

Ιόνιο Πανεπιστήμιο – Τμήμα Πληροφορικής  
Εισαγωγή στην Επιστήμη των Υπολογιστών  
2023-24

# Πράξεις με δυαδικούς αριθμούς

(λογικές πράξεις)

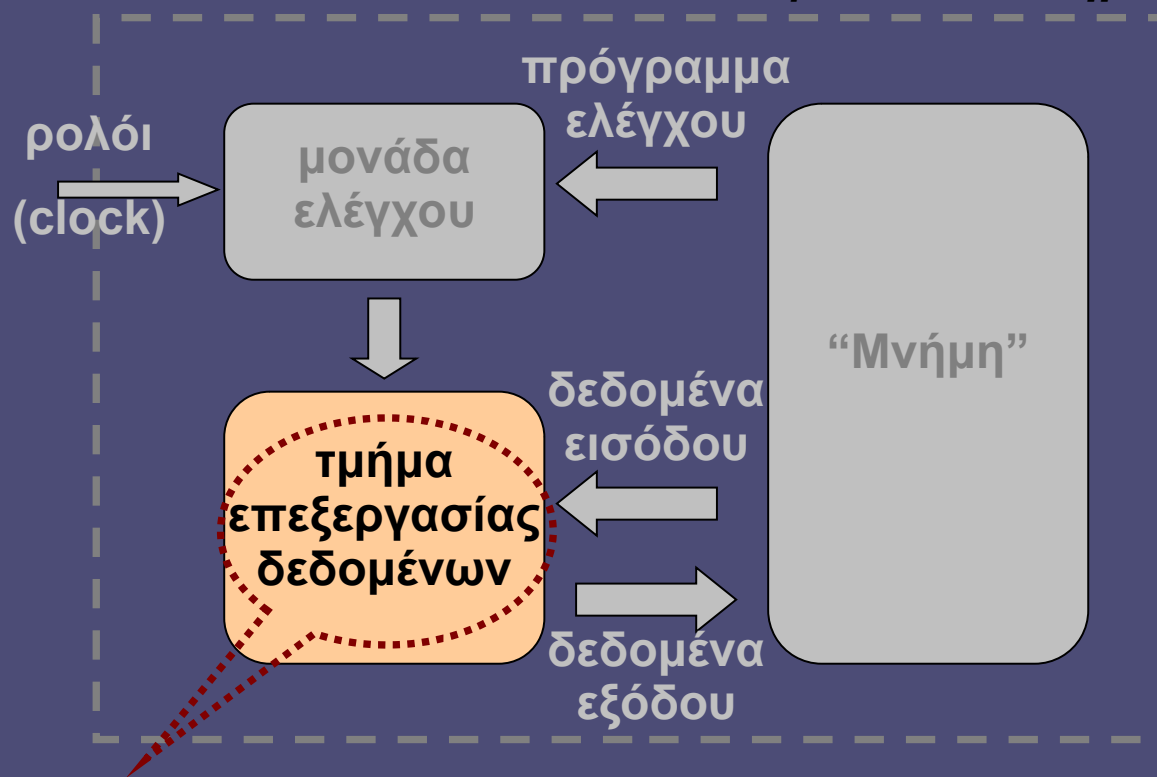
<http://mixstef.github.io/courses/csintro/>

Μ.Στεφανιδάκης



# Εκτέλεση πράξεων

*υπολογιστικό σύστημα*



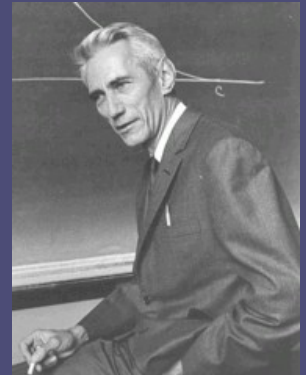
- Επεξεργασία από ψηφιακά δυαδικά κυκλώματα
  - που εκτελούν πράξεις μεταξύ σειρών 0 και 1...
  - ...οι οποίες αναπαριστούν δυαδικούς αριθμούς

# Πράξεις με δυαδικούς αριθμούς

- Ο υπολογιστής μπορεί να εκτελέσει
  - Λογικές πράξεις (δυαδικής λογικής)
  - Αριθμητικές πράξεις (πρόσθεση κλπ)
- Οι πράξεις εκτελούνται σε ομάδες bits που ονομάζουμε «δυαδικούς αριθμούς»
  - **Bit**: η μικρότερη λογική ποσότητα – η μικρότερη μονάδα δεδομένων – 0 ή 1.
  - **Byte**: ομάδα 8 bits
    - Συχνά η **ελάχιστη** ποσότητα που μπορεί να χειριστεί ο υπολογιστής κατά την εκτέλεση μιας πράξης

# Ψηφιακά Ηλεκτρονικά και Δυαδική Λογική

- Η δυαδική λογική ταιριάζει με την τεχνολογία του τρανζίστορ ως διακόπτη
  - 2 καταστάσεις: ON-OFF, 1-0
  - Ψηφιακά ηλεκτρονικά (2 στάθμες)
- Δυαδική άλγεβρα Boole
  - Λογική άλγεβρα
  - Υλοποίηση με διακοπτικά κυκλώματα
    - Η πρωτοποριακή εργασία του **Shannon**: “A Symbolic Analysis of Relay and Switching Circuits” (1938)



C.E.Shannon

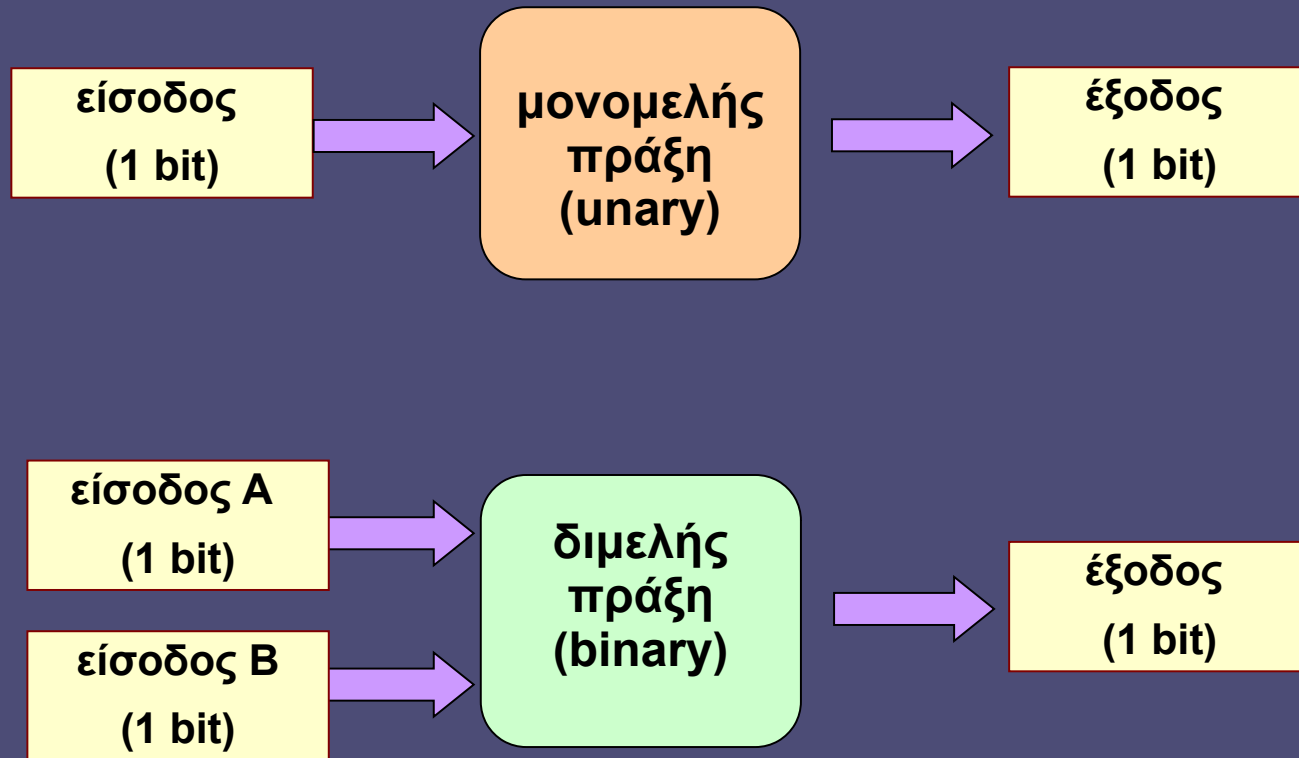
# Ψηφιακά Ηλεκτρονικά και Δυαδική Λογική (2)

- Στη δυαδική λογική άλγεβρα
  - Υπάρχουν 2 «ποσότητες» (σύμβολα):
    - Αληθές ή 1 ή ΝΑΙ
    - Ψευδές ή 0 ή ΟΧΙ
  - Ένα δυαδικό ψηφίο (bit) έχει τιμή 0 ή 1
- Στα ψηφιακά ηλεκτρονικά κυκλώματα ένα bit αναπαρίσταται με αντίστοιχη κατάσταση σε ένα ηλεκτρονικό κύκλωμα (ανάλογα με την τεχνολογία)
  - $0 \rightarrow$  «χαμηλή τάση» ή «η μια φορά ρεύματος»
  - $1 \rightarrow$  «υψηλή τάση» ή «η άλλη φορά ρεύματος»

# Πράξεις Δυαδικής Λογικής

- Στη δυαδική λογική άλγεβρα
  - Καθορίζονται λογικές πράξεις μεταξύ των λογικών ποσοτήτων 0 και 1 (bits)
- Στα ψηφιακά ηλεκτρονικά κυκλώματα:
  - Κύκλωμα δέχεται ως είσοδο την ηλεκτρική αναπαράσταση των 0 και 1
  - Και παράγει στην έξοδό του την ηλεκτρική αναπαράσταση του αποτελέσματος μιας λογικής πράξης
  - Το κύκλωμα υλοποίησης της λογικής πράξης ονομάζεται πύλη (gate).

# Λογικές πράξεις με bits



# Λογικές πράξεις με bits

- Μονομελής λογική πράξη
  - NOT (αντιστροφή)
- Διμελείς λογικές πράξεις
  - AND (λογικό-ΚΑΙ)
  - OR (λογικό-Η)
  - XOR (αποκλειστικό-Η)
  - κ.λ.π.



# Βασικές Λογικές Πράξεις

- Αντιστροφή (NOT)
  - Αντιστροφή ενός bit

είσοδος A

έξοδος NOT (A)  
ή A' ή  $\overline{A}$

A	Y
0	1
1	0

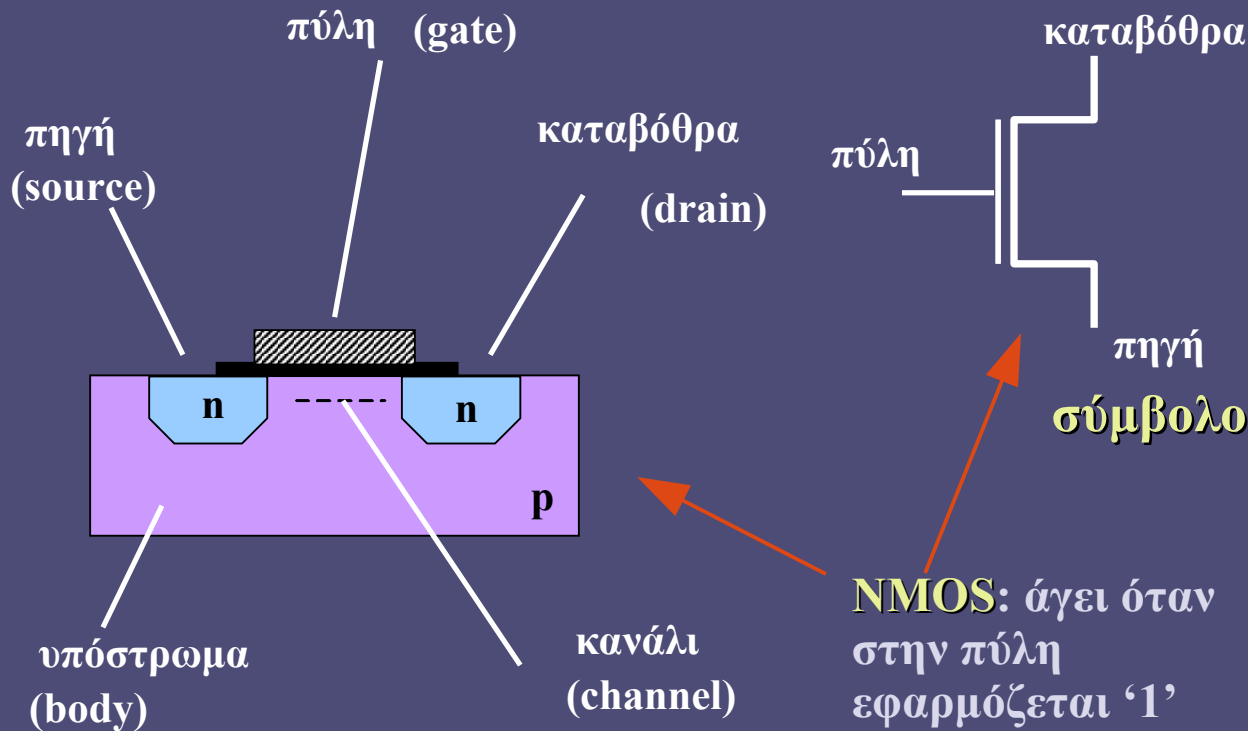
πιθανές τιμές εισόδου

αντίστοιχες τιμές εξόδου

Πίνακας Αλήθειας

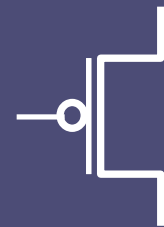
# Πώς μπορεί να υλοποιηθεί μια πύλη NOT;

- Από το προηγούμενο μάθημα:



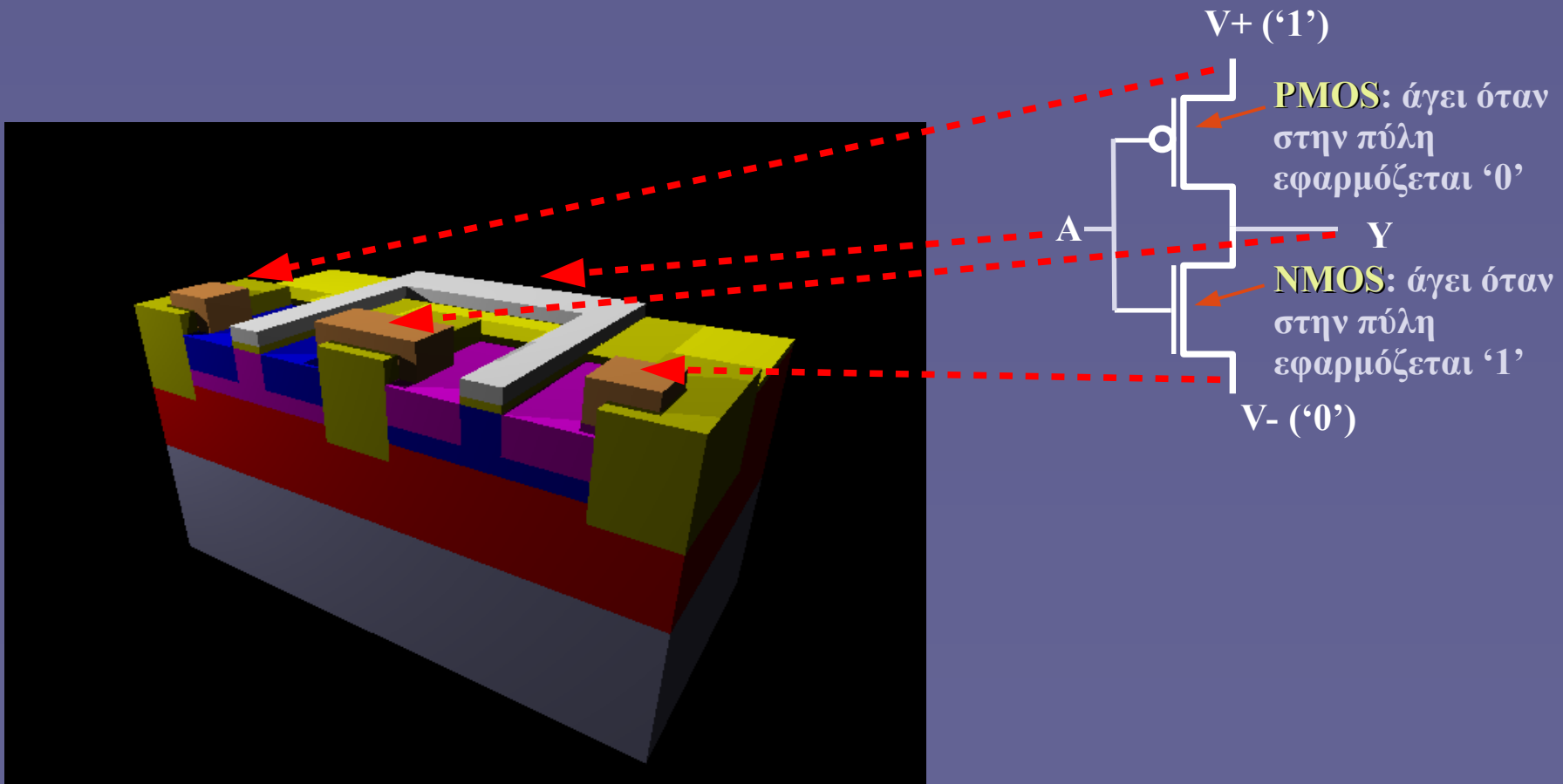
Το τρανζίστορ NMOS

**PMOS:** άγει όταν στην πύλη εφαρμόζεται '0'



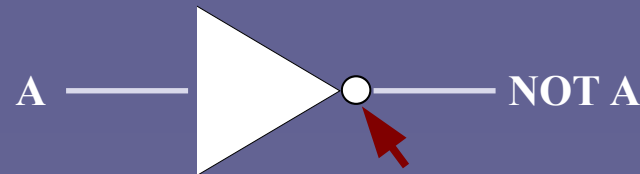
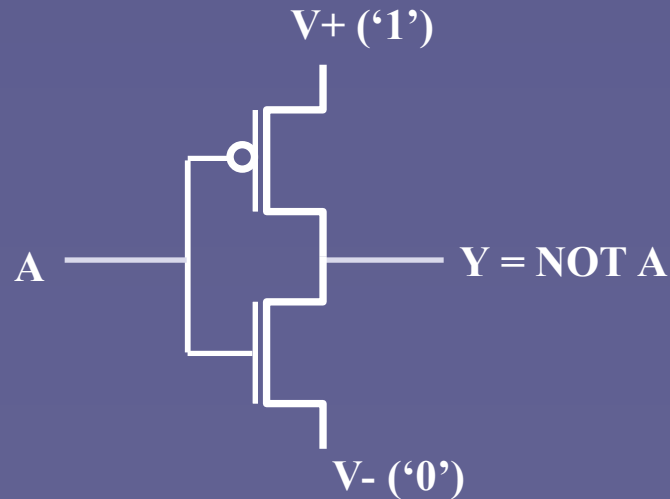
**NMOS:** άγει όταν στην πύλη εφαρμόζεται '1'

# Από το προηγούμενο μάθημα: ποια η λειτουργία του;



# Η πύλη NOT (αντιστροφέας)

A	Y
0	1
1	0



ο κύκλος συμβολίζει την αντιστροφή

**σύμβολο πύλης NOT**

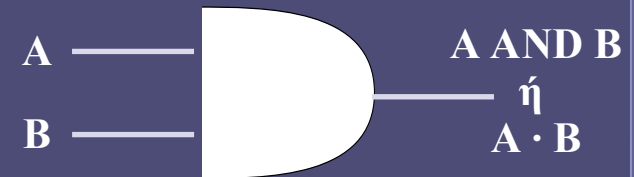
# Βασικές Λογικές Πράξεις

- Λογικό ΚΑΙ (AND)

- το αποτέλεσμα είναι 1, μόνο όταν και το A και το B είναι 1
- $0 \text{ AND } x = x \text{ AND } 0 = 0$
- $1 \text{ AND } x = x \text{ AND } 1 = x$   
( $x = \text{οποιαδήποτε τιμή, είτε } 0 \text{ είτε } 1$ )

Πίνακας Αλήθειας

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

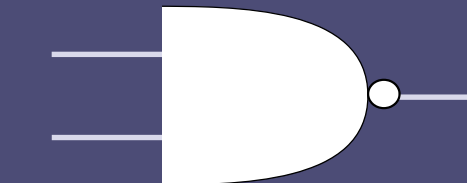
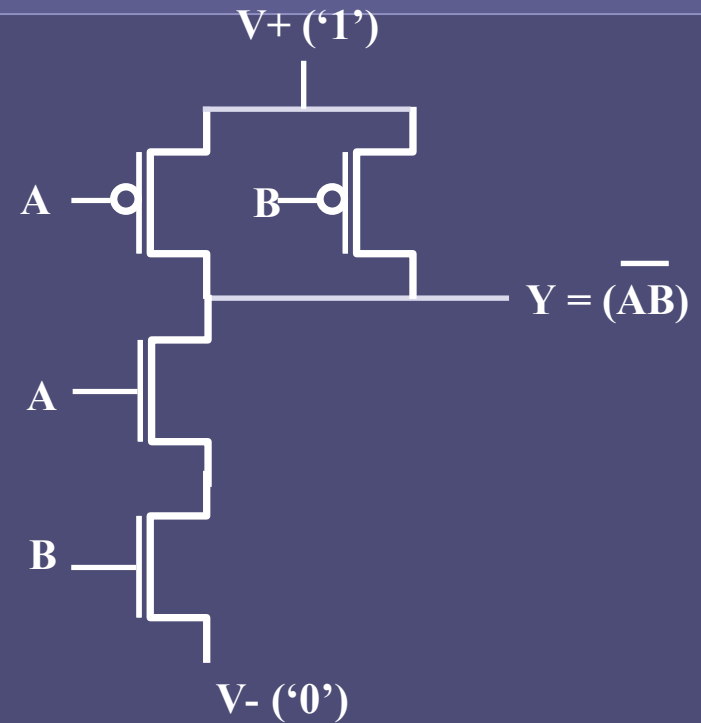
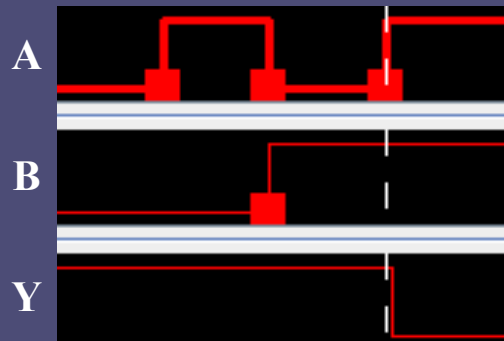


σύμβολο πύλης AND

# Παράδειγμα υλοποίησης: η πύλη NAND

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

Πίνακας Αλήθειας NAND

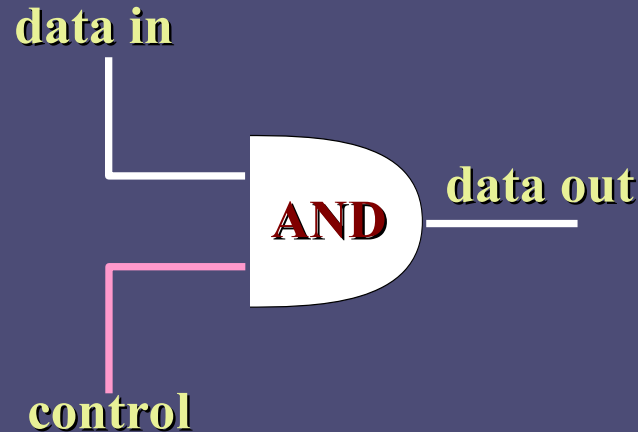


σύμβολο πύλης NAND

!  
NAND = NOT-AND

Υλοποίηση πύλης AND: χρησιμοποιώντας μια πύλη NAND και μια πύλη NOT

# Φραγή AND: για να θέσουμε σήμα στο 0

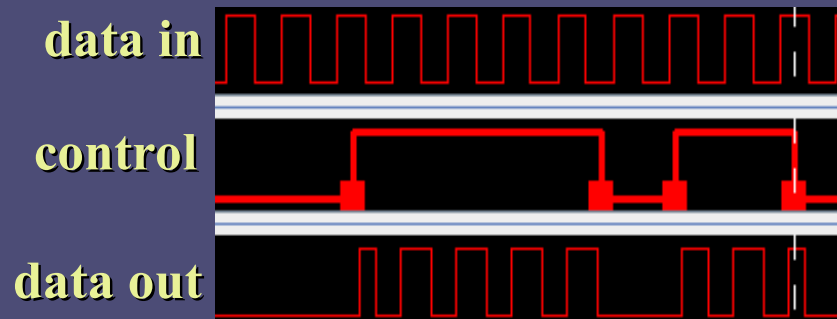


$\text{control} = 0 \rightarrow$  η έξοδος  $\text{data\_out}$  είναι πάντα 0

$\text{control} = 1 \rightarrow$  η έξοδος  $\text{data\_out}$  ισούται με το  $\text{data\_in}$

$$0 \text{ AND } x = 0$$

$$1 \text{ AND } x = x$$



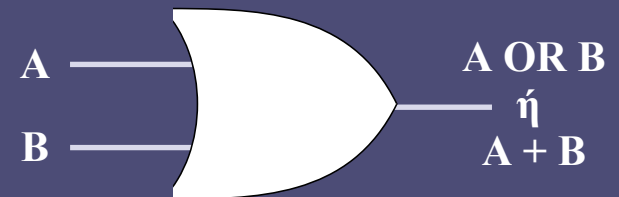
# Βασικές Λογικές Πράξεις

- Λογικό Ή (OR)

- το αποτέλεσμα είναι 1, όταν το A ή το B ή και τα δύο είναι 1
- $1 \text{ OR } x = x \text{ OR } 1 = 1$
- $0 \text{ OR } x = x \text{ OR } 0 = x$   
( $x = \text{οποιαδήποτε τιμή, είτε } 0 \text{ είτε } 1$ )

Πίνακας Αλήθειας

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1



σύμβολο πύλης OR



# Παράδειγμα υλοποίησης: η πύλη NOR

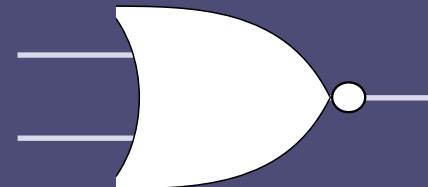
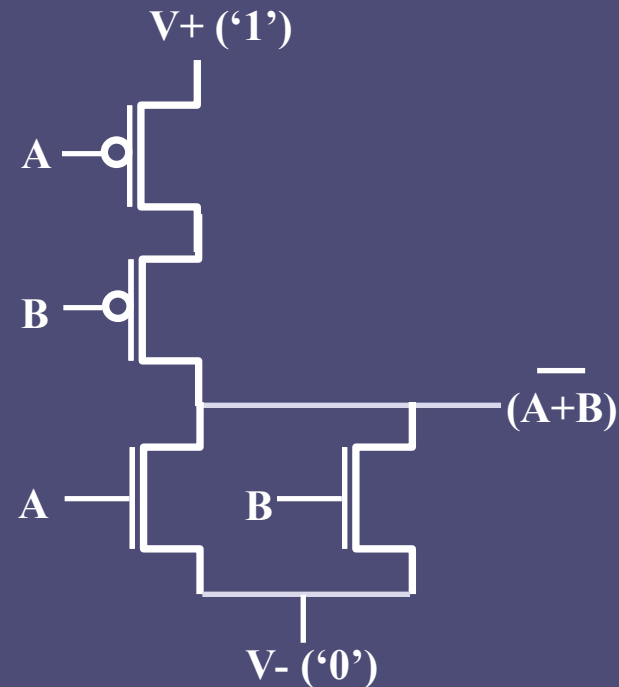
!

NOR = NOT-OR

Υλοποίηση πύλης  
OR: χρησιμοποι-  
ώντας μια πύλη  
NOR και μια πύλη  
NOT

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

Πίνακας Αλήθειας NOR



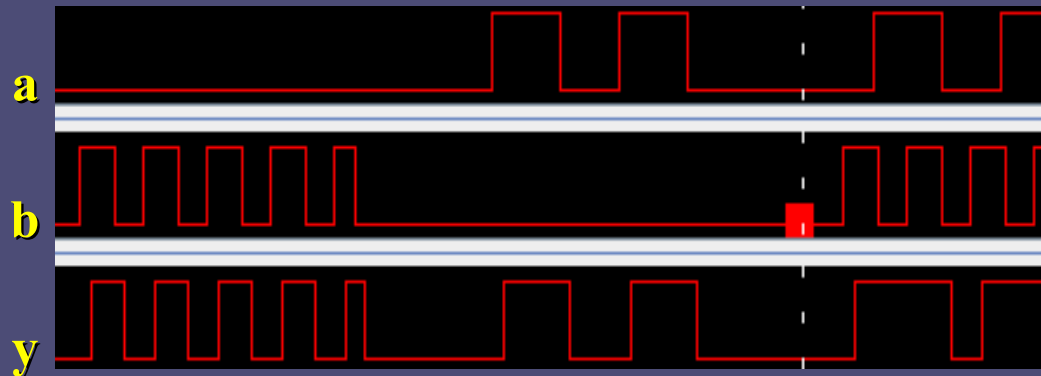
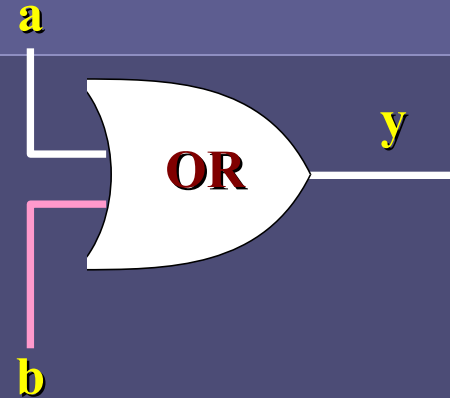
σύμβολο πύλης NOR

# Συγκέντρωση σημάτων με OR

$$0 \text{ OR } x = x$$

$$1 \text{ OR } x = 1$$

Προσοχή!  
Ποτέ δεν  
συνδέουμε  
εξόδους  
πυλών μαζί!



- Ουσιαστικά, **επιλογή** μεταξύ των  $a$  και  $b$ 
  - Για να λειτουργήσει όμως σωστά θα πρέπει ανά πάσα στιγμή όλα τα σήματα πλην ενός να είναι 0

# Βασικές Λογικές Πράξεις

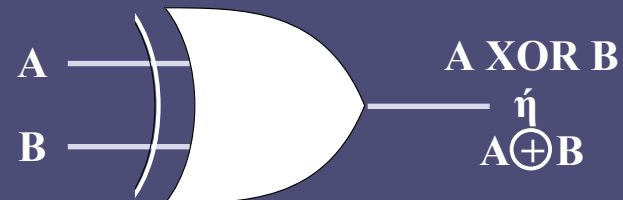
- Αποκλειστικό Ή (XOR)

- Το αποτέλεσμα είναι 1, όταν **μόνο το A** ή **μόνο το B** είναι 1
- Ορίζεται και ως  $A \text{ XOR } B = A \cdot B' + A' \cdot B$
- $1 \text{ XOR } x = x \text{ XOR } 1 = \text{NOT } x$
- $0 \text{ XOR } x = x \text{ XOR } 0 = x$

( $x = \text{οποιαδήποτε τιμή, είτε 0 είτε 1}$ )

Πίνακας Αλήθειας

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



**σύμβολο πύλης XOR**

# Βασικές Λογικές Πράξεις

- **XNOR**: Η συμπληρωματική συνάρτηση της XOR
  - Το αποτέλεσμα είναι 1, όταν τα A και B είναι όμοια
    - Συνάρτηση «ισοδυναμίας»
  - Ορίζεται και ως  $A \text{ XNOR } B = A \cdot B + A' \cdot B'$


Πίνακας Αλήθειας

A	B	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

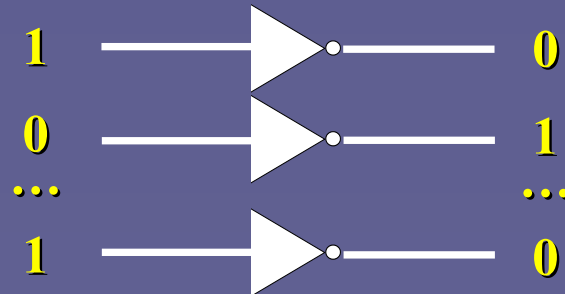
# Λογικές πράξεις σε ομάδες bits

- Ο υπολογιστής μπορεί να εφαρμόσει λογικές πράξεις στα δεδομένα μας
  - Δεδομένα = σειρές από 0 και 1
  - Όχι όμως σε μεμονωμένα bits
  - Αλλά: σε ομάδες («λέξεις») των 8, 16, 32 ή 64 bits ταυτόχρονα

$$\begin{array}{ccccccc} A_n & \dots & A_i & \dots & A_2 & A_1 & A_0 \\ B_n & \dots & B_i & \dots & B_2 & B_1 & B_0 \\ \hline Y_n & \dots & Y_i & \dots & Y_2 & Y_1 & Y_0 \end{array} \quad \text{op} \quad (= \text{AND, OR, XOR})$$

  $Y_i = A_i \text{ op } B_i$

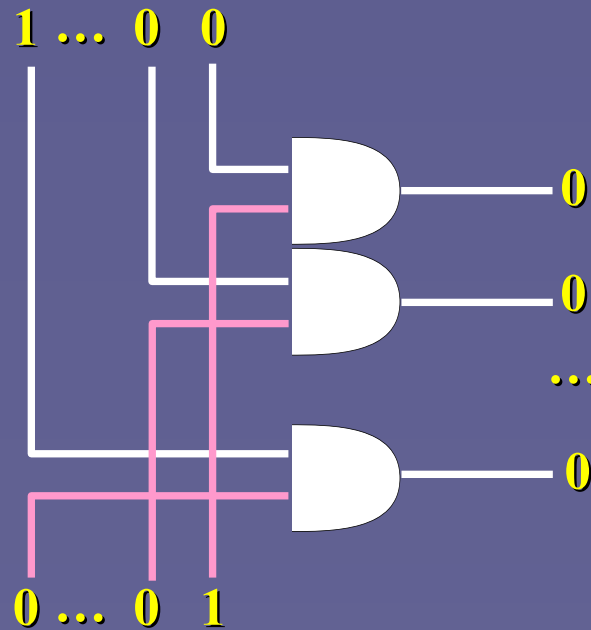
# Ο τελεστής NOT σε δυαδικούς αριθμούς



1	0	0	1	1	0	0	0	NOT
<hr/>								
0	1	1	0	0	1	1	1	

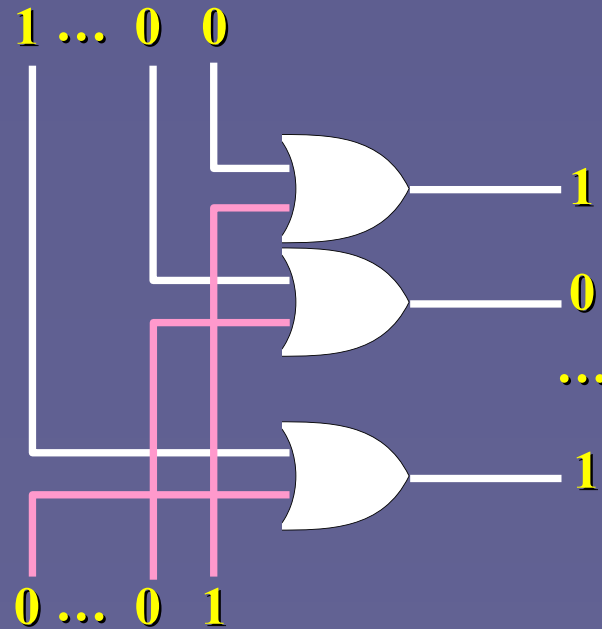
- Η έξοδος  $Y_i$  εξαρτάται μόνο από την είσοδο  $A_i$

# Ο τελεστής AND σε δυαδικούς αριθμούς



$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0 \\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ \hline 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \end{array} \quad \text{AND}$$

# Ο τελεστής OR σε δυαδικούς αριθμούς



1	0	0	1	1	0	0	0	OR
0	0	1	1	0	1	0	1	
<hr/>								
1	0	1	1	1	1	0	1	



# Μάσκες

- Για να αλλάξουμε την τιμή μεμονωμένων bits μέσα σε μια λέξη (bits εισόδου)
  - Για να θέσουμε συγκεκριμένα bits σε 1
  - Για να θέσουμε συγκεκριμένα bits σε 0
  - Για να αντιστρέψουμε συγκεκριμένα bits
  - Χωρίς όμως να επηρεάζουμε τα υπόλοιπα!
    - αυτά διατηρούν την τιμή τους, είτε 0 είτε 1
- Μάσκα: σειρά bits, επιλεγμένη ώστε:  
*Bits Εισόδου op Μάσκα → Νέα ομάδα bits*
  - op = AND, OR ή XOR
  - Η νέα ομάδα περιέχει το επιθυμητό αποτέλεσμα

# Μάσκα AND: για να θέσουμε bits στο 0

- Παράδειγμα: σε λέξη των 8 bits να τεθούν σε 0 τα 3 λιγότερο σημαντικά bits.

Λέξη:	1 0 0 1 1 0 1 0	AND
Μάσκα:	1 1 1 1 1 0 0 0	
Νέα:	1 0 0 1 1 0 0 0	

- Η AND μάσκα περιέχει:
  - 0 στα bits που θα γίνουν 0
  - 1 στα bits που θα παραμείνουν ως έχουν

$$0 \text{ AND } x = 0$$

$$1 \text{ AND } x = x$$

# Μάσκα OR: για να θέσουμε bits στο 1

- Παράδειγμα: σε λέξη των 8 bits να τεθούν σε 1 τα bits 0,4 και 5.

Λέξη:	1 0 0 1 1 0 0 0	OR
Μάσκα:	0 0 1 1 0 0 0 1	
Νέα:	1 0 1 1 1 0 0 1	

- Η OR μάσκα περιέχει:
  - 1 στα bits που θα γίνουν 1
  - 0 στα bits που θα παραμείνουν ως έχουν

$$0 \text{ OR } x = x$$

$$1 \text{ OR } x = 1$$

# Μάσκα XOR: για να αντιστρέψουμε bits

- Παράδειγμα: σε λέξη των 8 bits να αντιστραφούν τα bits 3,6 και 7.

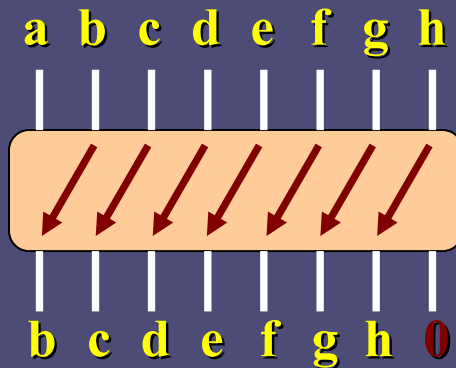
Λέξη:	1 0 0 1 1 0 0 0	XOR
Μάσκα:	1 1 0 0 1 0 0 0	
Νέα:	0 1 0 1 0 0 0 0	

- Η XOR μάσκα περιέχει:
  - 1 στα bits που θα αντιστραφούν
  - 0 στα bits που θα παραμείνουν ως έχουν

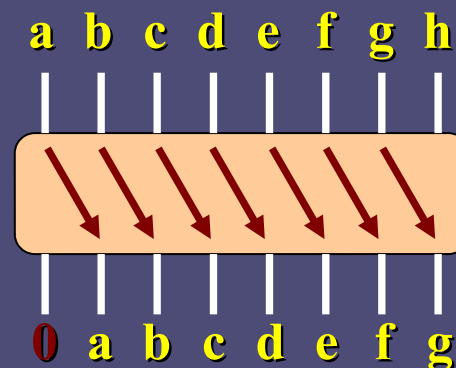
$$\begin{aligned}0 \text{ XOR } x &= x \\1 \text{ XOR } x &= x'\end{aligned}$$

# Ολίσθηση (Shift)

- Τυπικά δεν είναι πράξη της δυαδικής λογικής
  - Είναι όμως μια πολύ χρήσιμη και γρήγορη λειτουργία για πολλαπλασιασμό ή διαίρεση με δυνάμεις του 2 (2,4,8...)
  - Στο άκρο που «αδειάζει» εισάγεται το 0
    - Αν εισάγουμε το άλλο άκρο (που «χάνεται») έχουμε **περιστροφή** (rotation)



αριστερή ολίσθηση



δεξιά ολίσθηση