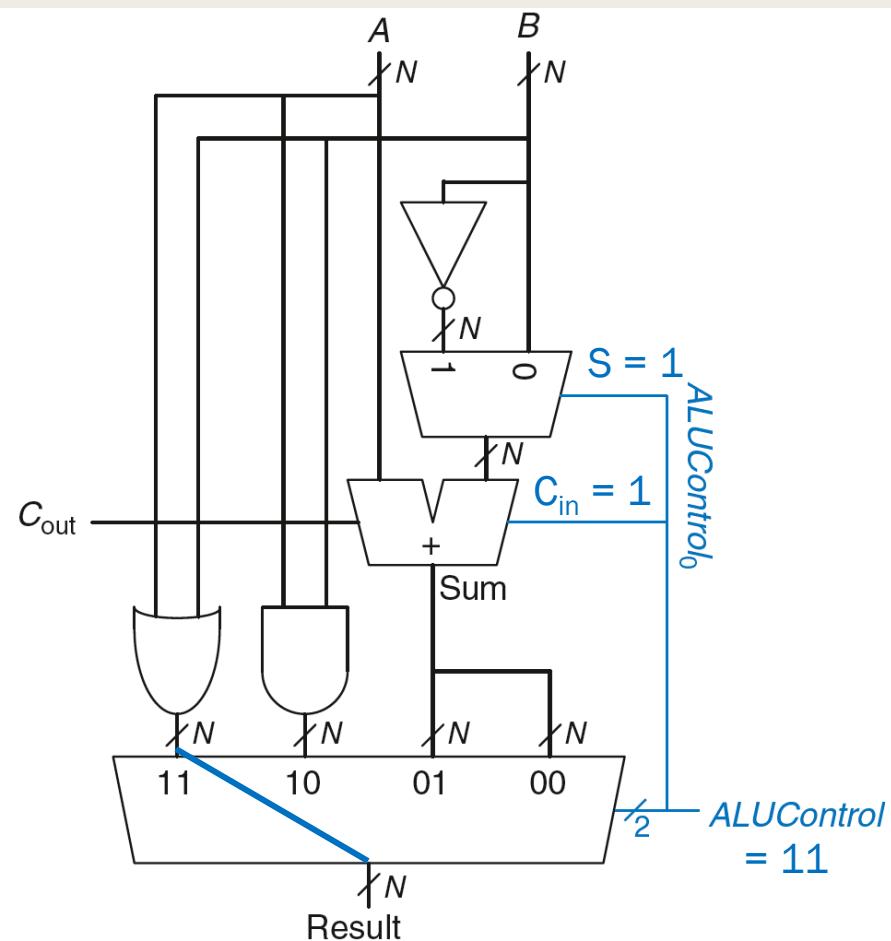
$ALUControl_{1:0}$	Πράξη
00	Add
01	Subtract
10	AND
11	OR



Υλοποίηση μίας απλής μονάδας ALU

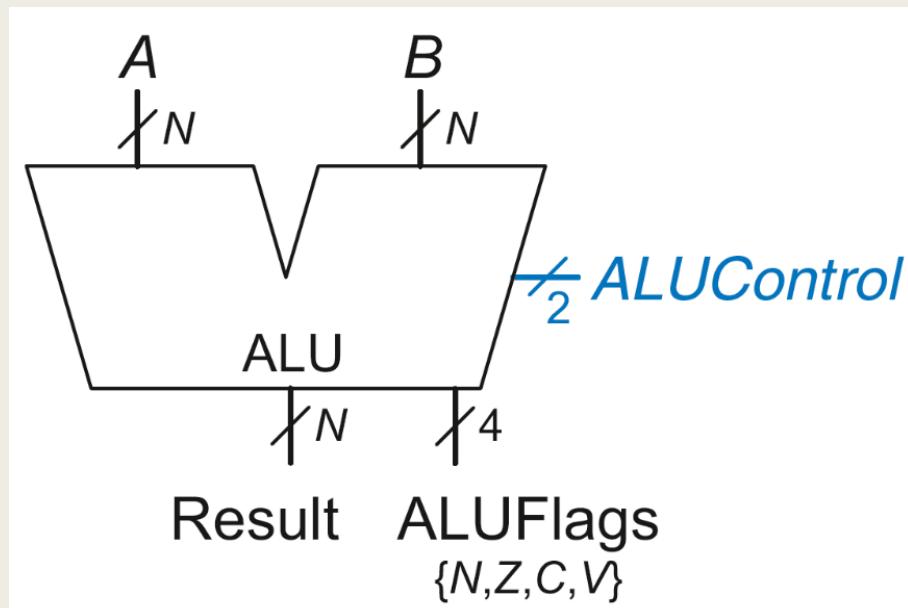
- Παράδειγμα: Εκτέλεση της λογικής πράξης **OR** (Result = A or B)
 - $ALUControl = 11$ και $ALUControl_0 = S = Cin = 1$
 - Η είσοδος B του αθροιστή αντιστρέφεται
 - Ο αθροιστής εκτελεί την πράξη της αφαίρεσης
 - Ο πολυπλέκτης 4 σε 1 επιλέγει την είσοδο 11 ως έξοδο της ALU

$ALUControl_{1:0}$	Πράξη
00	Add
01	Subtract
10	AND
11	OR



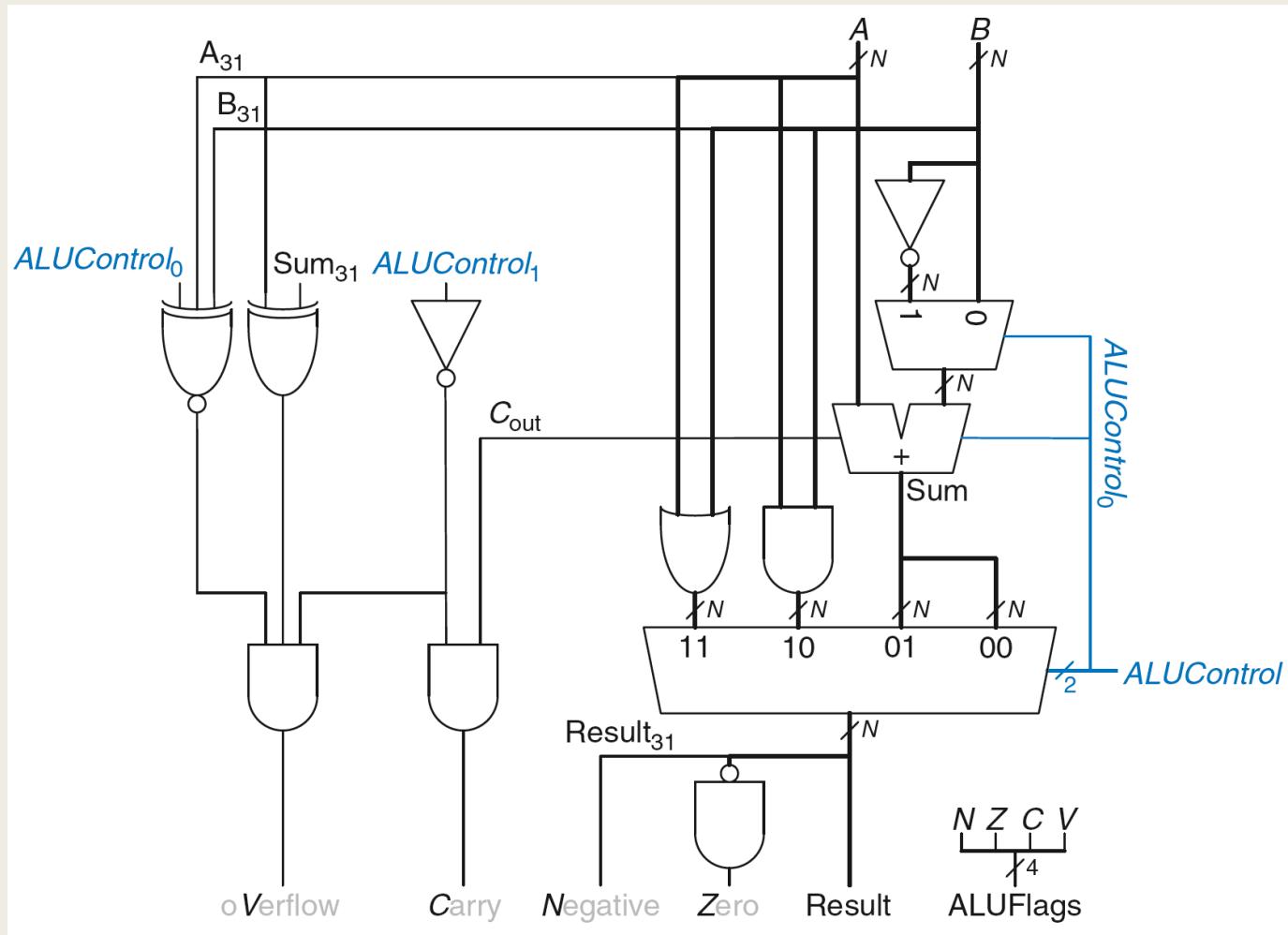
Σημαίες (ALU Status Flags)

Σημαία	Περιγραφή
N	Αρνητικό αποτέλεσμα (Negative)
Z	Μηδενικό αποτέλεσμα (Zero)
C	Ο αθροιστής παρήγαγε κρατούμενο εξόδου (Carry)
V	Ο αθροιστής υπερχείλισε (Overflowed)



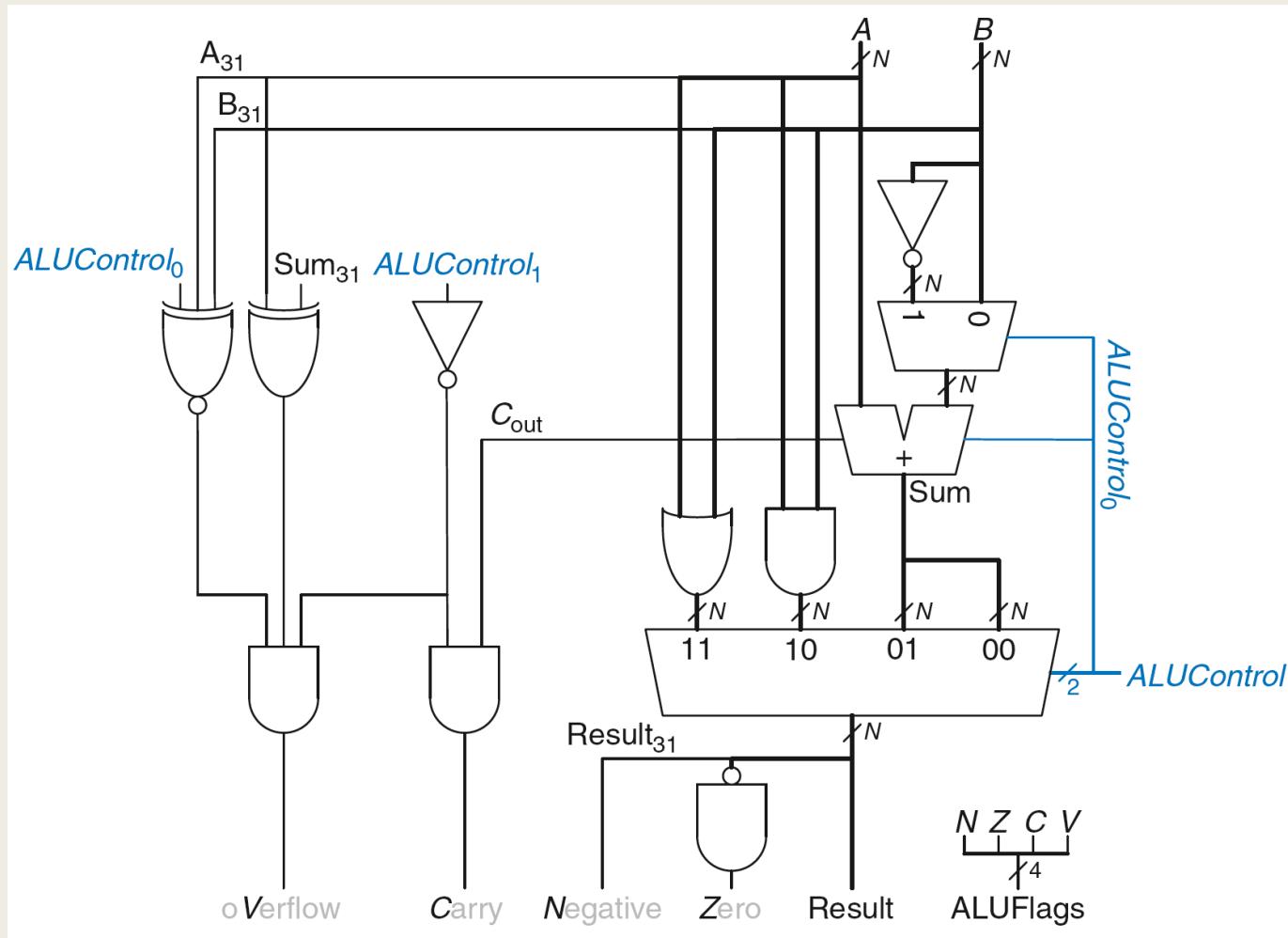
Σημαίες: Negative

- **N = 1**, εάν το αποτέλεσμα της μονάδας ALU είναι αρνητικό
- Το N είναι συνδεδεμένο με το πιο σημαντικό bit (MSB) του Result



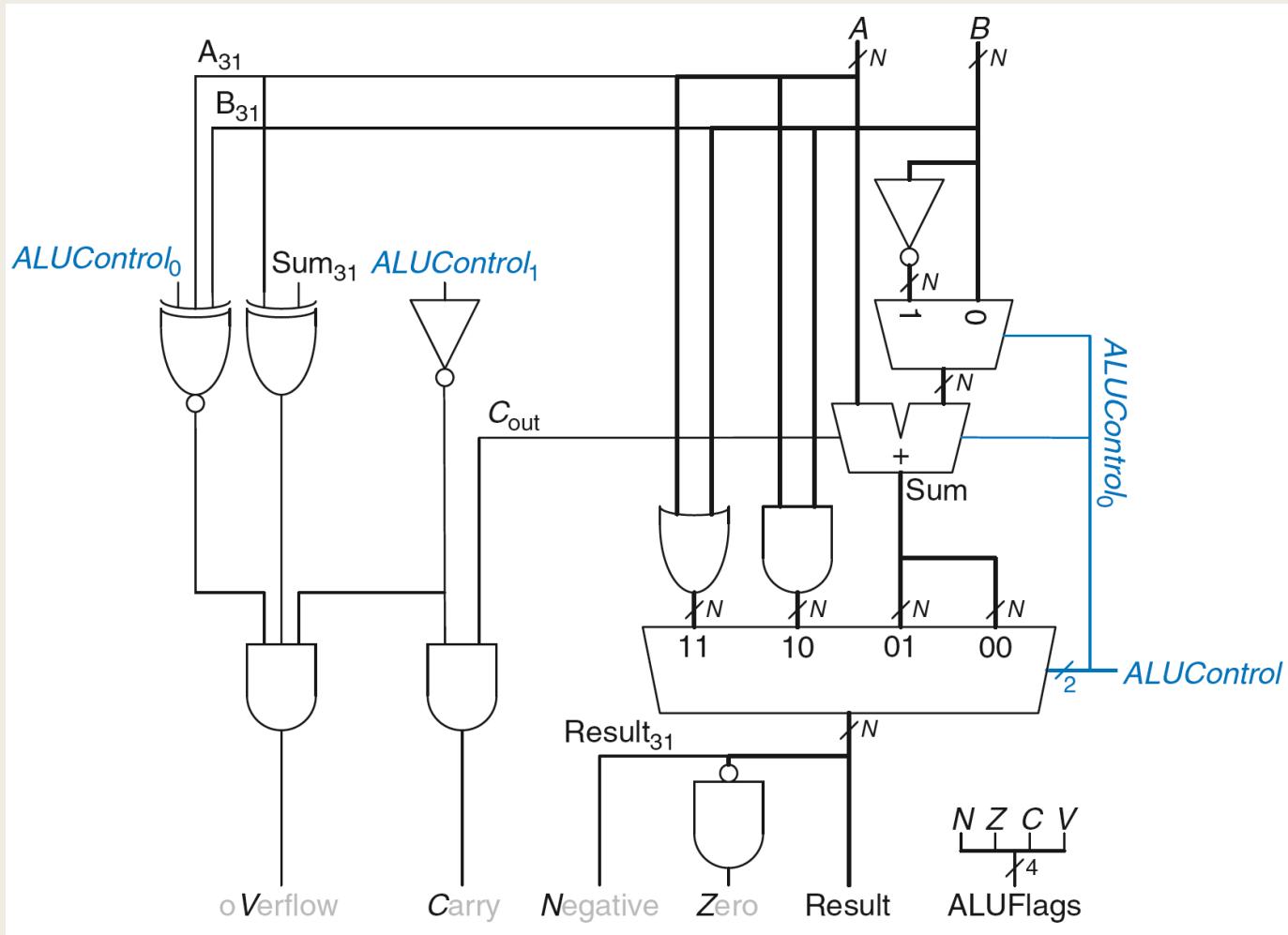
Σημαίες: Zero

- $Z = 1$, εάν όλα τα bit του Result είναι 0
- Το Z παράγεται από μία **πύλη NOR** των N bit



Σημαίες: Carry

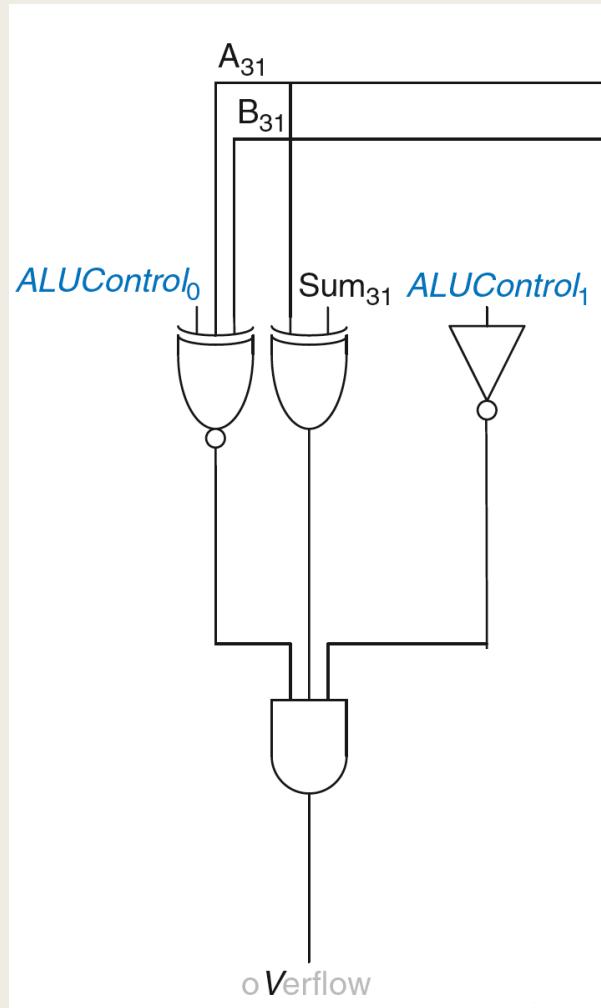
- $C = 1$, εάν ο αθροιστής παράγει κρατούμενο εξόδου (Cout) **KAI** η μονάδα ALU εκτελεί πρόσθεση ή αφαίρεση ($ALUControl_1 = 0$)



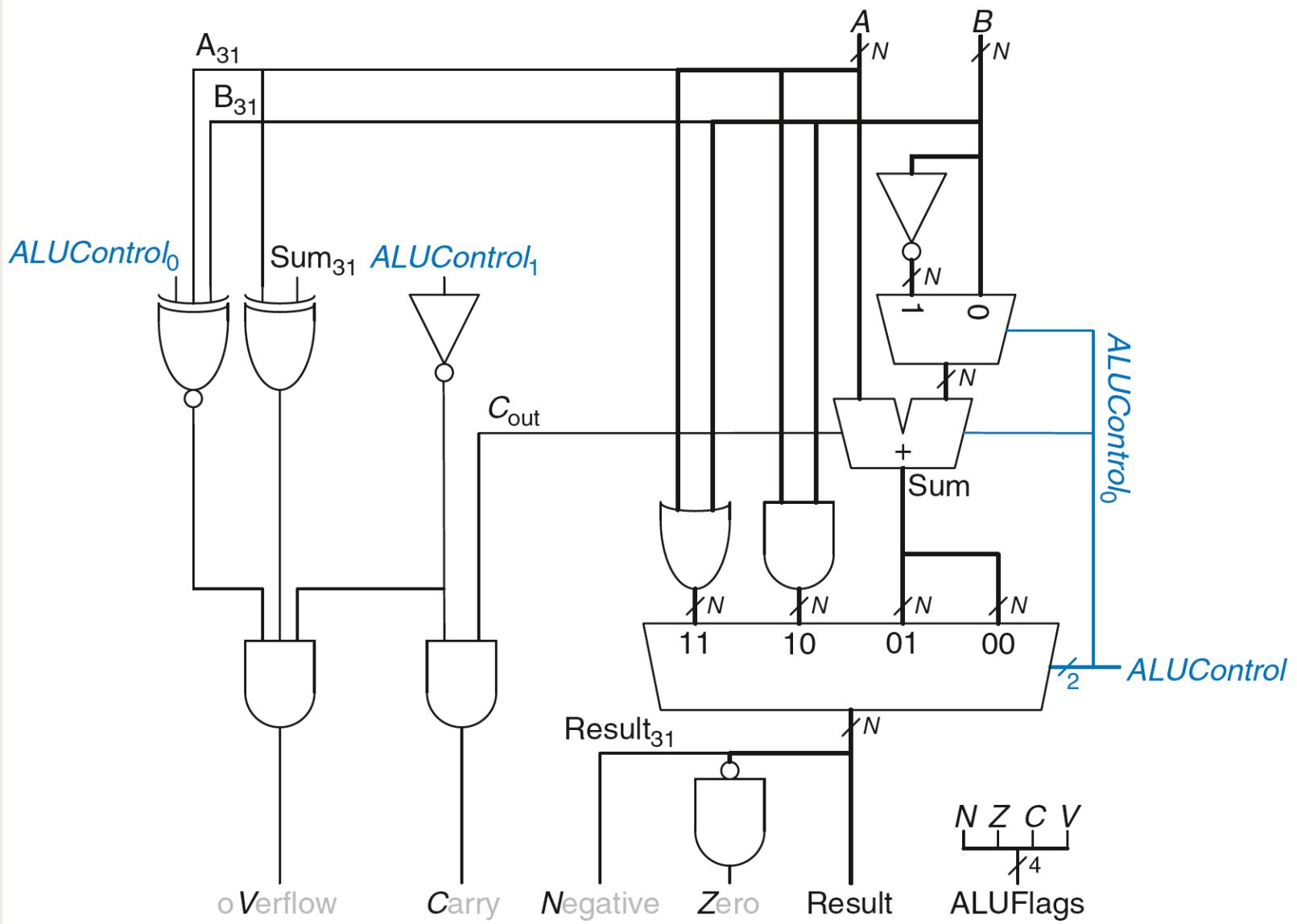
Σημαίες: oVerflow

- **V = 1**, εάν το αποτέλεσμα της αριθμητικής πράξης ($ALUControl_1 = 0$) έχει υπερχειλίσει

- Η υπερχειλίση εμφανίζεται όταν δύο ομόσημοι προσημασμένοι αριθμοί παράγουν άθροισμα διαφορετικού προσήμου
- Ισχύουν οι συνθήκες:
 - $ALUControl_1 = 0$ και
 - $A_{31} \text{ xor } Sum_{31} = 1$ και
 - $A_{31} \text{ xnor } B_{31} = 1$ και $ALUControl_0 = 0$ (+)
(000, 110)
- ή
- $A_{31} \text{ xnor } B_{31} = 0$ και $ALUControl_0 = 1$ (-)
(011, 101)



Σημαίες (ALU Status Flags)



Ολισθητές και περιστροφείς

- Οι **ολισθητές** (shifters) και οι **περιστροφείς** (rotators) μετακινούν bit
- Ο ολισθητής ολισθαίνει έναν δυαδικό αριθμό προς τα αριστερά ή προς τα δεξιά κατά έναν καθορισμένο αριθμό θέσεων, έστω n
 - πολλαπλασιάζουν ή διαιρούν με δυνάμεις του 2 (2^n)
- Ο περιστροφέας περιστρέφει τον αριθμό σε έναν κύκλο έτσι, ώστε οι κενές θέσεις να συμπληρώνονται με bit που ολισθαίνουν από το άλλο άκρο

Ολισθητές (shifters)

■ Λογικός ολισθητής

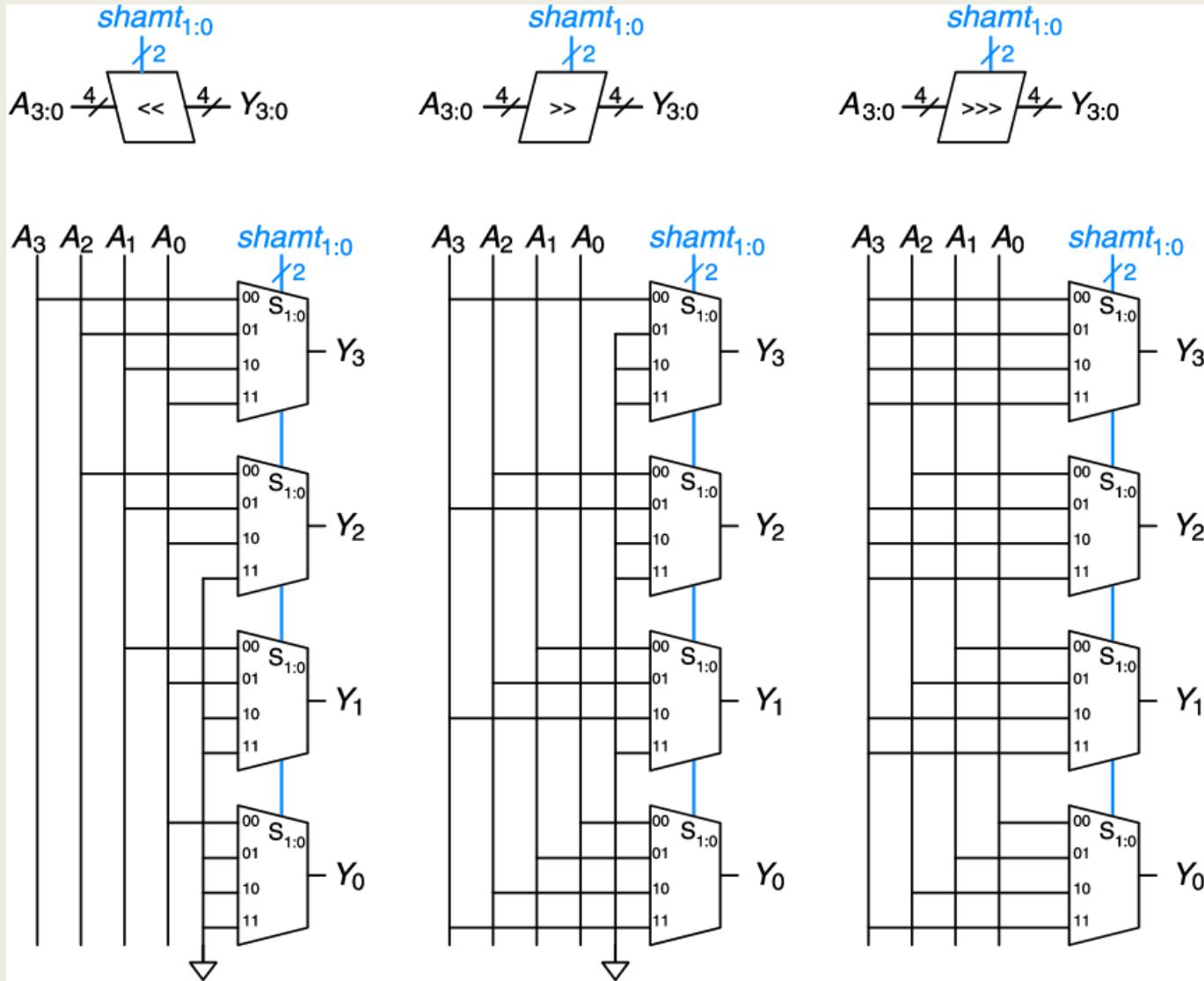
- Ολισθαίνει τον αριθμό προς τα αριστερά (*logical shift left, LSL, <<*) και συμπληρώνει τις κενές θέσεις με **μηδενικά**
 - **11001** << 2 => **00100**
- Ολισθαίνει τον αριθμό προς τα δεξιά (*logical shift right, LSR, >>*) και συμπληρώνει τις κενές θέσεις με **μηδενικά**
 - **11001** >> 2 => 00**110**

■ Αριθμητικός ολισθητής

- Ολισθαίνει τον προσημασμένο αριθμό προς τα αριστερά (*arithmetic shift left, ASL*) και συμπληρώνει τις κενές θέσεις με **μηδενικά** (όπως *LSL, <<*)
 - **00001** << 2 => **00100** (πολλαπλασιάζει κατά 4, από 1 σε 4)
- Ολισθαίνει τον προσημασμένο αριθμό προς τα δεξιά (*arithmetic shift right, ASR, >>>*) και συμπληρώνει τις κενές θέσεις με το **bit προσήμου** (0 εάν θετικός, 1 εάν αρνητικός)
 - **11000** >>> 2 => 11**110** (διαιρεί δια 4, από -8 σε -2)

Ολισθητές (shifters)

- Ένας ολισθητής των N bit υλοποιείται με N πολυπλέκτες N σε 1 (π.χ. $N = 4$)



Περιστροφέίς (rotators)

■ Περιστροφέας

- Περιστρέφει τον αριθμό σε έναν κύκλο έτσι ώστε οι κενές θέσεις να συμπληρώνονται με bit που ολισθαίνουν από το άλλο άκρο
- Η περιστροφή γίνεται είτε προς τα δεξιά (rotate right, ROR), είτε προς τα αριστερά (rotate left, ROL)
- Παραδείγματα:
 - 11001 ROR 2 = 01110
 - 11001 ROL 2 = 00111
- Ένας περιστροφέας των N bit υλοποιείται με N πολυπλέκτες N σε 1 (όπως και οι ολισθητές)
 - με κατάλληλη σύνδεση των εισόδων A

Δυαδικός πολλαπλασιασμός

- Ο πολλαπλασιασμός μη προσημασμένων δυαδικών αριθμών είναι παρόμοιος με τον πολλαπλασιασμό δεκαδικών αριθμών, με τη διαφορά ότι περιλαμβάνει μόνο άσους και μηδενικά
 - *Και στις δύο περιπτώσεις, σχηματίζονται μερικά γινόμενα μέσω του πολλαπλασιασμού ενός μόνο ψηφίου του πολλαπλασιαστή με ολόκληρο τον πολλαπλασιαστέο*
 - *Τα ολισθημένα μερικά γινόμενα αθροίζονται ώστε να προκύψει το αποτέλεσμα*

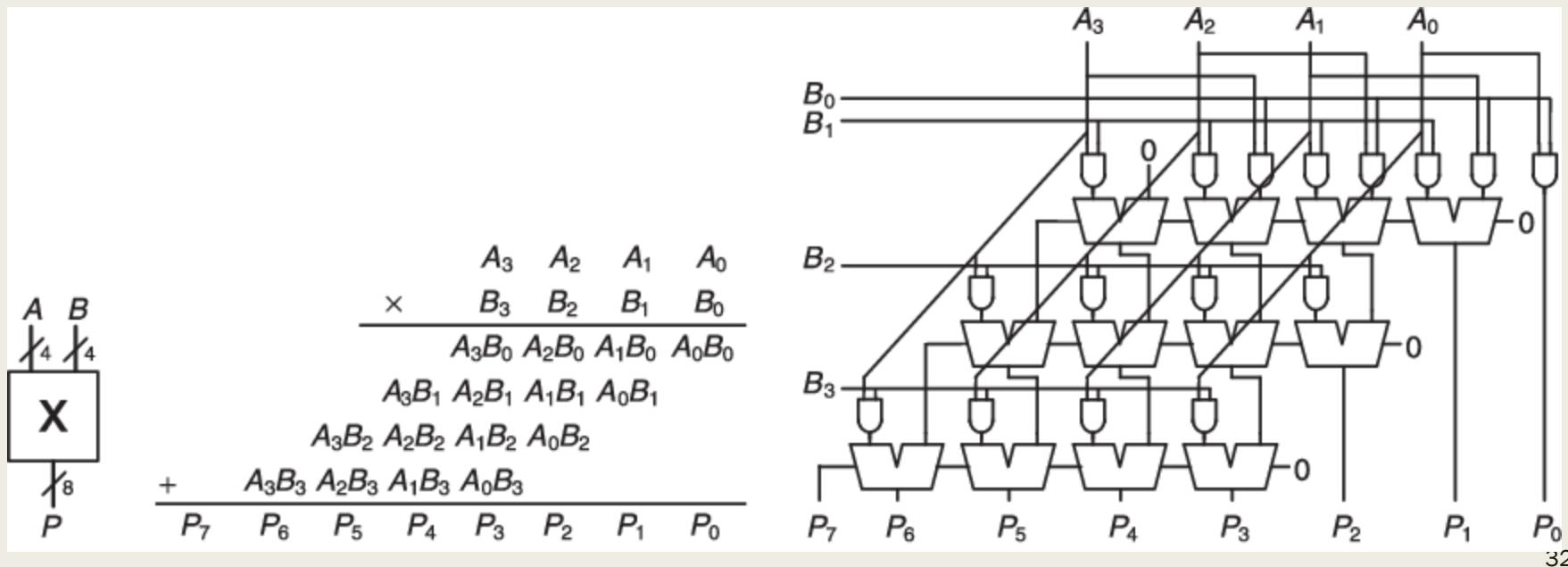
$\begin{array}{r} 230 \\ \times \quad 42 \\ \hline 460 \\ + 920 \\ \hline 9660 \end{array}$	<p>πολλαπλασιαστέος πολλαπλασιαστής μερικά γινόμενα</p>	$\begin{array}{r} 0101 \\ \times \quad 0111 \\ \hline 0101 \\ 0101 \\ 0101 \\ + 0000 \\ \hline 0100011 \end{array}$
		<p>αποτέλεσμα</p>

$$230 \times 42 = 9660$$

$$5 \times 7 = 35$$

Δυαδικός πολλαπλασιαστής

- Ένας δυαδικός πολλαπλασιαστής $N \times N$ πολλαπλασιάζει δύο αριθμούς των N bit και παράγει ένα αποτέλεσμα μεγέθους $2N$ bit
- Τα μερικά γινόμενα στον δυαδικό πολλαπλασιασμό είναι είτε ο πολλαπλασιαστέος είτε μια τιμή που περιέχει μόνο μηδενικά
- Ο πολλαπλασιασμός δυαδικών αριθμών μεγέθους 1 bit είναι ισοδύναμος με τη πράξη AND
- Τα μερικά γινόμενα παράγονται με πύλες AND και αθροίζονται με παρατάξεις (array) πλήρων αθροιστών (full adders)



Δυαδικός πολλαπλασιαστής

- Ο πολλαπλασιασμός προσημασμένων και μη προσημασμένων αριθμών διαφέρουν μεταξύ τους
- Παράδειγμα, έστω το γινόμενο $P = 0xFE \times 0xFD$
 - εάν ερμηνευθούν ως προσημασμένοι ακέραιοι, τότε:
 - $0xFE = -2$, $0xFD = -3$ και $P = 0x0006$
 - εάν ερμηνευθούν ως μη προσημασμένοι ακέραιοι, τότε:
 - $0xFE = 254$, $0xFD = 253$ και $P = 0xFB06$
 - και στις δύο περιπτώσεις, το λιγότερο σημαντικό byte είναι ίδιο (το $0x06$)

Πραγματικοί αριθμοί σταθερής υποδιαστολής

- Στην αναπαράσταση **σταθερής υποδιαστολής**, οι πραγματικοί αριθμοί είναι παρόμοιοι με τους δεκαδικούς αριθμούς
 - υποδηλώνεται η ύπαρξη μίας **υποδιαστολής** (.)
 - τα bit **αριστερά** της υποδιαστολής αναπαριστούν το **ακέραιο μέρος**
 - τα bit **δεξιά** της υποδιαστολής αναπαριστούν το **κλασματικό μέρος**
- Αναπαράσταση σταθερής υποδιαστολής μη προσημασμένου πραγματικού αριθμού με 4 bit ακέραιο μέρος και 4 bit κλασματικό μέρος
 - $01101100 \Rightarrow 0110.\textcolor{red}{1100} =$
 $0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + \textcolor{red}{1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4}} =$
 $4 + 2 + 0.50 + 0.25 = 6.75$
- Αναπαράσταση σταθερής υποδιαστολής προσημασμένου πραγματικού αριθμού σε αναπαράσταση συμπληρώματος ως προς δύο με 4 bit ακέραιο μέρος και 4 bit κλασματικό μέρος
 - $11011010 \Rightarrow \textcolor{blue}{1101}.\textcolor{red}{1010} =$
 $\textcolor{blue}{1 \times -2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4}} =$
 $-8 + 4 + 1 + 0.500 + 0.125 = -8 + 5.625 = -2.375$

Πραγματικό αριθμοί σταθερής υποδιαστολής

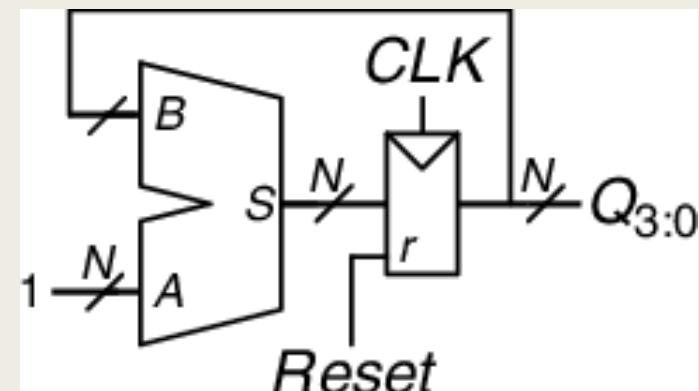
- Εύρεση συμπληρώματος ως προς 2 προσημασμένου πραγματικού αριθμού σταθερής υποδιαστολής σε αναπαράσταση συμπληρώματος ως προς δύο με 4 bit ακέραιο μέρος και 4 bit κλασματικό μέρος
 - συμπλήρωμα ως προς 2 του πραγματικού αριθμού **-0.625**

0000.1010	Διαδικό μέγεθος
1111.0101	Συμπλήρωμα ως προς ένα
+	Προσθέτουμε 1
	Συμπλήρωμα ως προς δύο

- Οι επεξεργαστές χειρίζονται τους πραγματικούς αριθμούς σταθερής υποδιαστολής όπως χειρίζονται τους ακεραίους
 - λόγω της απλότητας των πράξεων, οι πραγματικοί αριθμοί σταθερής υποδιαστολής χρησιμοποιούνται σε υλοποίηση αλγορίθμων στο υλικό, όπου απαιτούνται πραγματικοί αριθμοί
 - Ιδιαίτερα σε εφαρμογές **ψηφιακής επεξεργασίας σήματος (DSP)**
 - **απαιτείται μελέτη της επίδρασης της στρογγυλοποίησης στα αποτελέσματα**
- Από τη δεκαδική τιμή ενός ακέραιου αριθμού προκύπτει η τιμή του αντίστοιχου πραγματικού αριθμού σταθερής υποδιαστολής με κλασματικό μέρος N bit, που έχει την ίδια δυαδική αναπαράσταση:
 - εάν διαιρέσουμε την τιμή του ακέραιου αριθμού διά **2^N**

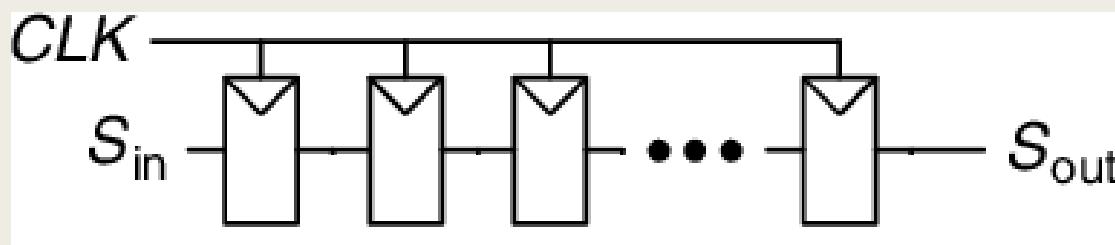
Μετρητής

- Ένας **δυαδικός μετρητής** των N bit είναι ένα σύγχρονο ακολουθιακό κύκλωμα αριθμητικής με εισόδους ρολογιού (CLK) και επαναφοράς στο 0 (*Reset*), και μια έξοδο Q των N bit
 - η είσοδος *Reset* δίνει στην έξοδο την αρχική τιμή 0
 - τη συνέχεια σαρώνει όλες τις 2^N πιθανές εξόδους σε δυαδική σειρά, αυξανόμενος βηματικά κατά 1 στην ανερχόμενη ακμή του CLK
 - Π.χ. 000, 001, 010, 011, 100, 101, 110, 111, 000, ...
- Οι δυαδικοί μετρητές χρησιμοποιούνται:
 - ως **Program Counter**
 - ο καταχωρητής που υποδεικνύει τη διεύθυνση της επόμενης προς εκτέλεση εντολής σε έναν επεξεργαστή
 - στην οθόνη ψηφιακού ρολογιού
 - στην καταμέτρηση συγκεκριμένων καταστάσεων και γεγονότων εντός του ψηφιακού συστήματος
- Υλοποιείται με αθροιστή των N bit με τη μία είσοδο σταθερή στο 1
 - που ονομάζεται αυξητής (*incrementer*)



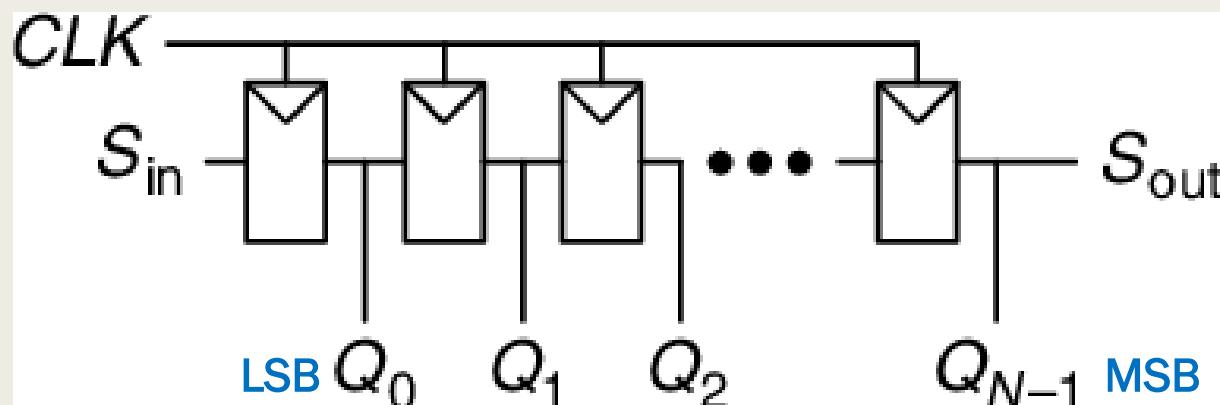
Καταχωρητής ολίσθησης

- Ένας **καταχωρητής ολίσθησης** των N bit είναι ένα σύγχρονο ακολουθιακό κύκλωμα προσωρινής αποθήκευσης σειριακών δεδομένων, που διαθέτει:
 - μία είσοδο ρολογιού (CLK)
 - μία σειριακή είσοδο S_{in}
 - σε κάθε ανερχόμενη ακμή του CLK ένα νέο bit εισέρχεται στον καταχωρητή ολίσθησης μέσω της σειριακής εισόδου S_{in} , ενώ τα ήδη υπάρχοντα bit εντός του καταχωρητή ολίσθησης ολισθαίνουν κατά μία θέση δεξιά και το τελευταίο bit χάνεται
 - μία σειριακή έξοδο S_{out}
 - το εκάστοτε τελευταίο bit του καταχωρητή ολίσθησης είναι διαθέσιμο στην σειριακή έξοδο S_{out}
- Κατασκευάζεται από N flip-flop συνδεδεμένα **σε σειρά** με κοινό CLK



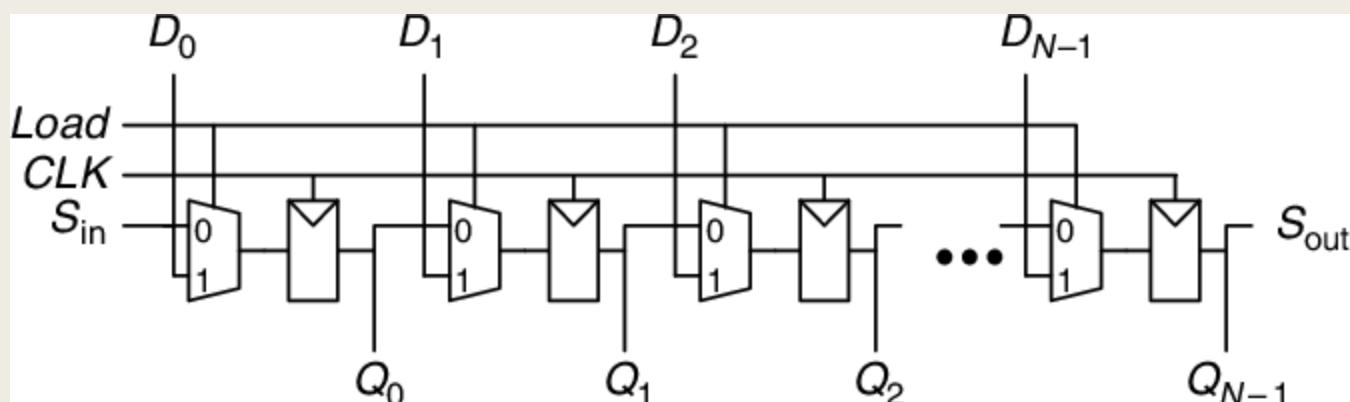
Καταχωρητής ολίσθησης

- Εάν ο καταχωρητής ολίσθησης διαθέτει και N παράλληλες εξόδους $Q_{N-1:0}$ ονομάζεται **μετατροπέας σειριακού σε παράλληλο** (serial-to-parallel converter)
 - Μετά από N κύκλους του CLK , οι τελευταίες N είσοδοι, που έχουν μεταδοθεί σειριακά μέσω της σειριακής εισόδου S_{in} , είναι διαθέσιμες παράλληλα στην έξοδο Q
 - Στο σχήμα θεωρείται ότι μεταδίδεται πρώτο το περισσότερο σημαντικό bit (MSB), ώστε να είναι διαθέσιμο στην έξοδο Q_{N-1} , και τελευταίο το λιγότερο σημαντικό bit (LSB), ώστε να είναι διαθέσιμο στην έξοδο Q_0 (μετάδοση big-endian)
 - Κάποιοι καταχωρητές ολίσθησης διαθέτουν και ένα σήμα Reset για να καθορίζουν την αρχική τιμή του καταχωρητή ολίσθησης



Καταχωρητής ολίσθησης

- Ένα σχετικό κύκλωμα είναι ο **μετατροπέας παράλληλου σε σειριακό** (parallel-to-serial converter) που φορτώνει N bit παράλληλα, και κατόπιν τα ολισθαίνε στην έξοδο, ένα τη φορά
- Μπορούμε να τροποποιήσουμε έναν καταχωρητή ολίσθησης ώστε να εκτελεί τις δύο μετατροπές και παράλληλου σε σειριακό και σειριακού σε παράλληλο, προσθέτοντας μια παράλληλη είσοδο $D_{N-1:0}$ και ένα σήμα ελέγχου **Load**
 - Όταν $Load = 1$, τα D flip-flop φορτώνονται παράλληλα από τις D εισόδους στην επόμενη ακμή του CLK (και το κύκλωμα λειτουργεί ως παράλληλος καταχωρητής των N bit)
 - Όταν $Load = 0$, το κύκλωμα λειτουργεί ως καταχωρητής ολίσθησης των N bit

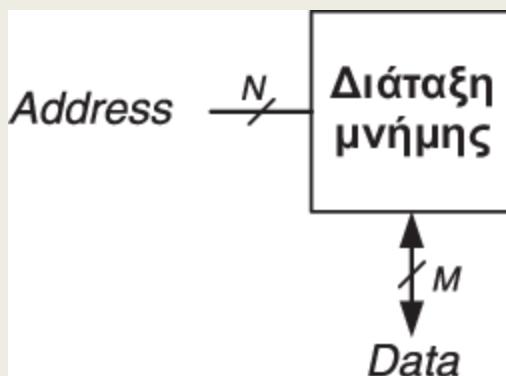


Διατάξεις μνήμης (memory arrays)

- Τα ψηφιακά συστήματα διαθέτουν **μνήμες** για να αποθηκεύουν δεδομένα
- Οι καταχωρητές που έχουν κατασκευαστεί με D flip-flop αποτελούν ένα είδος μνήμης στο οποίο αποθηκεύονται προσωρινά μικρές ποσότητες δεδομένων
- Οι **διατάξεις μνήμης** (memory arrays) μπορούν να αποθηκεύουν με αποδοτικό τρόπο μεγάλες ποσότητες δεδομένων
- Οι τρείς πιο συνηθισμένοι τύποι μνήμης είναι:
 - Δυναμική μνήμη τυχαίας προσπέλασης (DRAM)
 - Στατική μνήμη τυχαίας προσπέλασης (SRAM)
 - Μνήμη μόνο ανάγνωσης (ROM)

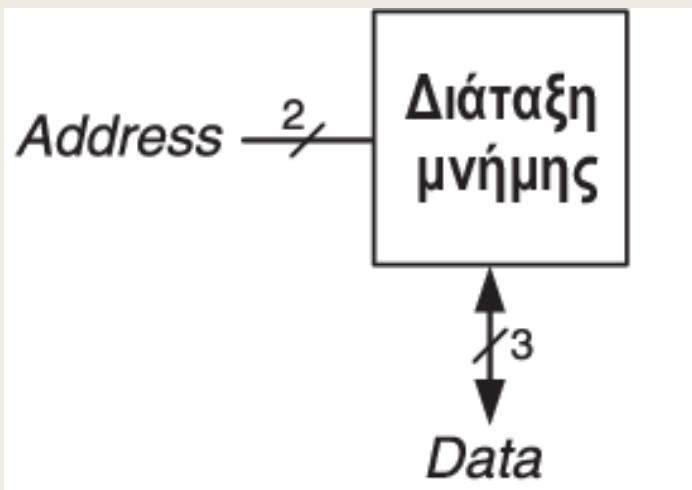
Διάταξης μνήμης

- Η μνήμη είναι οργανωμένη ως μια διδιάστατη **διάταξη κελιών μνήμης** (memory cells)
- Η μνήμη **διαβάζει** ή **εγγράφει** τα περιεχόμενα μίας από τις σειρές (rows) της διάταξης μνήμης, που ονομάζονται **δεδομένα** (data)
- Η σειρά της διάταξης μνήμης καθορίζεται από μια **διεύθυνση** (address)
- Μια διάταξη μνήμης με διευθύνσεις των N bit και δεδομένων των M bit έχει **2^N σειρές και M στήλες**
- Κάθε σειρά δεδομένων ονομάζεται **λέξη** (word)
 - η διάταξη μνήμης περιέχει 2^N λέξεις, με μέγεθος M bit η καθεμία



Παράδειγμα διάταξης μνήμης

- Στην εικόνα μπορείτε να δείτε μια διάταξη μνήμης με δύο bit διεύθυνσης και τρία bit δεδομένων
 - Τα δύο bit της διεύθυνσης ($N = 2$) καθορίζουν μία από τις τέσσερις σειρές ($2^N = 4$) της διάταξης μνήμης
 - Σε κάθε σειρά αποθηκεύεται μία λέξη δεδομένων που έχει μέγεθος $M = 3$ bit
 - Παράδειγμα: η λέξη των 3 bit που αποθηκεύεται στη διεύθυνση 10₂ είναι η 100₂
- Μέγεθος μιας διάταξης μνήμης (array size): **βάθος × πλάτος** ($= 4 \times 3 = 12$)
 - **Βάθος (depth)**: το πλήθος των σειρών (των λέξεων δεδομένων)
 - **Πλάτος (width)**: το πλήθος των στηλών (μέγεθος της λέξης δεδομένων)

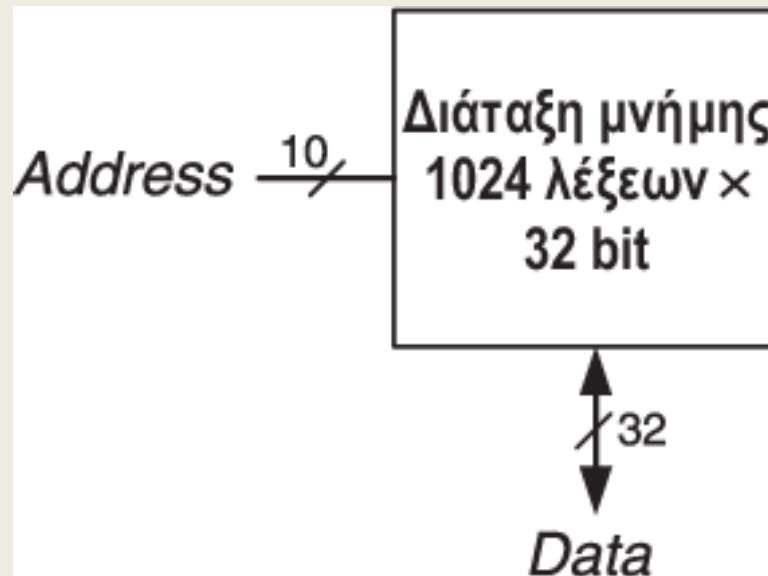


	Address	Data
11		0 1 0
10		1 0 0
01		1 1 0
00		0 1 1

A double-headed vertical arrow on the right side of the table is labeled "βάθος" (depth), indicating the number of rows (4). A double-headed horizontal arrow at the bottom is labeled "πλάτος" (width), indicating the number of columns (3).

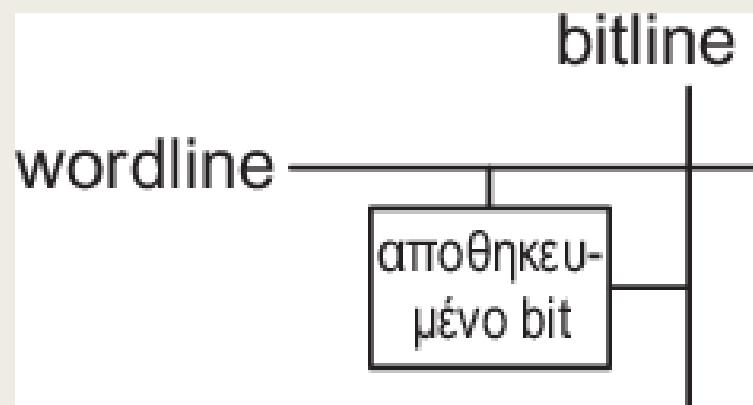
Παράδειγμα διάταξης μνήμης

- Στην εικόνα μπορείτε να δείτε το σύμβολο για μια διάταξη μνήμης 1024 λέξεων \times 32 bit
- Το συνολικό μέγεθος αυτής της διάταξης μνήμης είναι ίσο με 32 kilobit (Kb)



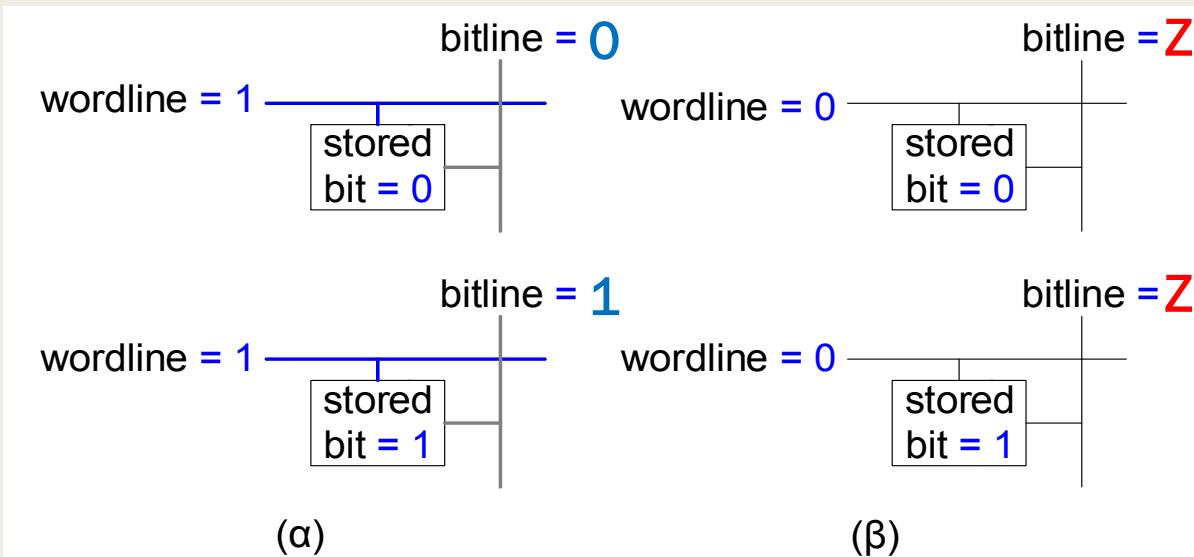
Κελιά μνήμης

- Οι διατάξεις μνήμης κατασκευάζονται ως μια διδιάστατη διάταξη **κελιών μνήμης** (memory cells)
 - σε καθένα από τα οποία αποθηκεύεται **1 bit δεδομένων**
- Στην εικόνα φαίνεται ότι καθένα κελί μνήμης του 1 bit συνδέεται:
 - οριζόντια με μία γραμμή **wordline** που προσδιορίζει μία λέξη δεδομένων, και
 - κάθετα με μία γραμμή **bitline** για τη μεταφορά δεδομένων από και προς τη μνήμη



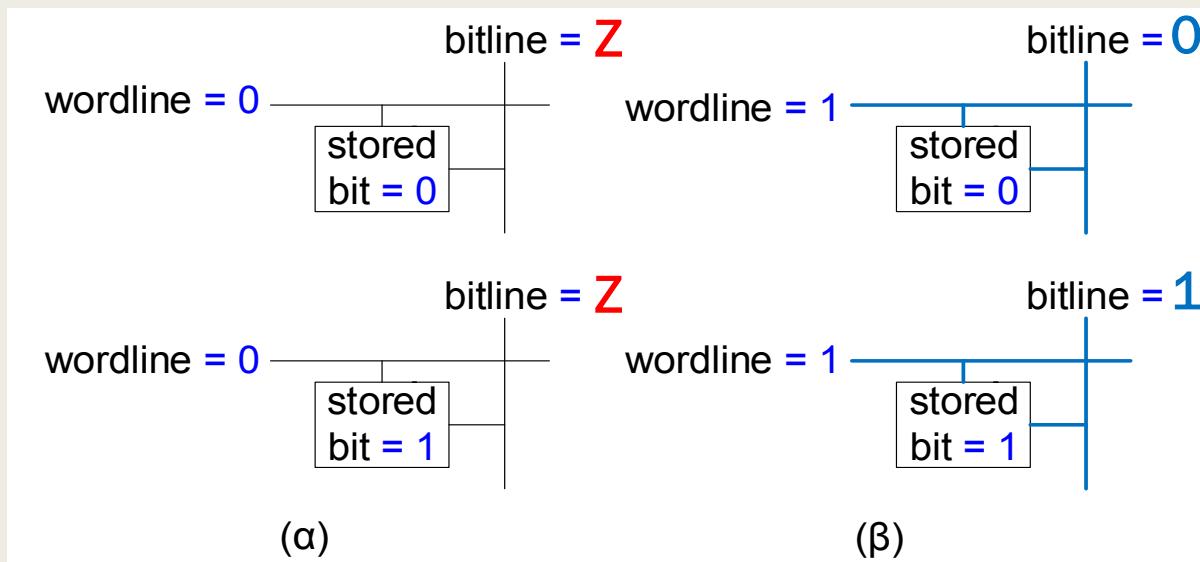
Κελιά μνήμης

- Για κάθε συνδυασμό τιμών των bit της διεύθυνσης (Address), η μνήμη ενεργοποιεί μόνο μία γραμμή **wordline** (δηλαδή τη θέτει στην τιμή HIGH), η οποία με τη σειρά της ενεργοποιεί μόνο τα κελιά μνήμης της συγκεκριμένης σειράς της διάταξης μνήμης
- (α) Όταν η γραμμή wordline έχει την τιμή HIGH, το αποθηκευμένο bit μεταφέρεται προς ή από τη γραμμή bitline
- (β) Όταν η γραμμή wordline έχει την τιμή LOW, η γραμμή bitline αποσυνδέεται από το κελί bit και παραμένει μετέωρη (Z)



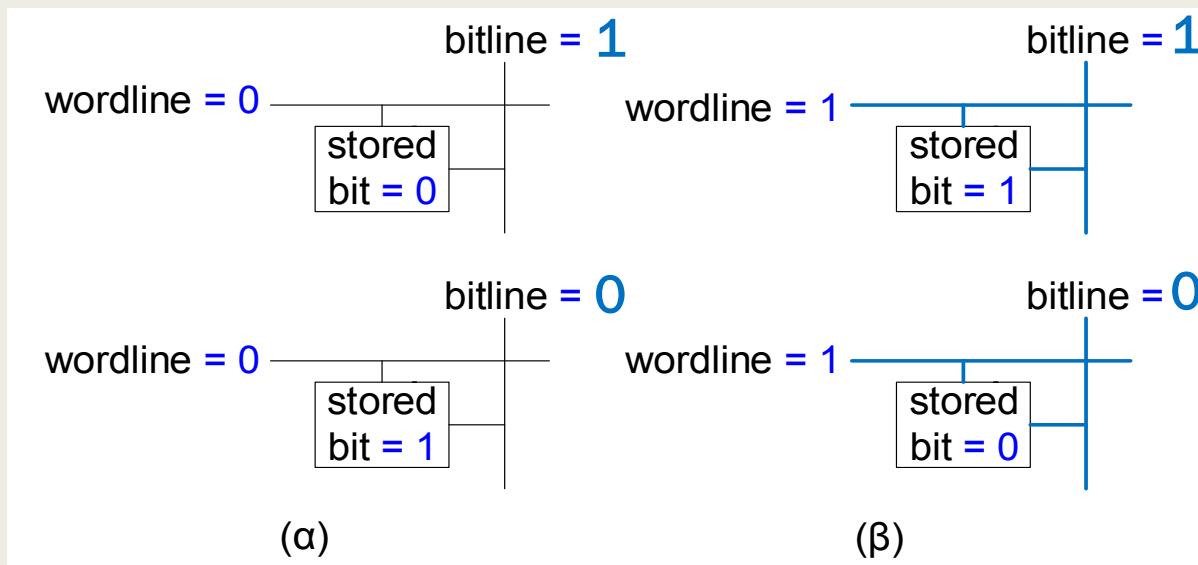
Κελιά μνήμης: Ανάγνωση

- (α) Η γραμμή bitline αφήνεται **αρχικά μετέωρη** (Z)
- (β) Η γραμμή **wordline ενεργοποιείται** (γίνεται HIGH)
 - Αυτό επιτρέπει στην αποθηκευμένη τιμή να οδηγήσει τη γραμμή bitline στο 0 ή στο 1, αντίστοιχα



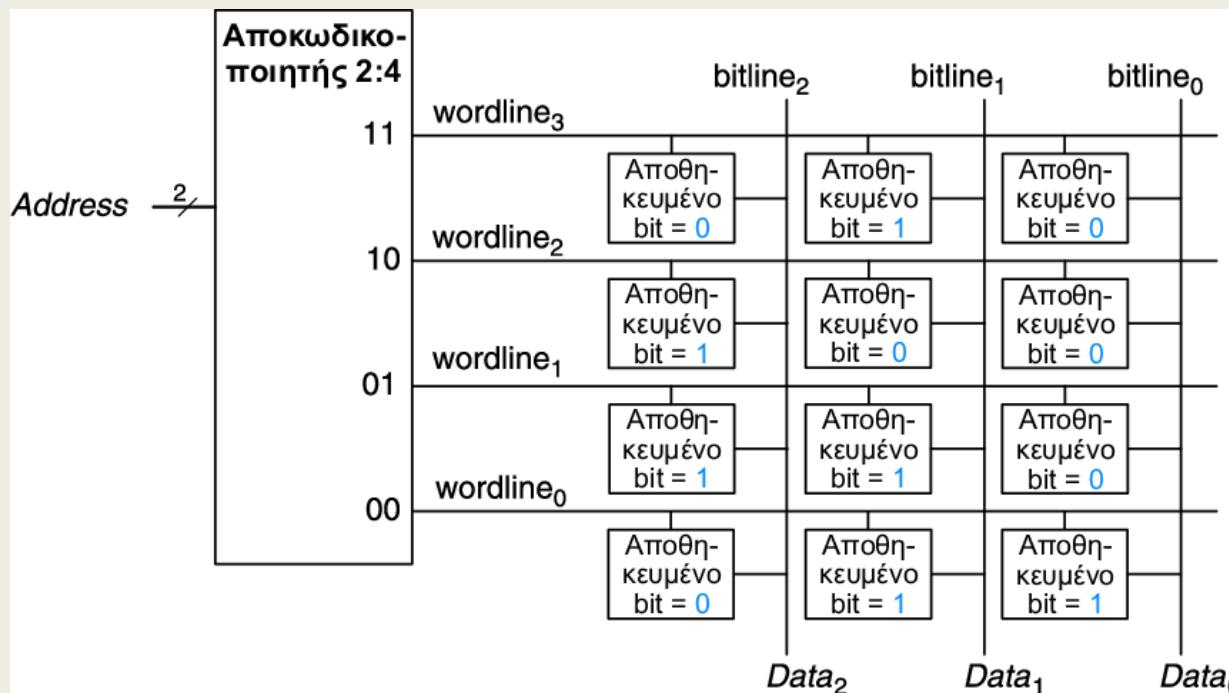
Κελιά μνήμης: Εγγραφή

- (α) Η γραμμή bitline **οδηγείται ισχυρά** προς την επιθυμητή τιμή (0 ή 1), ενώ η γραμμή wordline είναι απενεργοποιημένη (είναι LOW)
- (β) Στη συνέχεια η γραμμή **wordline ενεργοποιείται** (γίνεται HIGH), συνδέοντας έτσι τη γραμμή bitline με το κελί μνήμης
 - Αυτό επιτρέπει στην ισχυρά οδηγούμενη γραμμή bitline να υπερισχύει των περιεχομένων του κελιού μνήμης, εγγράφοντας την επιθυμητή τιμή στο συγκεκριμένο κελί μνήμης



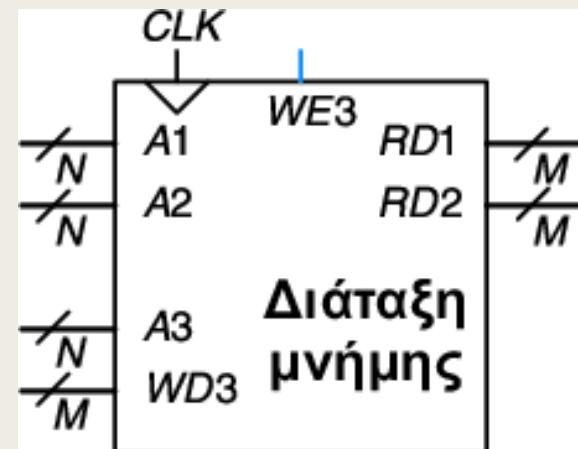
Οργάνωση μνήμης

- Η μνήμη απαρτίζεται από έναν **αποκωδικοποιητή N σε 2^N** και τα **$2^N \times M$ κελιά μνήμης**
 - ο αποκωδικοποιητής, ανάλογα με τη διεύθυνση που λαμβάνει στην είσοδό του, ενεργοποιεί **μόνο μία** από τις 2^N γραμμές **wordline**
 - η ενεργοποιημένη γραμμή wordline, με τη σειρά της, ενεργοποιεί μόνο τα **M κελιά μνήμης** της συγκεκριμένης σειράς της διάταξης μνήμης (**λέξης δεδομένων**)
- Παράδειγμα: Διάταξη μνήμης 4 λέξεων \times 3 bit (N =2, M=3)



Θύρες μνήμης

- Όλες οι μνήμες διαθέτουν μία ή περισσότερες θύρες (ports)
- Κάθε θύρα παρέχει πρόσβαση για ανάγνωση ή/και εγγραφή από/σε μία διεύθυνση μνήμης (στη λέξη δεδομένων που αντιστοιχεί σε μία διεύθυνση μνήμης)
- Οι **πολύθυρες** μνήμες (multiported memories) μπορούν να προσπελάσουν πολλές διευθύνσεις μνήμης **ταυτόχρονα**
- Στην εικόνα φαίνεται μια τρίθυρη μνήμη με δύο θύρες ανάγνωσης και μία θύρα εγγραφής
 - Η θύρα 1 διαβάζει ασύγχρονα τα δεδομένα από τη διεύθυνση A1 και τα μεταφέρει στην έξοδο RD1
 - Η θύρα 2 διαβάζει ασύγχρονα τα δεδομένα από τη διεύθυνση A2 και τα μεταφέρει στην έξοδο RD2
 - Η θύρα 3 εγγράφει σύγχρονα (κατά την ανερχόμενη ακμή του ρολογιού) τα δεδομένα από την είσοδο WD3 στη διεύθυνση A3, εάν το σήμα ελέγχου WE3 έχει την τιμή 1



Τύποι μνήμης

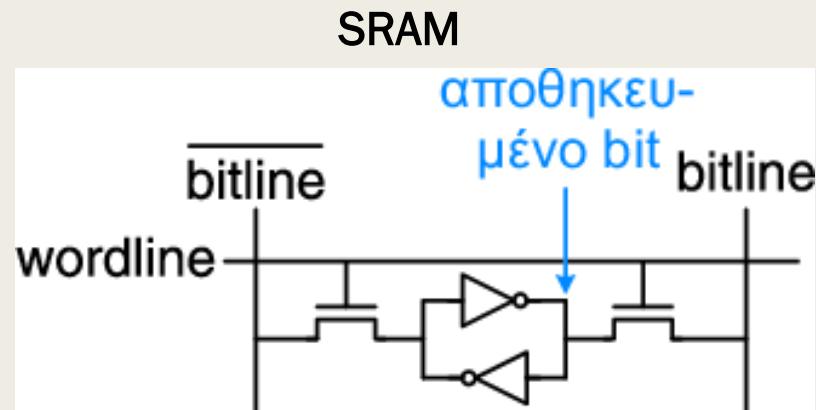
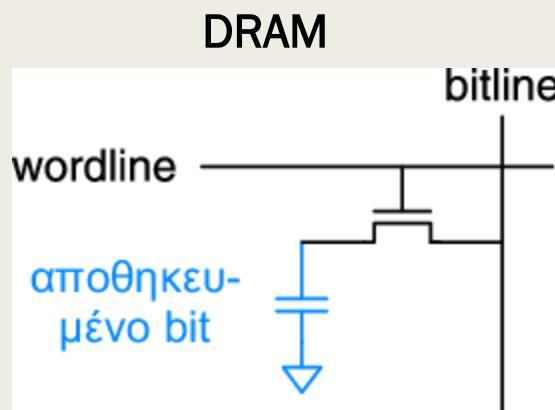
- Οι διατάξεις μνήμης καθορίζονται από το **μέγεθός** τους (βάθος × πλάτος), καθώς και από το **πλήθος και τον τύπο των θυρών**
- Όλες οι διατάξεις μνήμης αποθηκεύουν τα δεδομένα με τη μορφή μιας σειράς κελιών μνήμης του 1 bit, αλλά διαφέρουν ως προς τον **τρόπο αποθήκευσης των bit** στα κελιά μνήμης
- Οι μνήμες **ταξινομούνται** με βάση τον **τρόπο αποθήκευσης των bit** σε δύο μεγάλες κατηγορίες
 - **τις μνήμες τυχαίας προσπέλασης** (*random access memory, RAM*)
 - **τις μνήμες μόνο για ανάγνωση** (*read only memory, ROM*)
- Η μνήμη RAM είναι **πτητική** (*volatile*), το οποίο σημαίνει ότι χάνει τα δεδομένα της όταν απενεργοποιείται η τροφοδοσία ρεύματος
- Η μνήμη ROM είναι **μη πτητική** (*nonvolatile*), που σημαίνει ότι διατηρεί τα δεδομένα της για αόριστο χρονικό διάστημα, ακόμα και χωρίς να τροφοδοτείται με ρεύμα

Τύποι μνήμης

- Η **μνήμη RAM** ονομάζεται μνήμη τυχαίας προσπέλασης επειδή οποιαδήποτε λέξη δεδομένων **προσπελάζεται με την ίδια καθυστέρηση**
 - Απεναντίας, μια μνήμη **σειριακής προσπέλασης**, όπως το κασετόφωνο, προσπελάζει τα κοντινά δεδομένα πιο γρήγορα από ό,τι τα μακρινά δεδομένα
- Η **μνήμη ROM** ονομάζεται μνήμη μόνο για ανάγνωση επειδή, από ιστορική άποψη, αρχικά τα δεδομένα μπορούσαν μόνο να διαβαστούν από αυτήν και όχι να εγγραφούν

Τύποι μνήμης RAM

- Δύο κύριοι τύποι της μνήμης RAM
 - Δυναμική μνήμη RAM (*Dynamic RAM, DRAM*)
 - Στατική μνήμη RAM (*Static RAM, SRAM*)
- Διαφέρουν στο πως αποθηκεύουν τα δεδομένα
 - Στη DRAM τα δεδομένα αποθηκεύονται ως **φορτίο σε έναν πυκνωτή**
 - Απαιτεί **επανεγγραφή** μετά την ανάγνωση και περιοδική **αναζωογόνηση** για να διατηρηθεί το φορτίο του πυκνωτή
 - Στη SRAM αποθηκεύονται με χρήση δύο **διασυζευγμένων αντιστροφέων**



Επιφάνεια υλοποίησης και καθυστέρηση

- Τα **D Flip-flop**, οι **μνήμες SRAM** και οι **μνήμες DRAM** είναι μεν πτητικές μνήμες, αλλά διαθέτουν διαφορετικά χαρακτηριστικά όσον αφορά στην επιφάνεια υλοποίησης και στην καθυστέρηση
- Το bit δεδομένων που αποθηκεύεται σε ένα D Flip-flop είναι **διαθέσιμο άμεσα στην έξοδο** του κυκλώματος (μικρός λανθάνων χρόνος), όμως η υλοποίηση του D Flip-flop απαιτεί περίπου 20 τρανζίστορ
 - Όσα περισσότερα τρανζίστορ εχει μία διάταξη μνήμης, τόσο περισσότερη επιφάνεια υλοποίησης, ισχύ και κόστος απαιτεί
- Η μνήμη DRAM έχει **μεγάλο λανθάνοντα χρόνο** (latency) γιατί πρέπει να περιμένει μέχρι να μετακινηθεί φορτίο (σχετικά) αργά από τον πυκνωτή στη γραμμή bitline

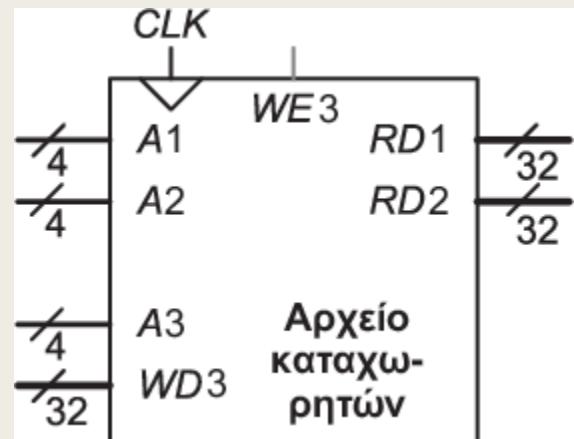
Τύπος μνήμης	Τρανζίστορ ανά κελί μνήμης	Λανθάνων χρόνος
D Flip-flop	~20	μικρός
SRAM	6	μέτριος
DRAM	1	μεγάλος

Επιφάνεια υλοποίησης και καθυστέρηση

- Η μνήμη DRAM έχει **μικρότερη διεκπεραιωτική ικανότητα** (throughput) σε σύγκριση με τη μνήμη SRAM, λόγω της ανάγκης για **επανεγγραφή** μετά την ανάγνωση και περιοδική **αναζωγόνηση**
- Για την αντιμετώπιση αυτού του προβλήματος έχουν αναπτυχθεί τεχνολογίες DRAM, όπως:
 - η **σύγχρονη DRAM** (synchronous DRAM, SDRAM) και
 - η **σύγχρονη DRAM διπλού ρυθμού (μεταφοράς) δεδομένων** (double data rate synchronous DRAM, DDR SDRAM).
- Η μνήμη SDRAM προσπελαύνεται σύγχρονα στην ανερχόμενη ακμή του CLK
- Η μνήμη DDR SDRAM, προσπελαύνεται σύγχρονα και **στην ανερχόμενη και στην κατερχόμενη ακμή του CLK**, διπλασιάζοντας έτσι τη διεκπεραιωτική ικανότητα για δεδομένη συχνότητα του ρολογιού
 - Η μνήμη DDR προτυποποιήθηκε για πρώτη φορά το 2000 και «έτρεχε» στα 100 έως 200 MHz. Τα μεταγενέστερα πρότυπα (DDR2, DDR3 και DDR4) αύξησαν την ταχύτητα του ρολογιού, με τη σχετική τιμή να ξεπερνάει το 1GHz το 2015

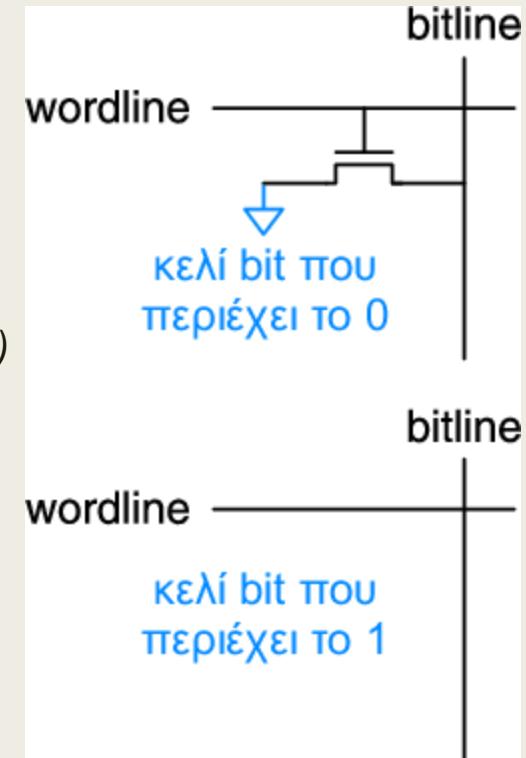
Αρχεία καταχωρητών

- Τα ψηφιακά συστήματα συχνά χρησιμοποιούν αρκετούς καταχωρητές για την αποθήκευση προσωρινών μεταβλητών που ονομάζεται **αρχείο καταχωρητών** (register file)
 - συνήθως κατασκευάζεται ως μια μικρή, πολύθυρη διάταξη μνήμης SRAM (πιο συνεπτυγμένη από μια διάταξη με *D Flip-flop*)
- Στην εικόνα φαίνεται ένα τρίθυρο αρχείο καταχωρητών, με **16 καταχωρητές × 32 bit**, το οποίο υλοποιείται με μία **τρίθυρη μνήμη SRAM**
 - Το αρχείο καταχωρητών διαθέτει δύο θύρες ανάγνωσης ($A1/RD1$ και $A2/RD2$) και μία θύρα εγγραφής ($A3/WD3$)
 - Καθεμία από τις διευθύνσεις μήκους 4 bit ($A1$, $A2$ και $A3$) μπορεί να προσπελάσει όλους τους $2^4 = 16$ καταχωρητές
 - Οι θύρες 1 και 2 διαβάζουν ασύγχρονα τα δεδομένα από τους καταχωρητές $A1$ και $A2$ και τα μεταφέρουν στις εξόδους $RD1$ και $RD2$, αντίστοιχα
 - Η θύρα 3 εγγράφει σύγχρονα (κατά την ανερχόμενη ακμή του ρολογιού) τα δεδομένα από την είσοδο $WD3$ στον καταχωρητή $A3$, εάν το σήμα ελέγχου $WE3$ έχει την τιμή 1



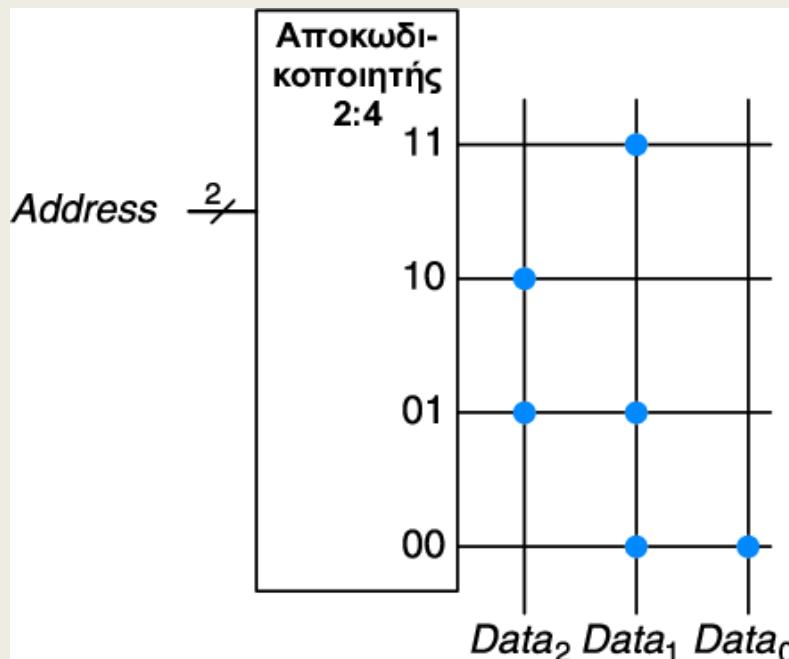
Μνήμη ROM

- Στη μνήμη ROM, η αποθήκευση δεδομένων βασίζεται στην **σταθερή παρουσία ή απουσία ενός τρανζίστορ**
- Για να **διαβαστεί** το κελί μνήμης, η γραμμή **bitline** **οδηγείται ασθενώς** στην τιμή HIGH
- Κατόπιν η γραμμή **wordline** ενεργοποιείται:
 - Αν **υπάρχει** το τρανζίστορ, οδηγεί τη γραμμή **bitline** στην τιμή **LOW** (το **κελί bit αποθηκεύει το 0**)
 - Αν **δεν υπάρχει** το τρανζίστορ, η γραμμή **bitline** **παραμένει στο HIGH** (το **κελί bit αποθηκεύει το 1**)
- Η μνήμη ROM είναι **μη πτητική** (nonvolatile), που σημαίνει ότι διατηρεί τα δεδομένα της για αόριστο χρονικό διάστημα, ακόμα και χωρίς να τροφοδοτείται με ρεύμα



Μνήμη ROM: Υλοποίηση συνδυαστικής λογικής

- Για τα περιεχόμενα μιας μνήμης ROM μπορεί να χρησιμοποιηθεί ο **συμβολισμός με τελείες** (dot notation)
 - Μια τελεία στην τομή μιας σειράς και μιας στήλης υποδεικνύει ότι το bit δεδομένων έχει την τιμή 1, αλλιώς έχει την τιμή 0
- Παράδειγμα: Διάταξη μνήμης ROM 4 λέξεων \times 3 bit ($N = 2$, $M = 3$) που υλοποιεί τις συναρτήσεις $Data_0$, $Data_1$ και $Data_2$

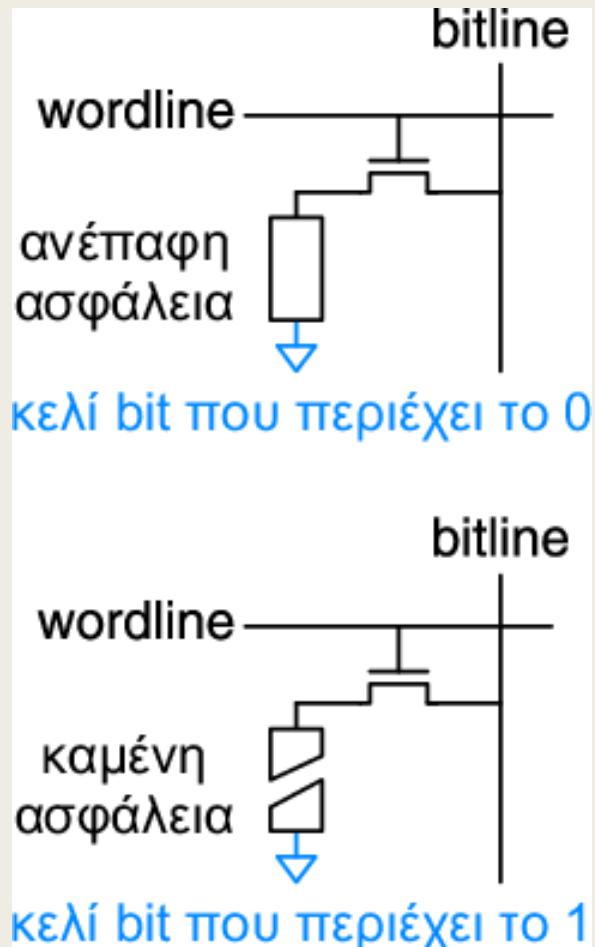


Address	Data
11	0 1 0
10	1 0 0
01	1 1 0
00	0 1 1

$$Data_2 = A_1 \oplus A_0 \quad Data_1 = \overline{A_1} + A_0 \quad Data_0 = \overline{A_1} \overline{A_0}$$

Προγραμματιζόμενη μνήμη ROM με ασφάλειες

- Στη μνήμη PROM (programmable ROM) με ασφάλειες, η αποθήκευση δεδομένων βασίζεται στην **παρουσία μίας ασφάλειας** μεταξύ του τρανζίστορ και της γείωσης
- Ο χρήστης προγραμματίζει τη μνήμη PROM εφαρμόζοντας μια υψηλή τάση που καίει επιλεκτικά κάποιες από τις ασφάλειες.
 - Αν η ασφάλεια παραμείνει **ανέπαφη**, το τρανζίστορ συνδέεται με τη γείωση (GND) και το κελί bit **αποθηκεύει την τιμή 0**
 - Αν η ασφάλεια **καεί**, το τρανζίστορ αποσυνδέεται από τη γείωση και το κελί bit **αποθηκεύει την τιμή 1**
- Η μνήμη PROM με ασφάλειες **προγραμματίζεται μόνο μία φορά** (one-time programmable, OTP)

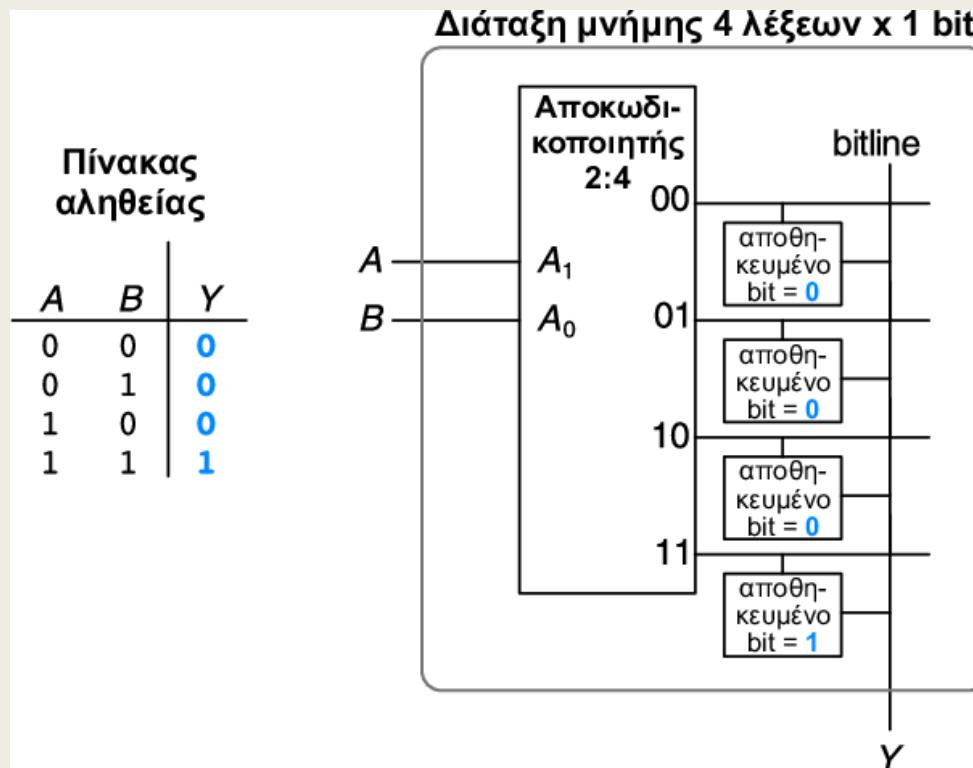


Μνήμες EPROM, EEPROM και Flash

- Οι επαναπρογραμματιζόμενες μνήμες PROM διαθέτουν αντιστρέψιμο μηχανισμό για τη σύνδεση ή την αποσύνδεση του τρανζίστορ με/από τη γείωση (GND)
- Οι **απαλείψιμες μνήμες PROM** (Erasable PROM, EPROM) αντικαθιστούν το τρανζίστορ και την ασφάλεια με ένα **τρανζίστορ μετέωρης πύλης**
 - Με κατάλληλα υψηλές τάσεις, ηλεκτρόνια διοχετεύονται επιλεκτικά στη μετέωρη πύλη και το τρανζίστορ άγει (προγραμματισμός μνήμης PROM)
 - Με έκθεση σε έντονο υπεριώδες φως για μισή ώρα, τα ηλεκτρόνια απομακρύνονται από τη μετέωρη πύλη και το τρανζίστορ πλέον δεν άγει (απαλειφή μνήμης PROM)
- Οι **ηλεκτρικά απαλείψιμες μνήμες PROM** (Electrically Erasable PROM, EEPROM) και η **μνήμη Flash** χρησιμοποιούν ηλεκτρονικά κυκλώματα για τον προγραμματισμό και την απαλοιφή της μνήμης PROM
 - Δεν είναι απαραίτητο το υπεριώδες φως
 - Στις μνήμες EEPROM τα περιεχόμενα κάθε κελιού bit της μνήμης μπορούν να απαλειφθούν ξεχωριστά
 - Στις μνήμες Flash απαλείφονται ταυτόχρονα μεγαλύτερες ομάδες από bit
 - Είναι φθηνότερη, επειδή απαιτούνται λιγότερα κυκλώματα απαλοιφής

Πίνακες αναζήτησης (LUT)

- Οι διατάξεις μνήμης που χρησιμοποιούνται για την υλοποίηση λογικής ονομάζονται **πίνακες αναζήτησης** (lookup tables, LUT)
 - Υλοποιείται ο πίνακας αλήθειας
- Παράδειγμα: Διάταξη μνήμης 4 λέξεων \times 1 bit η οποία χρησιμοποιείται ως πίνακας αναζήτησης για την υλοποίηση της συνάρτησης $Y = AB$
 - Χρησιμοποιούνται στις διατάξεις *FPGA*

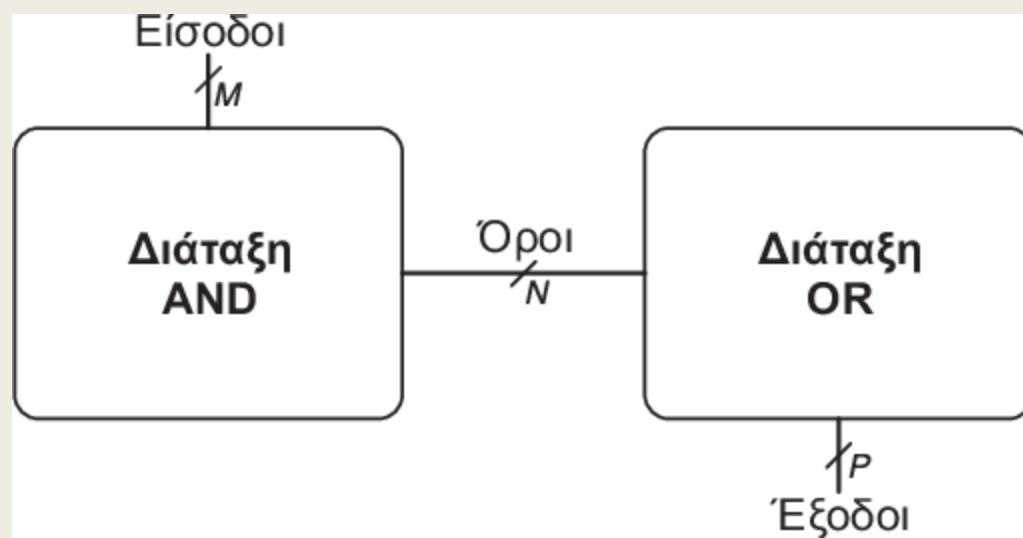


Διατάξεις λογικής (Logic arrays)

- **Προγραμματιζόμενες διατάξεις λογικής**
(programmable logic arrays, PLA)
 - υλοποιούν συνδυαστική λογική δύο επιπέδων σε μορφή αθροίσματος γινομένων (sum-of-product, SOP)
 - περιλαμβάνουν μια διάταξη AND ακολουθούμενη από μια διάταξη OR
- **Προγραμματιζόμενες από τον χρήστη διατάξεις πυλών**
(field programmable gate arrays, FPGA)
 - υλοποιούν συνδυαστική και ακολουθιακή λογική με τη χρήση **πινάκων αναζήτησης** (lookup tables, LUT) και **στοιχείων αποθήκευσης**
 - υλοποιούν πολυεπίπεδες λογικές συναρτήσεις, σε αντίθεση με τις διατάξεις PLA οι οποίες περιορίζονται στη λογική δύο επιπέδων
 - διαθέτουν **μνήμη διαμόρφωσης** (configuration memory), όπου αποθηκεύεται η πληροφορία διαμόρφωσης που υλοποιείται σε διάφορες τεχνολογίες διατάξεων μνήμης (SRAM, OTP-PROM, Flash)
 - Υλοποιούν σχετικά **μεγάλο πλήθος λογικών πυλών** (από 1.000 μέχρι πάνω από 3.000.000 πύλες)
 - Χρησιμοποιούνται στην υλοποίηση **ενσωματωμένων συστημάτων σε ένα τσιπ** (System on Chip, SoC)

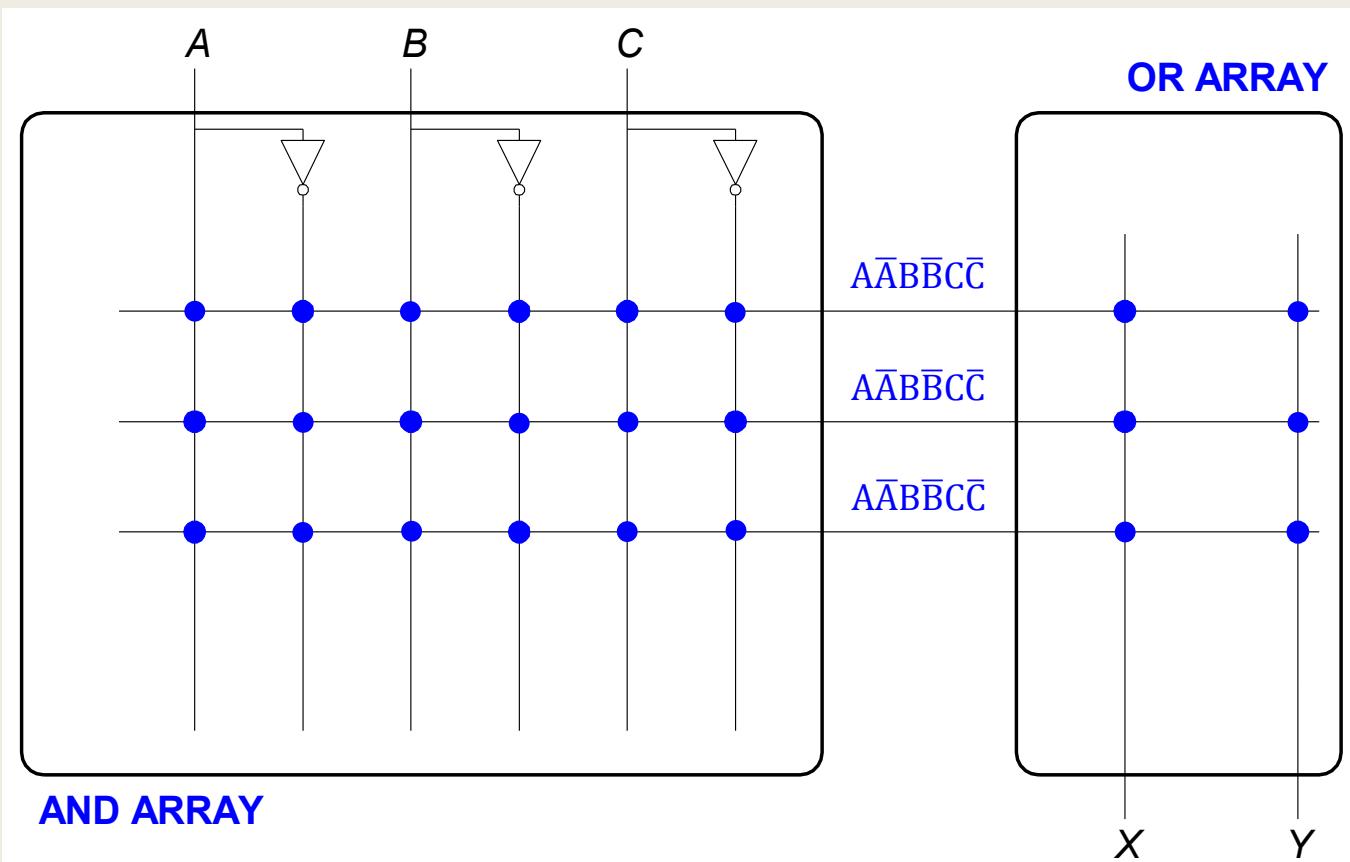
Προγραμματιζόμενες διατάξεις λογικής (PLA)

- Περιλαμβάνουν μια διάταξη AND ακολουθούμενη από μια διάταξη OR των οποίων οι είσοδοι προγραμματίζονται
 - Οι είσοδοι (σε κανονική και συμπληρωματική μορφή) οδηγούν μια διάταξη AND, που παράγει όρους (γινόμενα λεκτικών), που με τη σειρά τους συνδυάζονται σε αθροίσματα όρων μέσω της διάταξης OR, ώστε να σχηματισθούν οι έξοδοι που υλοποιούν συνδυαστική λογική δύο επιπέδων σε μορφή αθροίσματος γινομένων
 - Μια διάταξη PLA με διαστάσεις $M \times N \times P$ bit διαθέτει M εισόδους, N όρους και P εξόδους



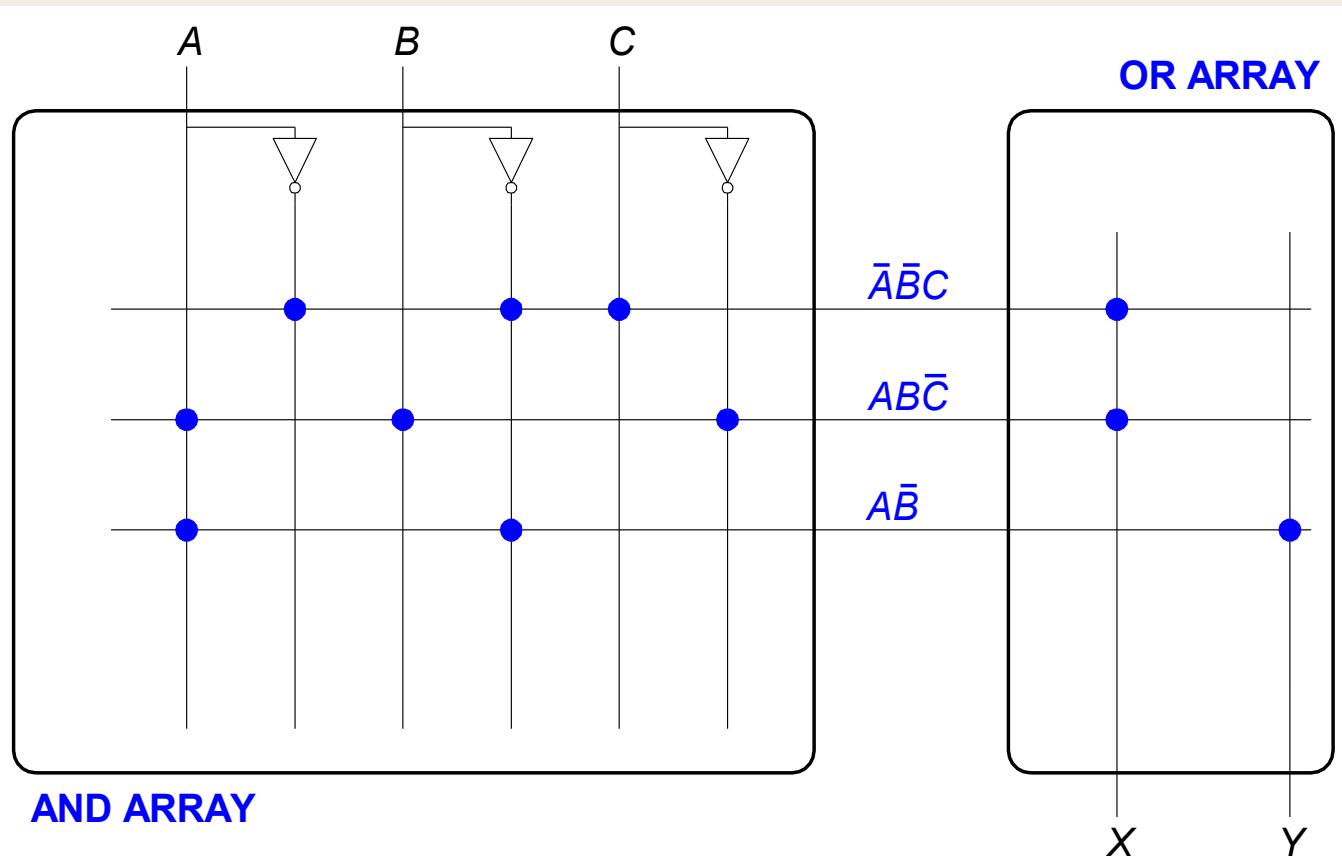
Προγραμματιζόμενες διατάξεις λογικής (PLA)

- Παράδειγμα: Διάταξη PLA με διαστάσεις $3 \times 2 \times 3$ bit που διαθέτει $M=3$ εισόδους, $N=3$ όρους και $P=2$ εξόδους
 - Πριν τον προγραμματισμό υπάρχουν συνδέσεις παντού, ώστε $X = Y = 0$
 - Για τις συνδέσεις μπορεί να χρησιμοποιηθεί ο **συμβολισμός με τελείες**



Προγραμματιζόμενες διατάξεις λογικής (PLA)

- Παράδειγμα: Διάταξη PLA με διαστάσεις $3 \times 2 \times 3$ bit που διαθέτει M=3 εισόδους, N=3 όρους και P=2 εξόδους
 - Με τον προγραμματισμό αφαιρούνται οι ανεπιθύμητες συνδέσεις, ώστε να υλοποιηθούν οι λογικές συναρτήσεις $X = \bar{A}\bar{B}C + A\bar{B}\bar{C}$ και $Y = A\bar{B}$
 - Για τις συνδέσεις μπορεί να χρησιμοποιηθεί ο **συμβολισμός με τελείες**

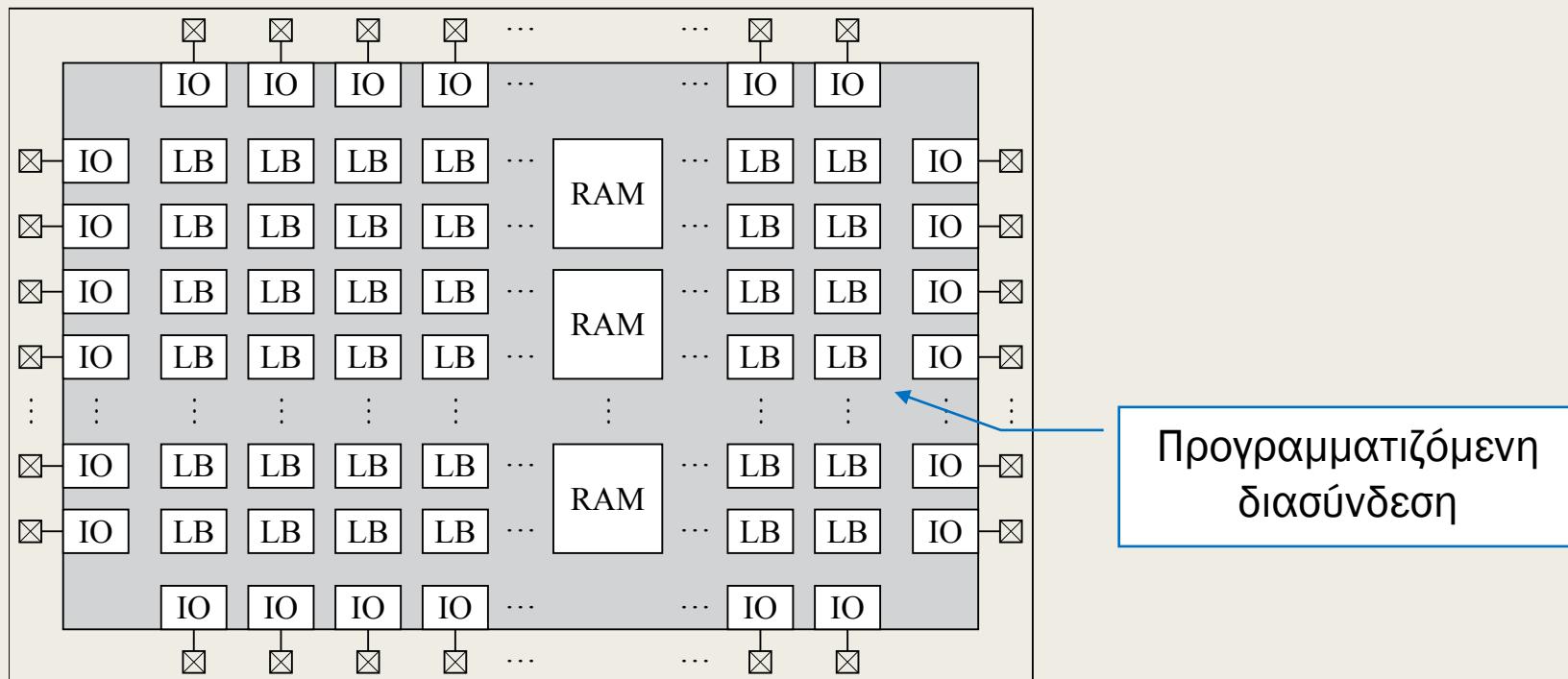


Προγραμματιζόμενες από τον χρήστη διατάξεις πυλών (FPGA)

- Παρέχουν δυνατότητα σχεδίασης **VLSI κυκλωμάτων χαμηλού κόστους** στο πεδίο, με τη χρήση σχετικά φθηνών εργαλείων λογισμικού (CAD)
 - κόστος ανεκτό από μικρές εταιρείες που δραστηριοποιούνται στην ανάπτυξη επιταχυντών υλικού και γενικότερα πυρήνων IP
- Είναι **επαναπρογραμματίσιμες** και **επαναδιατάξιμες** (ολικώς ή μερικώς) ακόμα και κατά τη διάρκεια της κανονικής λειτουργίας
- Παρέχουν **μεγάλη ευελιξία** στη σχεδίαση ψηφιακών συστημάτων.
- Διαθέτουν **ενσωματωμένες μνήμες, πολλαπλασιαστές, ειδικές μονάδες** για ψηφιακή επεξεργασία σήματος, εισόδους/εξόδους υψηλών ταχυτήτων, μετατροπείς δεδομένων (όπως ADC, DAC), ακόμα και **πυρήνες επεξεργαστών** σε κάποιες περιπτώσεις
- Η γενικότερη τεχνολογική εξέλιξη σε θέματα κόστους, αποδόσεων, κατανάλωσης ισχύος και αξιοπιστίας έχει σαν αποτέλεσμα οι σύγχρονες διατάξεις FPGA να χρησιμοποιούνται ευρέως σε **εμπορικές, βιομηχανικές, αμυντικές και διαστημικές εφαρμογές**

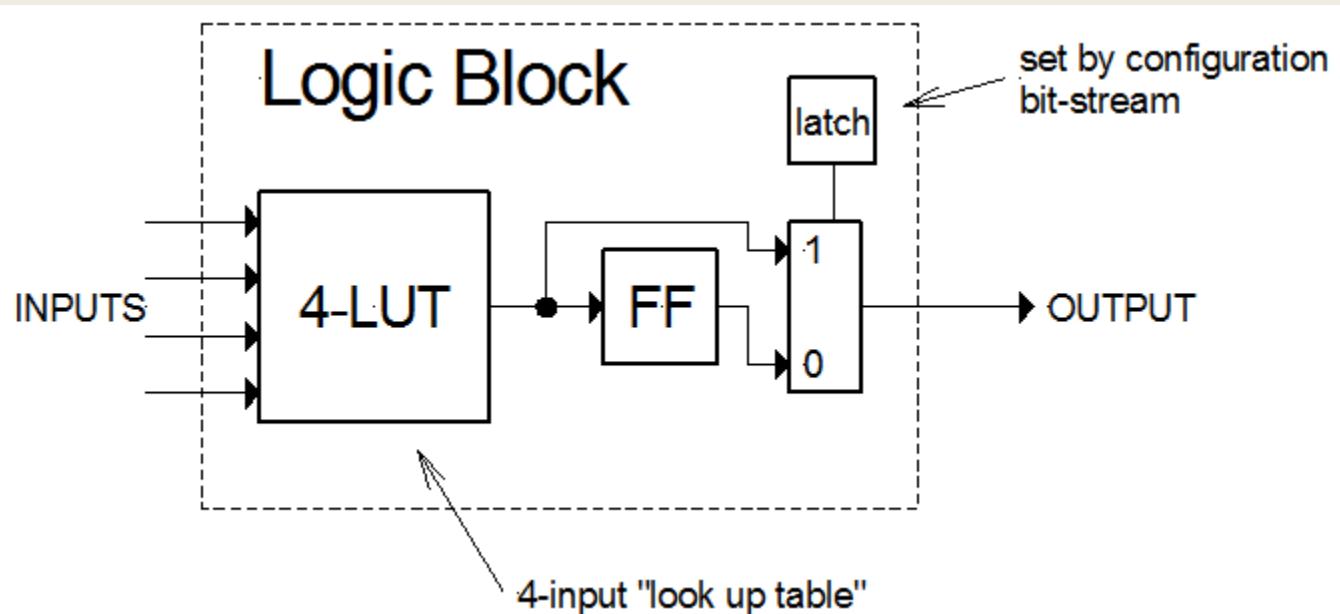
Field Programmable Gate Arrays (FPGAs)

- Προγραμματιζόμενες από τον χρήστη διατάξεις πυλών:
 - Μικρότερα *logic blocks* (*LB*), ενσωματωμένες *SRAM*
 - Ισοδύναμα με χιλιάδες ή εκατομμύρια πύλες



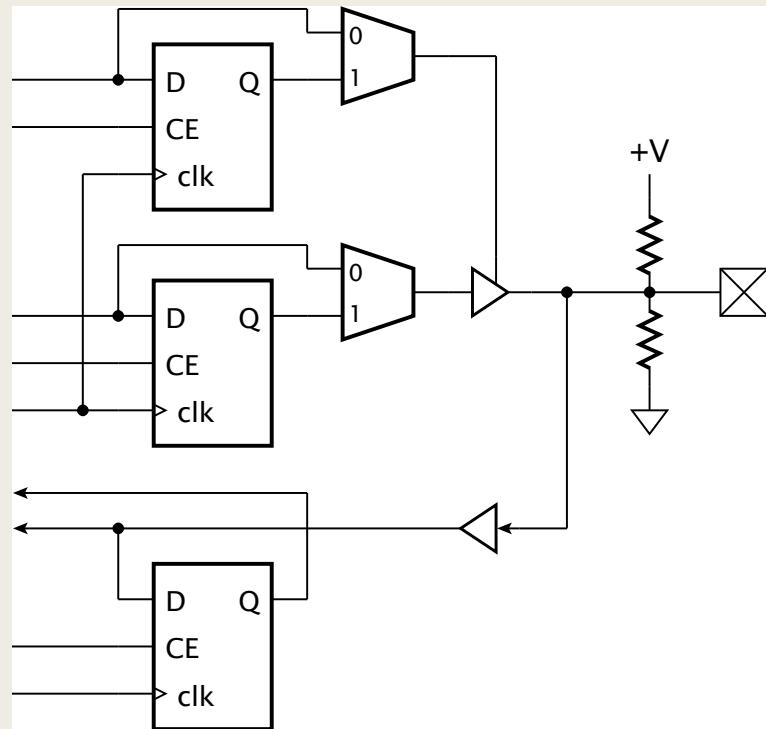
Ένα «Ιδανικό» FPGA logic block

- **Πίνακες αναζήτησης (Look-Up Table, LUT)** των 4 εισόδων
 - είναι διατάξις μνήμης που υλοποιούν τον πίνακα αλήθειας συνδυαστικής λογικής
 - οι νέες τεχνολογίες FPGA έχουν *LUT* των 6 εισόδων
 - ο χρονισμός είναι ανεξάρτητος της συνάρτησης
- **Flip-flop**
 - μπορεί να αποθηκεύει την έξοδο του *LUT*



Μπλοκ Εισόδου/Εξόδου (I/O)

- Συνήθως, υποστηρίζουν συνδυαστική είσοδο/έξοδο ή μέσω καταχωρητή ή/και απομονωτή τριών καταστάσεων
 - Προγραμματιζόμενα επίπεδα λογικής, ρυθμός μεταβολής, κατώφλι εισόδου, ...

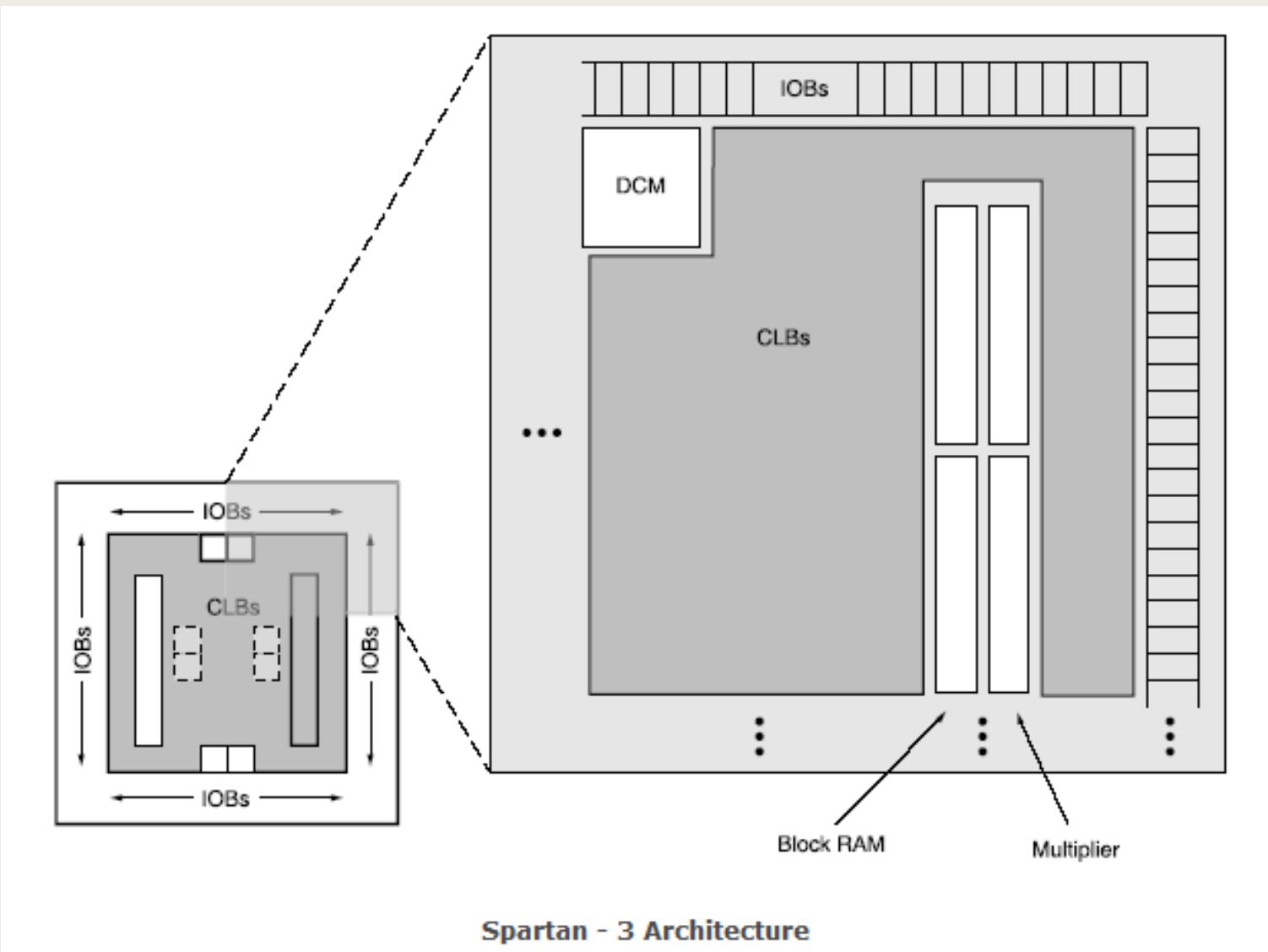


Η Αρχιτεκτονική των FPGAs της XILINX

- Παράδειγμα: Η αρχιτεκτονική του **Spartan III**
 - **Configurable Logic Blocks (CLBs)**
που περιέχουν πίνακες αναζήτησης (LUT), που υλοποιούν συνδυαστική λογική, και στοιχεία αποθήκευσης που μπορούν να χρησιμοποιηθούν ως D Flip-flops ή Latches
 - **Input/Output Blocks (IOBs)**
που ελέγχουν την ροή δεδομένων μεταξύ των I/O pins και της εσωτερικής λογικής
 - **Block RAMs**
που παρέχουν αποθηκευτικό χώρο μνήμης, μεγέθους για παράδειγμα 18Kbit (το μέγεθος εξαρτάται από την τεχνολογία)
 - **Multiplier Blocks**
σε πολλές οικογένειες οι πολλαπλασιαστές έχουν εξελιχθεί σε ειδικές μονάδες ψηφιακής επεξεργασίας σήματος
 - **Digital Clock Manager (DCM) Blocks**
που παρέχουν αυτορρυθμιζόμενες πλήρως ψηφιακές λύσεις για κατανομή, καθυστέρηση, διαίρεση και ρύθμιση της φάσης των ρολογιών
 - Τα blocks διασυνδέονται μέσω **προγραμματιζόμενων πινάκων διακοπτών** (switch matrices)

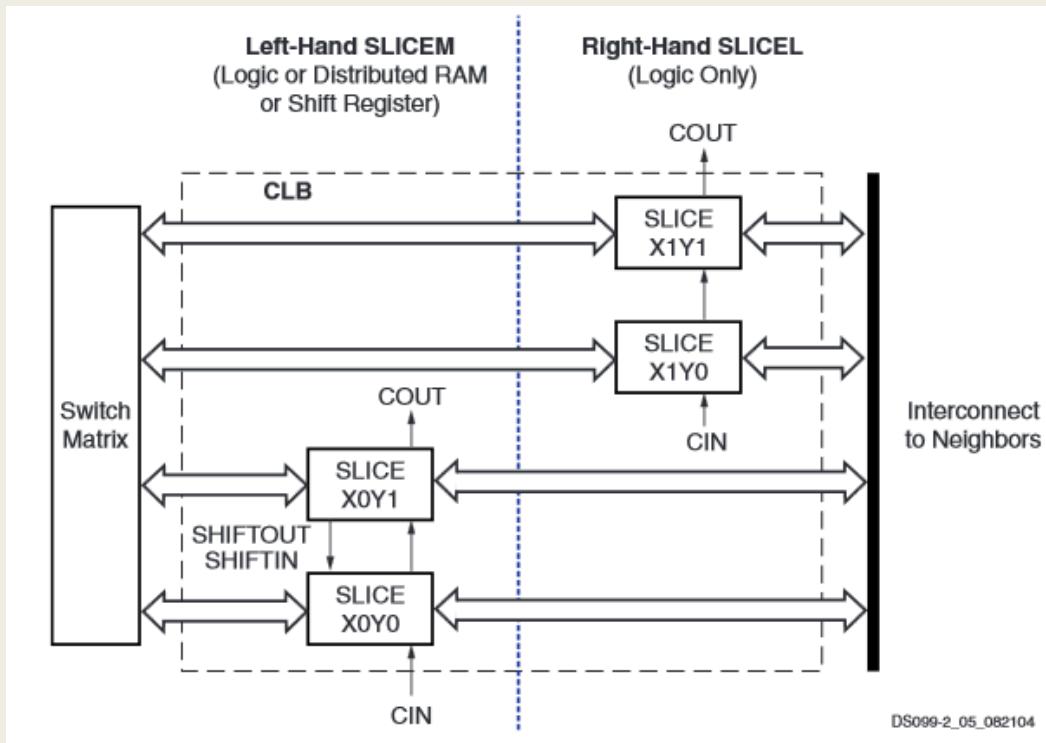
Η Αρχιτεκτονική των FPGAs της XILINX

- Παράδειγμα: Η αρχιτεκτονική του Spartan III



Η Αρχιτεκτονική των FPGAs της XILINX

- Παράδειγμα: Η αρχιτεκτονική του **Spartan III**
 - Κάθε CLB αποτελείται από τέσσερα συνδεδεμένα **slices** τα οποία ομαδοποιούνται σε ζευγάρια και κάθε ζευγάρι σχηματίζει μια ανεξάρτητη αλυσίδα διάδοσης κρατουμένου.
 - Το αριστερό ζευγάρι υλοποιεί λογική, κατανεμημένη μνήμη ή καταχωρητή ολίσθησης
 - Το δεξιό ζευγάρι υλοποιεί αποκλειστικά λογική



Η Αρχιτεκτονική των FPGAs της XILINX

■ Παράδειγμα: Η αρχιτεκτονική του Spartan III

- Κάθε slice του CLB έχει : 2 πίνακες αναζήτησης (LUT) των 4 εισόδων, 2 στοιχεία αποθήκευσης, πολυπλέκτες, αλυσίδα διάδοσης κρατούμενου και πύλες για αριθμητική λογική

