Έκδοση RISC_V 2025

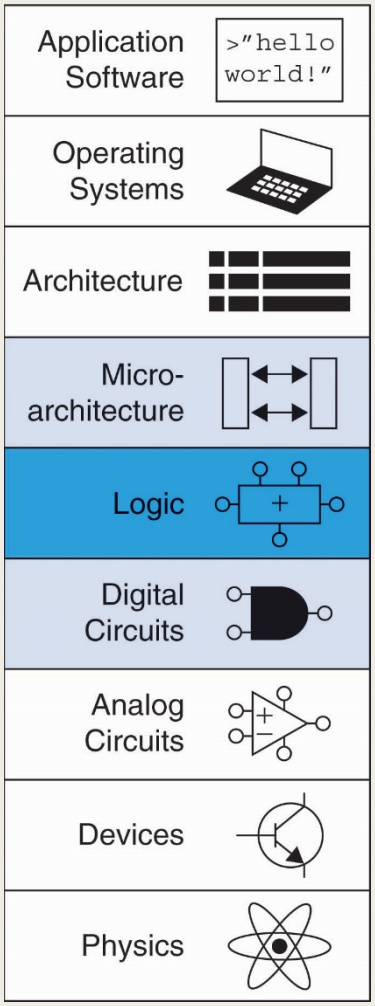
Κεφάλαιο 5

Ψηφιακά δομικά στοιχεία

Καθηγητής Αντώνης Πασχάλης

Περιεχόμενα κεφαλαίου

- Αριθμητικά κυκλώματα
  - *Αθροιστές – Αφαιρέτες*
  - *Συγκριτές*
  - *Αριθμητική/Λογική Μονάδα (ALU)*
  - *Ολισθητές και περιστροφείς*
  - *Δυαδικός πολλαπλασιαστής*
- Πραγματικοί αριθμοί σταθερής υποδιαστολής
- Μετρητές
  - *Ψηφιακά ελεγχόμενος ταλαντωτής*
- Καταχωρητές ολίσθησης
- Αρχεία καταχωρητών
- Διατάξεις μνήμης (DRAM, SRAM, ROM)
- Διατάξεις λογικής (PLA, FPGA)



Αριθμητικά κυκλώματα

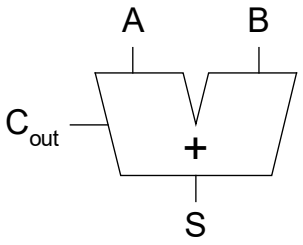
- Τα αριθμητικά κυκλώματα αποτελούν τα κεντρικά δομικά στοιχεία των υπολογιστών και των ψηφιακών συστημάτων γενικότερα
- Οι υπολογιστές και τα στοιχεία ψηφιακής λογικής εκτελούν πολλές αριθμητικές λειτουργίες ή πράξεις:
  - *πρόσθεση*
  - *αφαίρεση*
  - *συγκρίσεις*
  - *ολισθήσεις*
  - *πολλαπλασιασμό*
  - *διαίρεση*
- Οι αριθμητικές πράξεις μεταξύ ακεραίων αριθμών εκτελούνται στην **Αριθμητική/Λογική Μονάδα (Arithmetic Logic Unit – ALU)** των υπολογιστών

Αριθμητικά κυκλώματα

- Τα δομικά στοιχεία επιδεικνύουν τα τρια «Α»: 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

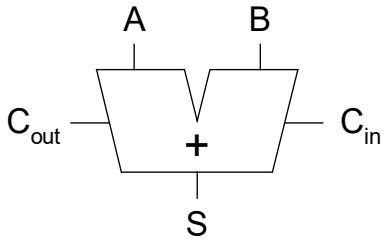


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$

$C_{out} = AB$

Full Adder



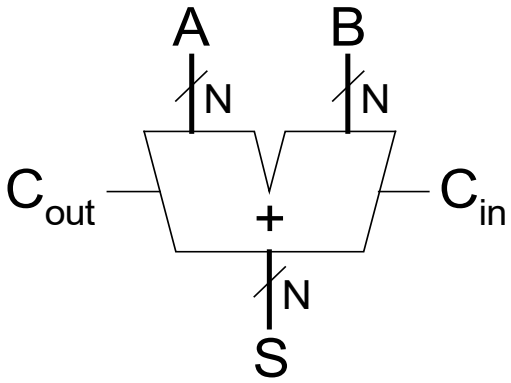
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

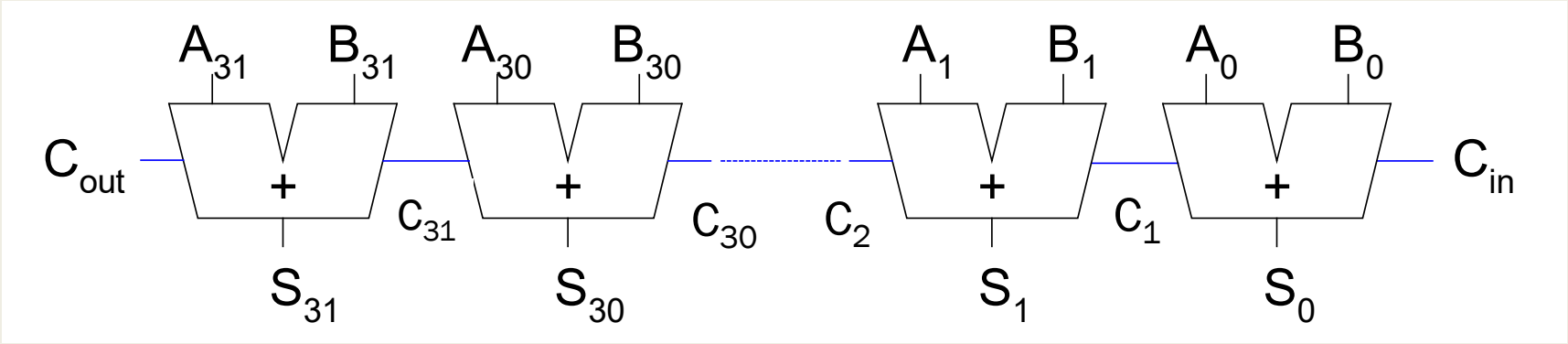
$S = A \oplus B \oplus C_{in}$

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

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)*

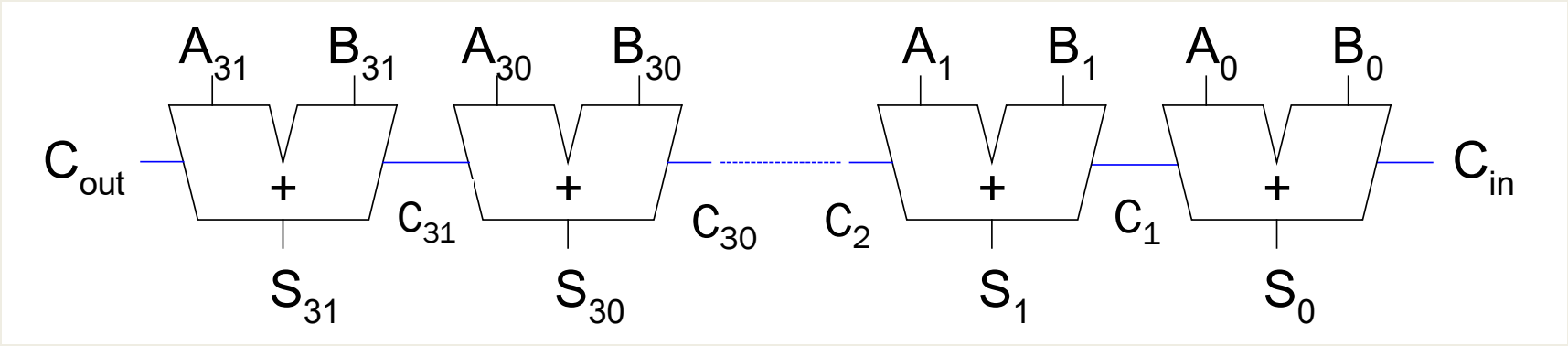




Αθροιστής κυμάτωσης κρατουμένου

- 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*
- Για έναν αθροιστή κυμάτωσης κρατούμενου των ***N*** bit η καθυστέρηση διάδοσης είναι ***N*** φορές η καθυστέρηση διάδοσης του πλήρη αθροιστή t_{FA} (υποθέτοντας την ίδια καθυστέρηση διάδοσης για S και C)

$$t_{RCA} = Nt_{FA}$$



Αθροιστής (adder) των *N* bit

- **Αθροιστής Πρόβλεψης Κρατουμένου (carry lookahead adder - CLA)**
  - *Διαμερίζει 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 (generate)** και **διάδοσης (propagate) 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 $G_i = A_i B_i$***
    - 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:

$$C_4 = G_3 + P_3 C_3$$

$$C_3 = G_2 + P_2 C_2$$

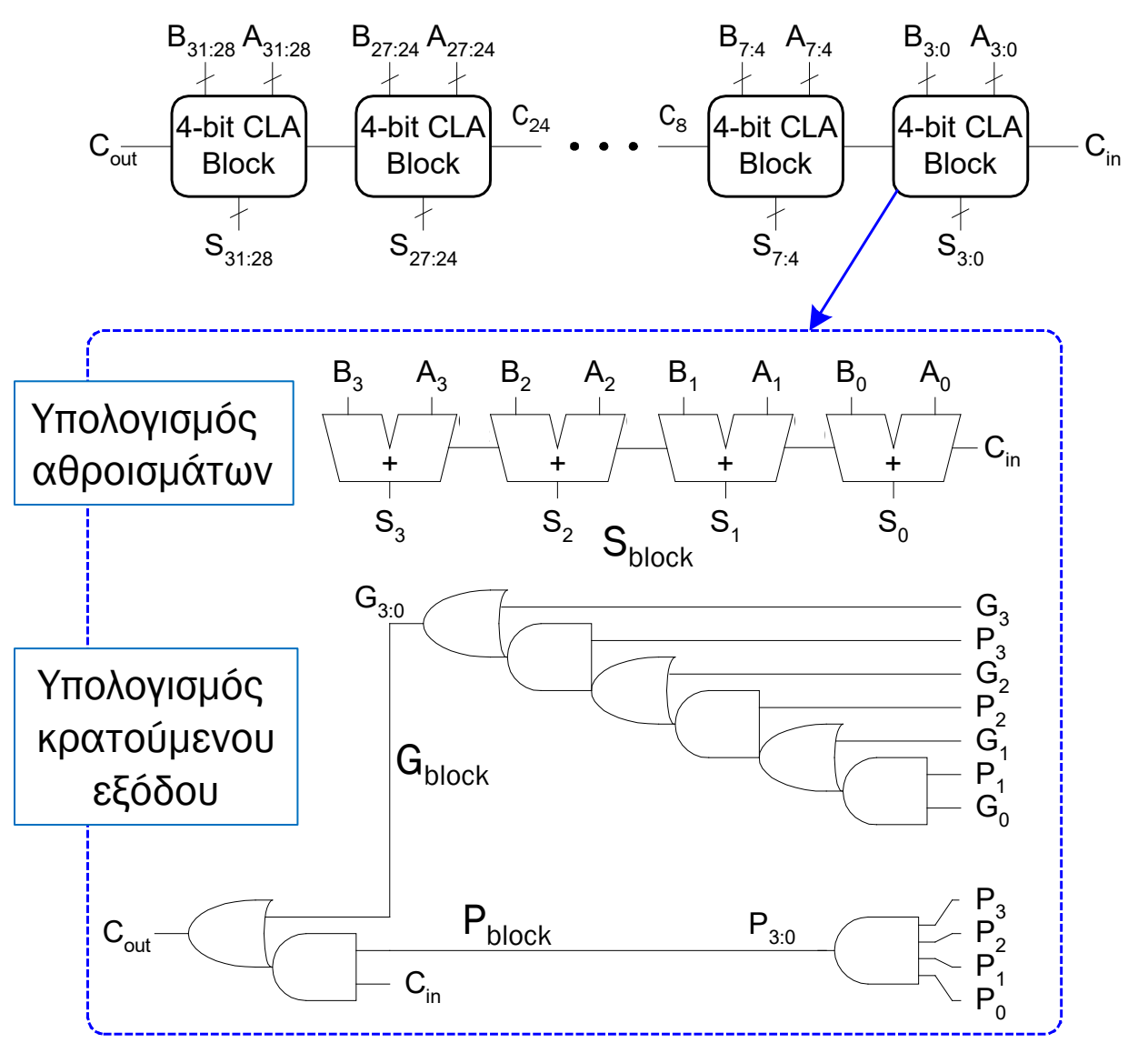
$$C_2 = G_1 + P_1 C_1$$

$$C_1 = G_0 + P_0 C_0$$

$$C_4 = G_3 + P_3 (G_2 + P_2 (G_1 + P_1 (G_0 + P_0 C_0)))$$

$$C_{out} = G_3 + P_3 (G_2 + P_2 (G_1 + P_1 G_0)) + (P_3 P_2 P_1 P_0) C_{in}$$

Αθροιστής πρόβλεψης κρατουμένου



$$C_{out} = G_3 + P_3 (G_2 + P_2 (G_1 + P_1 G_0)) + (P_3 P_2 P_1 P_0) C_{in}$$

Αθροιστής πρόβλεψης κρατουμένου

- Βήμα 1: Υπολογίζει παράλληλα όλα τα σήματα παραγωγής και διάδοσης κρατούμενου $G_i = A_i B_i$ και $P_i = A_i + B_i$
- Βήμα 2: Υπολογίζει παράλληλα όλα τα σήματα G_{block} και P_{block} για κάθε ένα από τους τμηματικούς αθροιστές των *k=4* bit ξεχωριστά
- Βήμα 3: Υπολογίζει το κρατούμενο εξόδου C_{out} του τμηματικού αθροιστή αμέσως μόλις γίνει γνωστή η τιμή του κρατουμένου εισόδου του C_{in} με διάδοση κρατούμενου στο επίπεδο του τμηματικού αθροιστή
- Βήμα 4: Υπολογίζει τα αθροίσματα S_{block} του τμηματικού αθροιστή των *k=4* bit αμέσως μόλις γίνει γνωστή η τιμή του κρατουμένου εισόδου του C_{in} με διάδοση κρατούμενου στο επίπεδο του τμηματικού αθροιστή

Στο επίπεδο του τμηματικού αθροιστή (CLA block) των *k=4* bit η καθυστέρηση διάδοσης από το C_{in} στο C_{out} (βήμα 3) είναι πολύ μικρότερη από την καθυστέρηση διάδοσης από το C_{in} στο S_{block} (βήμα 4)

Αθροιστής πρόβλεψης κρατουμένου

- Σε όλους τους τμηματικούς αθροιστές όλα τα σήματα παραγωγής και διάδοσης κρατούμενου δημιουργούνται παράλληλα με καθυστέρηση διάδοσης t_{pg}
- Η κρίσιμη διαδρομή ξεκινάει με τον υπολογισμό των G_0 και $G_{3:0}$ (G_{block}) στον πρώτο τμηματικό αθροιστή (CLA block) με καθυστέρηση διάδοσης t_{pg_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 με καθυστέρηση διάδοσης t_{AND_OR}
- Τέλος, η κρίσιμη διαδρομή μέσω του τελευταίου τμηματικού αθροιστή περιέχει έναν μικρό αθροιστή κυμάτωσης κρατουμένου των $k=4$ bit που υπολογίζει τα αθροίσματα $S_{31:28}$ (S_{block}) όταν εμφανισθεί το C_{28} με καθυστέρηση διάδοσης $k=4$ φορές την καθυστέρηση διάδοσης του πλήρη αθροιστή t_{FA} (υποθέτοντας την ίδια καθυστέρηση διάδοσης για S και C)
- Για έναν αθροιστή πρόβλεψης κρατούμενου των N bit που απαρτίζεται από τμηματικούς αθροιστές (CLA blocks) των $k=4$ bit η καθυστέρηση διάδοσης είναι:

$$t_{CLA} = t_{pg} + t_{pg\_block} + (N/4 - 1)t_{AND\_OR} + 4t_{FA}$$

Συγκρίσεις καθυστερήσεων διάδοσης στους αθροιστές κυμάτωσης και πρόβλεψης κρατούμενου

- Καθυστέρηση διάδοσης του **αθροιστή κυμάτωσης** μεγέθους N bit

$$t_{RCA} = Nt_{FA}$$

- Καθυστέρηση διάδοσης του **αθροιστή πρόβλεψης** μεγέθους N bit (με CLA blocks των 4 bit)

$$t_{CLA} = t_{pg} + t_{pg\_block} + (N/4 - 1)t_{AND\_OR} + 4t_{FA}$$

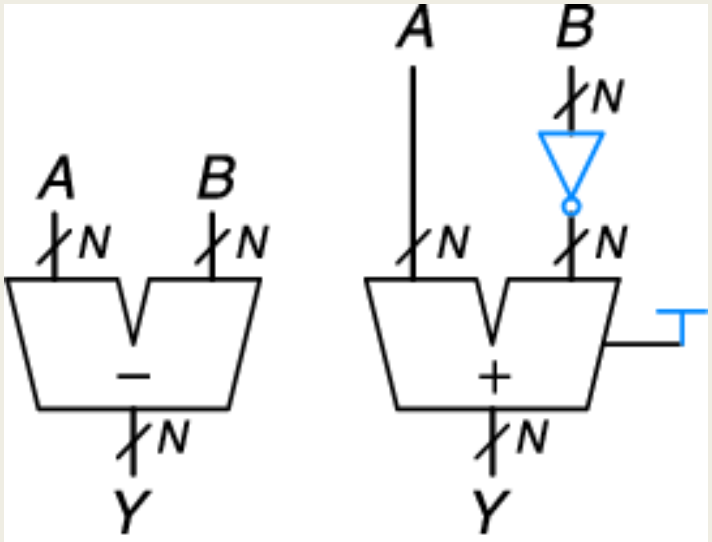
- Παραδείγματα για N=8, N=16 και N=32*
  - *Υποθέτοντας καθυστερήσεις διάδοσης στις πύλες 2 εισόδων στα 100 ps και στον πλήρη αθροιστή στα 250 ps*

N	8	16	32
$t_{RCA} = Nt_{FA}$	2.000 ps	4.000 ps	8.000 ps
t_{pg} (1 πύλη)	100 ps	100 ps	100 ps
t_{pg_block} (6 πύλες)	600 ps	600 ps	600 ps
$(N/4-1)t_{AND\_OR}$ (2 πύλες)	200 ps	600 ps	1.400 ps
$4t_{FA}$	1.000 ps	1.000 ps	1.000 ps
$t_{CLA} = t_{pg} + t_{pg\_block} + (N/4-1)t_{AND\_OR} + 4t_{FA}$	1.900 ps	2.300 ps	3.100 ps

* Αντίστοιχο Παράδειγμα 5.1 με t_{FA}= 300 ps

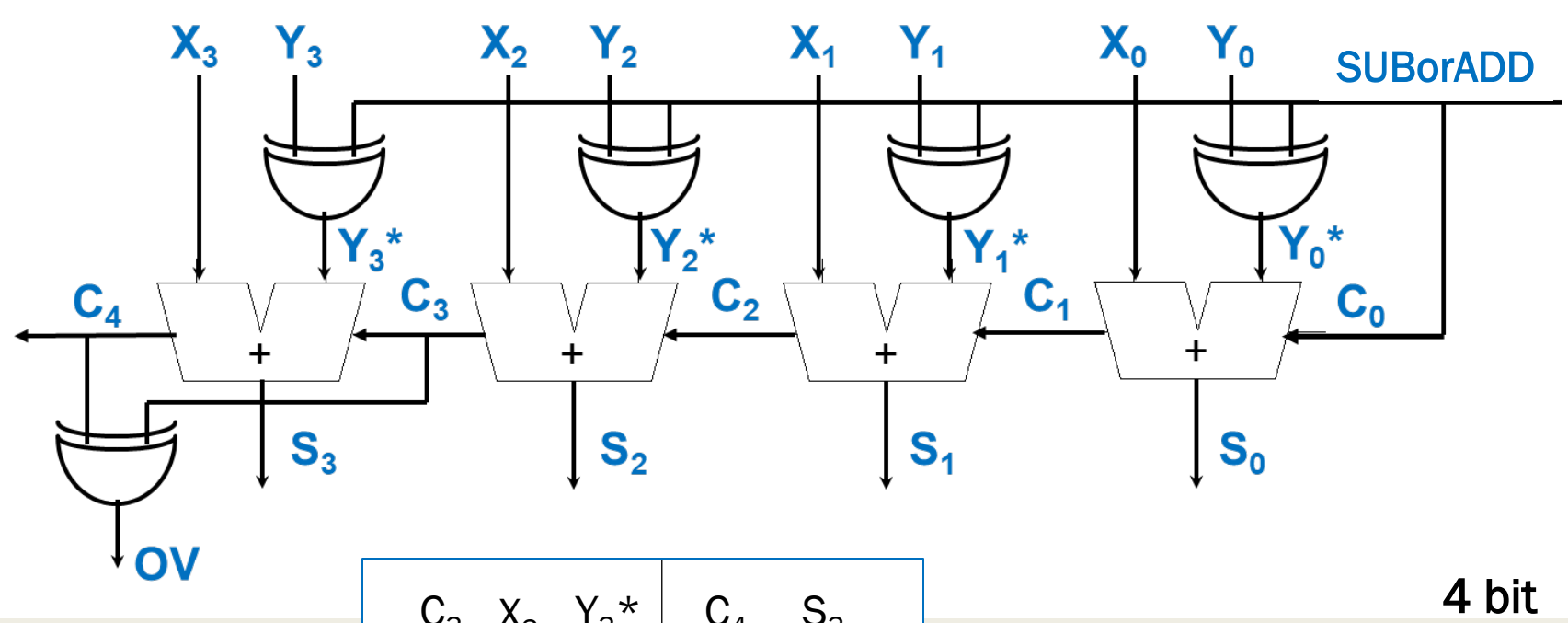
Αφαιρέτης (subtracter) των N bit

- 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

Αθροιστής/αφαιρέτης με επιλογή και υπερχείλιση προσημασμένων



OV

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

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

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

Πρόσθεση
SUBorADD = 0

$C_3\,C_2\,C_1\,C_0=0$

$X_3\,X_2\,X_1\,X_0$

$+\,Y_3\,Y_2\,Y_1\,Y_0$

$\not{C}_4\,S_3\,S_2\,S_1\,S_0$

Αφαίρεση
SUBorADD = 1

$C_3\,C_2\,C_1\,C_0=1$

$\overline{X_3}\,\overline{X_2}\,\overline{X_1}\,\overline{X_0}$

$+\,Y_3\,Y_2\,Y_1\,Y_0$

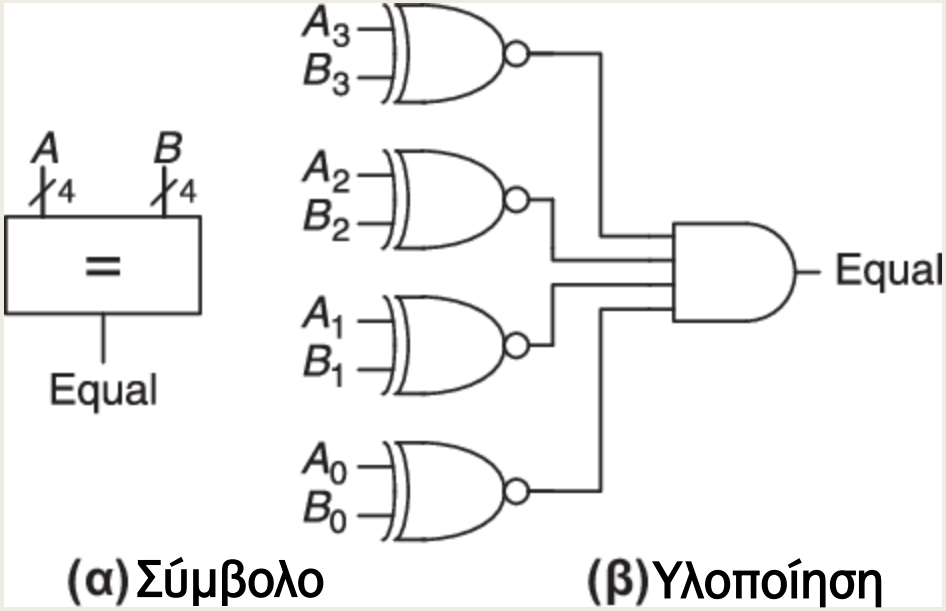
$\not{C}_4\,S_3\,S_2\,S_1\,S_0$

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*
    - Θα τον υλοποιήσουμε με την χρήση της Αριθμητικής και Λογικής Μονάδας (ALU)

Συγκριτής ισότητας των 4 bit

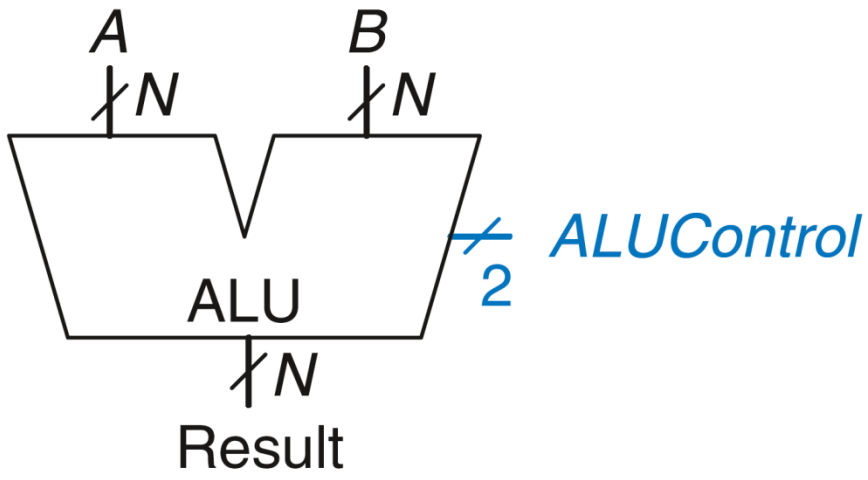
- Ο συγκριτής πρώτα χρησιμοποιεί πύλες XNOR για να ελέγξει αν τα αντίστοιχα bit σε κάθε στήλη των *A* και *B* είναι ίσα
- Οι αριθμοί είναι ίσοι αν τα αντίστοιχα bit σε όλες τις στήλες είναι ίσα
  - *Έστω* $A = (A_3, A_2, A_1, A_0)$ *και* $B = (B_3, B_2, B_1, B_0)$
  - ***Εάν*** $A_3 = B_3$ ***και*** $A_2 = B_2$ ***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 Α και Β είναι μεταξύ τους ίσοι



Αριθμητική/Λογική Μονάδα (ALU)

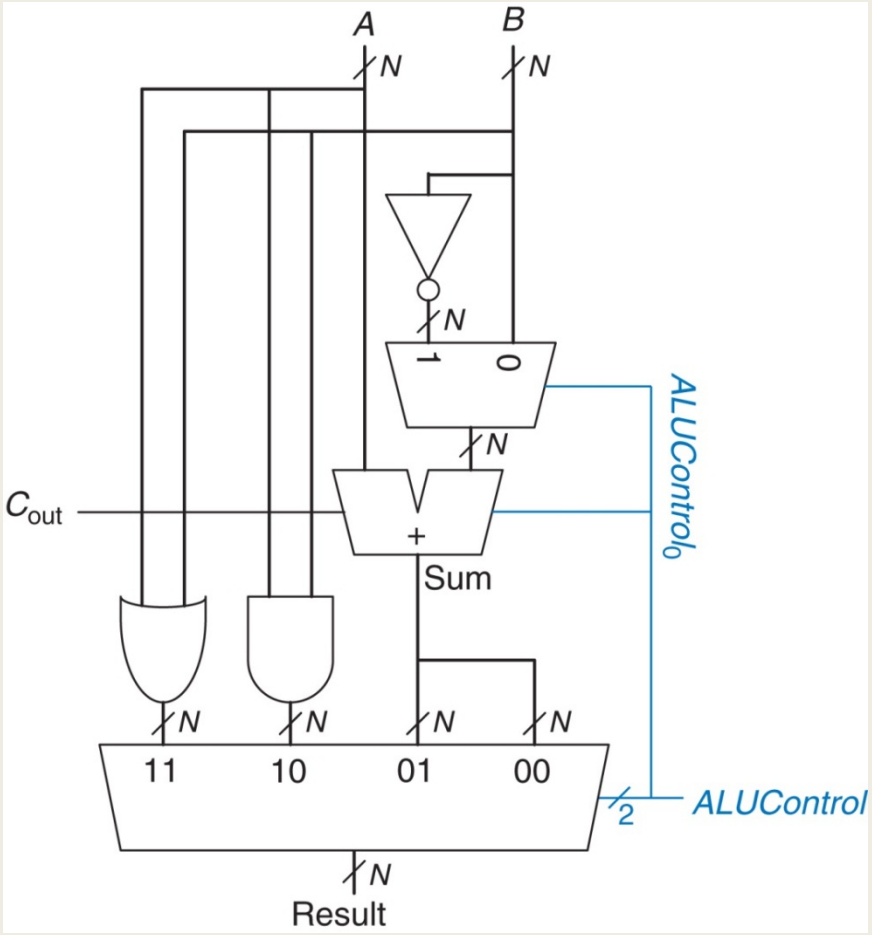
- Μια *Αριθμητική/Λογική Μονάδα* (Arithmetic/Logical Unit, ALU) υλοποιεί ποικίλες αριθμητικές και λογικές πράξεις σε μία μονάδα.
  - *Πρόσθεσης, αφαίρεσης προσημασμένων ακεραίων αριθμών*
  - *Λογικές πράξεις ανά bit (π.χ. AND και OR)*
- Η μονάδα ALU αποτελεί τον πυρήνα των περισσότερων υπολογιστικών 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

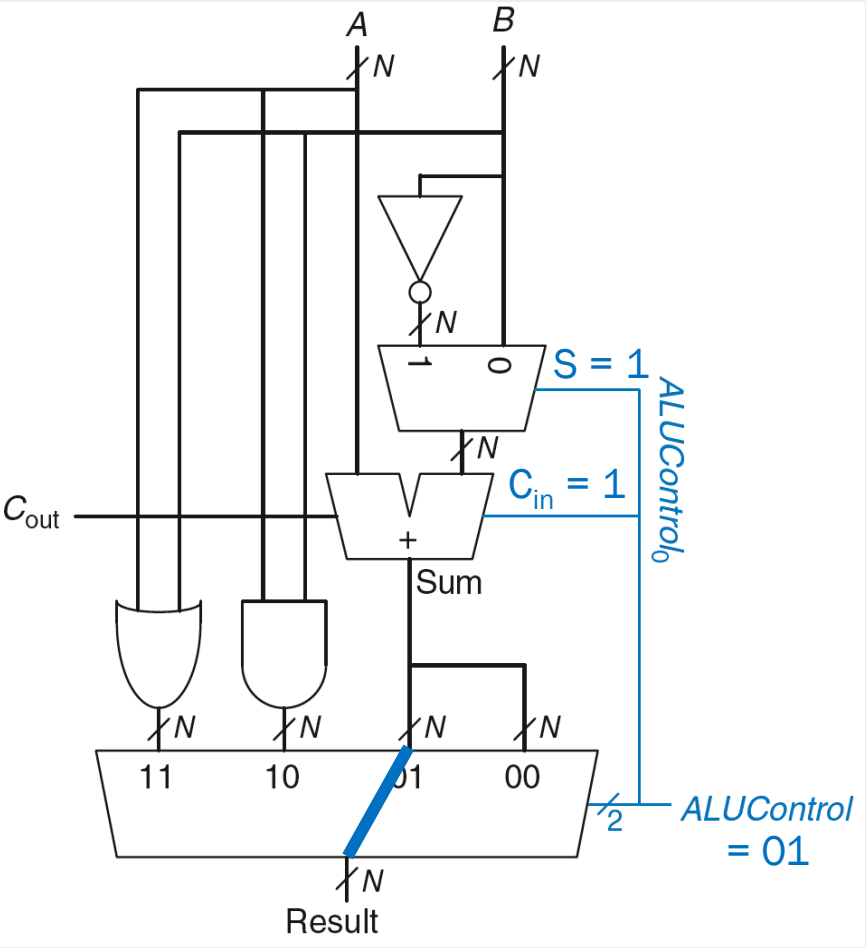
- Εσωτερικά η μονάδα ALU περιέχει:
  - Έναν ***αθροιστή των N bit*** με εισόδους A, B και έξοδο Sum των N bit
  - ***N πύλες AND*** με δύο εισόδους
  - ***N πύλες OR*** με δύο εισόδους
  - ***N αντιστροφείς και έναν πολυπλέκτη 2 σε 1 των N bit***, ώστε η είσοδος B του αθροιστή να αντιστρέφεται όταν ενεργοποιείται το ALUControl0
  - 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*

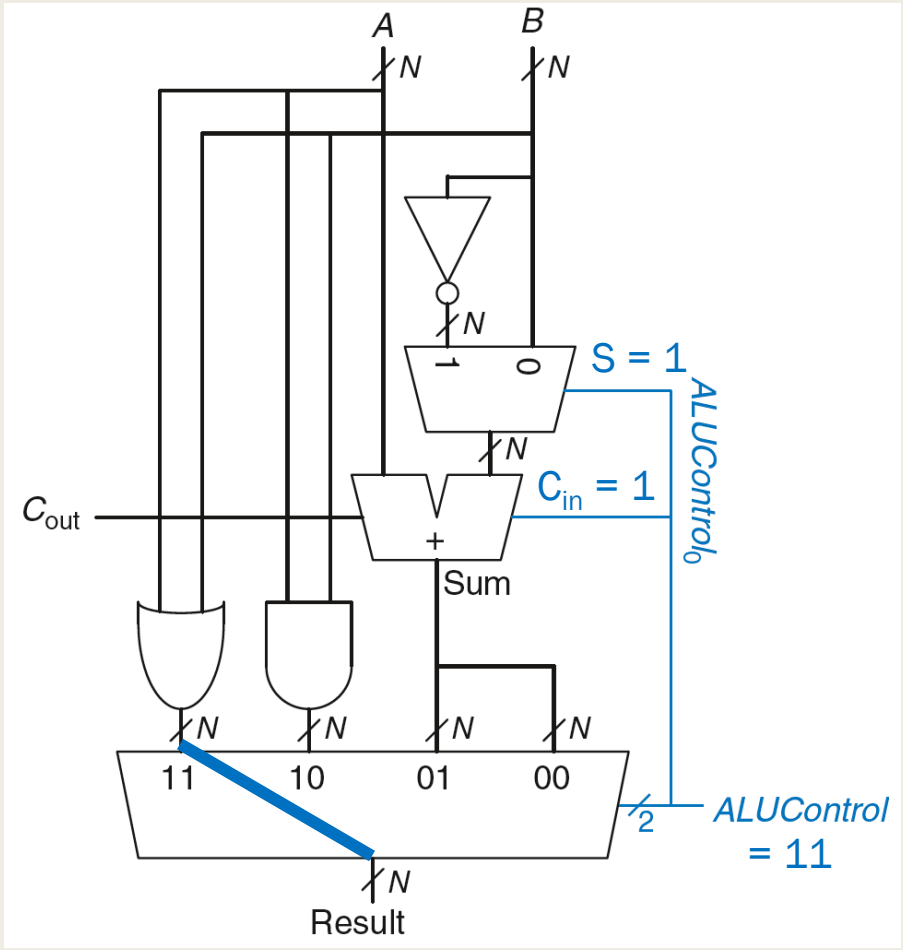
$ALUControl_{1:0}$	Πράξη
00	Add
01	Subtract
10	AND
11	OR



Υλοποίηση μίας απλής μονάδας ALU

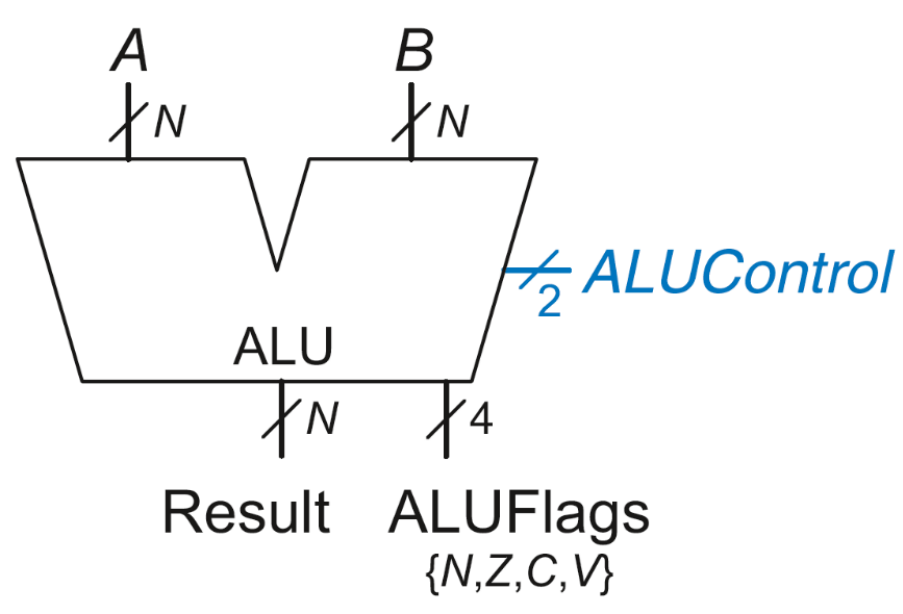
- Παράδειγμα: Εκτέλεση της λογικής πράξης **OR** (Result = A or B)
  - *ALUControl = 11 και $ALUControl_0$ = S = Cin = 1*
  - *Η είσοδος B του αθροιστή αντιστρέφεται*
  - *Ο αθροιστής εκτελεί την πράξη της αφαίρεσης*
  - *Ο πολυπλέκτης 4 σε 1 επιλέγει την είσοδο 11 ως έξοδο της ALU*

$ALUControl_{1:0}$	Πράξη
00	Add
01	Subtract
10	AND
11	OR



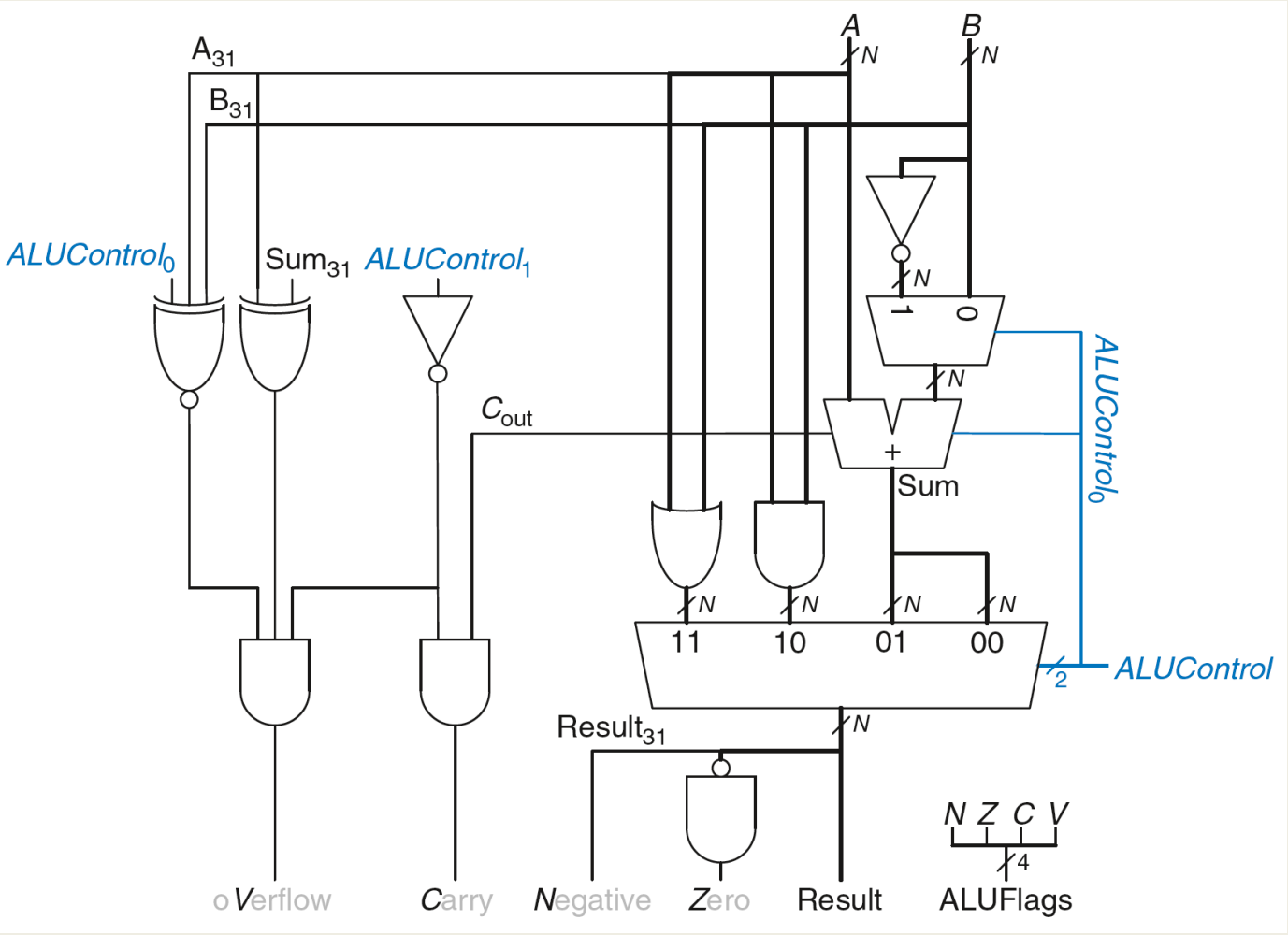
Σημαίες (ALU Status Flags)

Σημαία	Περιγραφή
N	Αρνητικό αποτέλεσμα (Negative)
Z	Μηδενικό αποτέλεσμα (Zero)
C	Ο αθροιστής παρήγαγε κρατούμενο εξόδου (Carry)
V	Ο αθροιστής υπερχείλισε (oVerflowed)



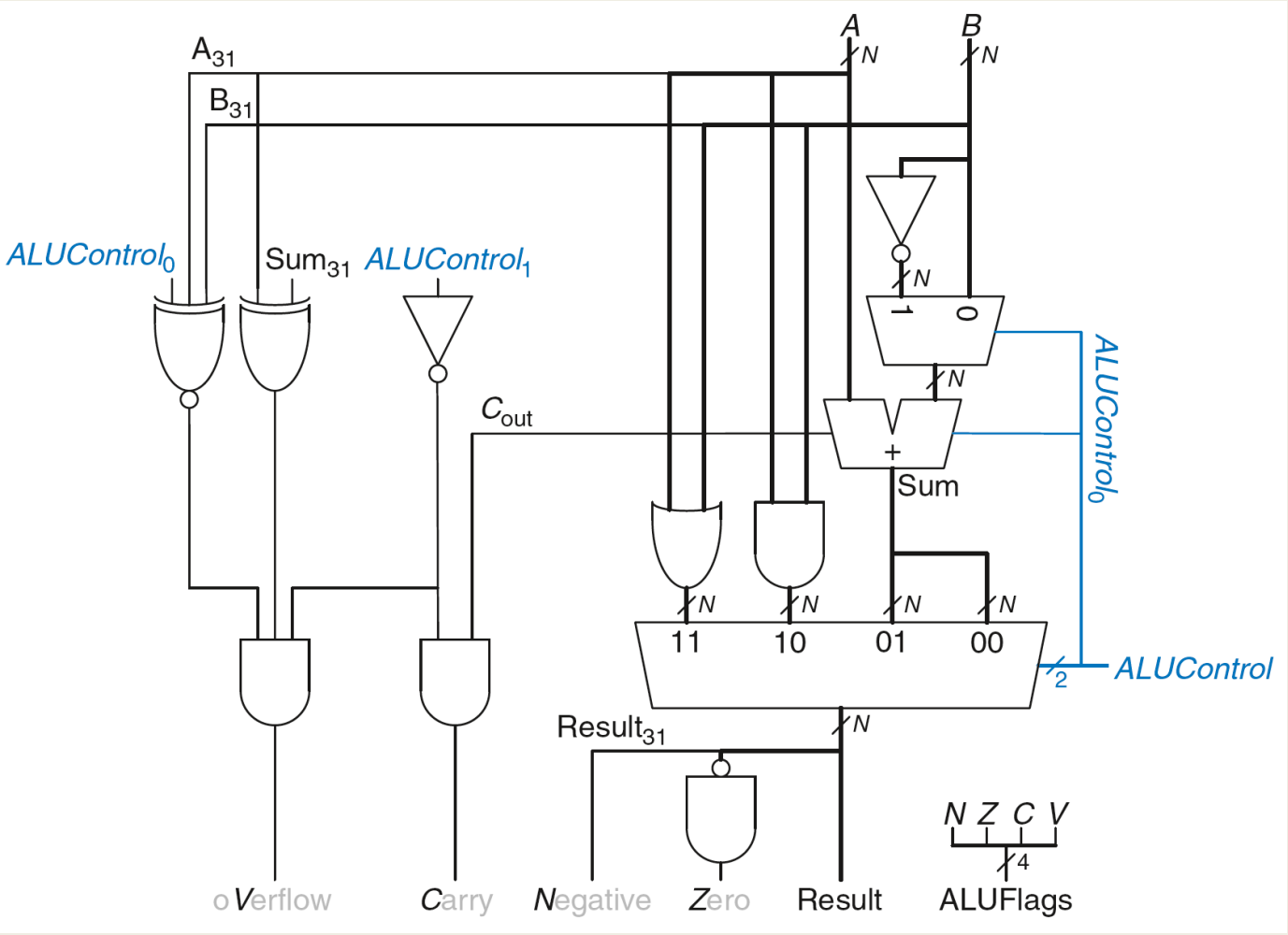
Σημαίες: Negative

- N = 1, εάν το αποτέλεσμα της μονάδας ALU είναι αρνητικό
- Το N είναι συνδεδεμένο με το πιο σημαντικό bit (MSB) του Result



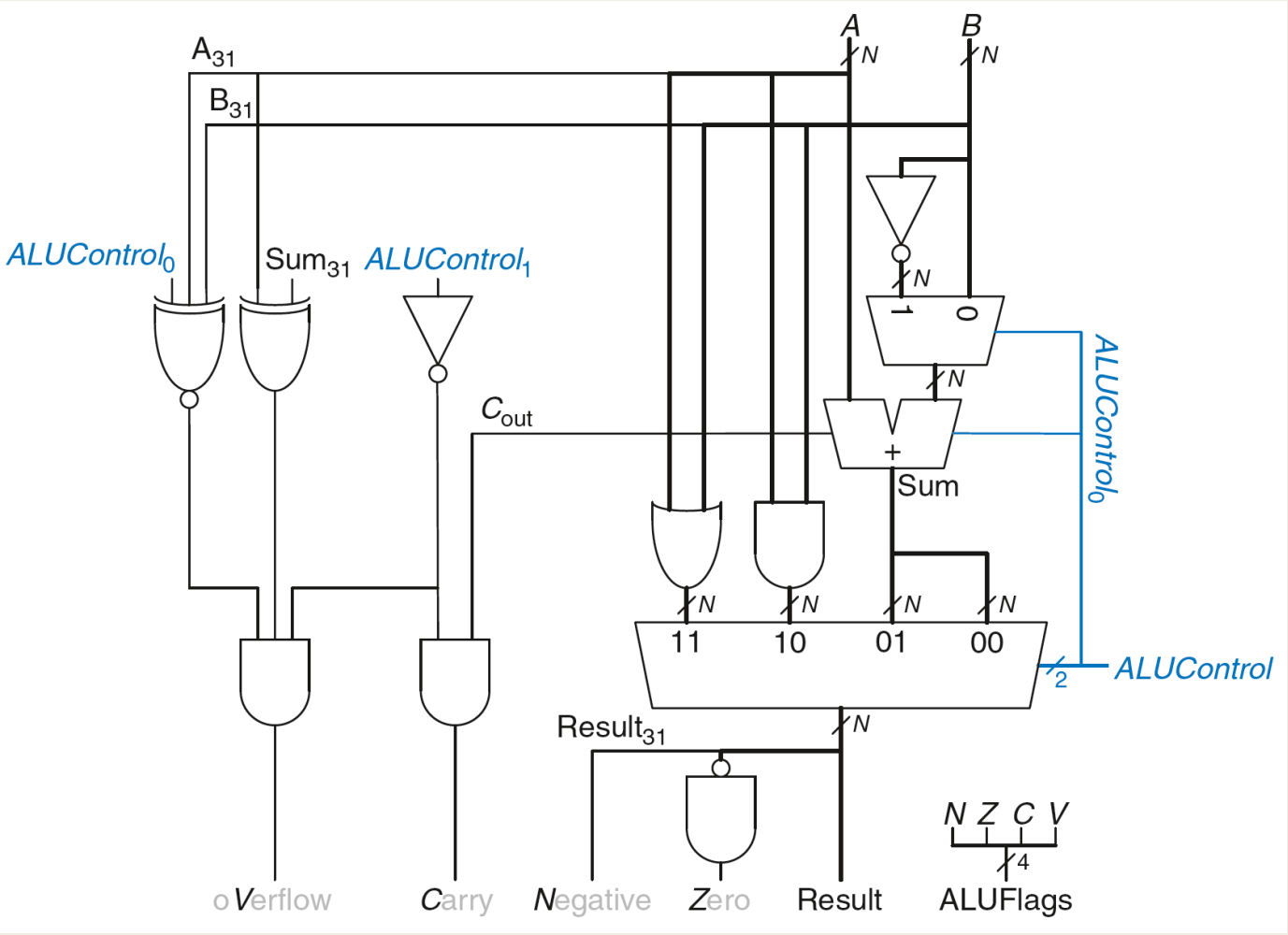
Σημαίες: Zero

- **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) ***ΚΑΙ*** η μονάδα ALU εκτελεί πρόσθεση ή αφαίρεση ($ALUControl_1 = 0$)

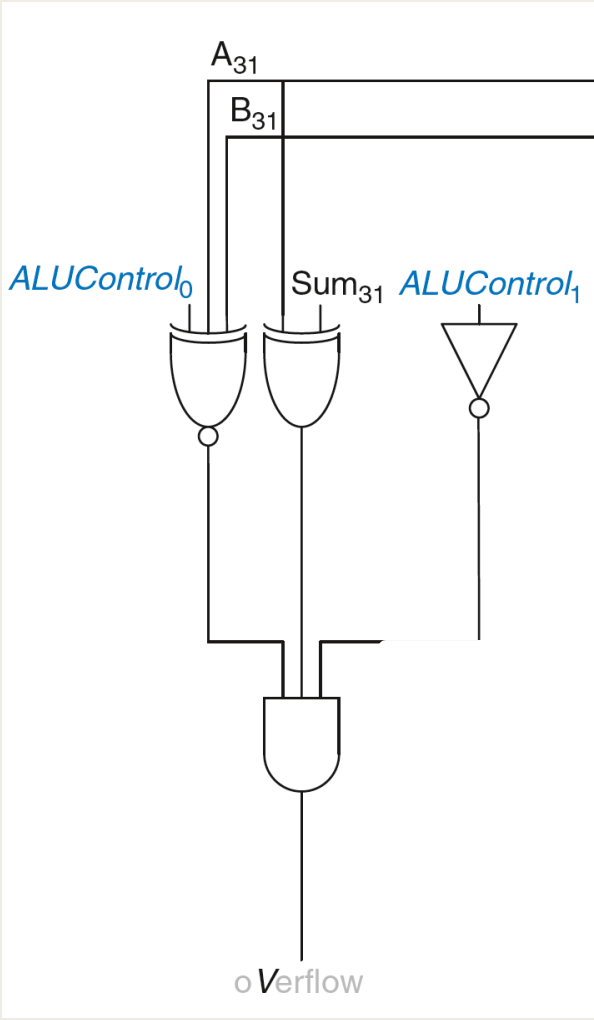


Σημαίες: oVerflow

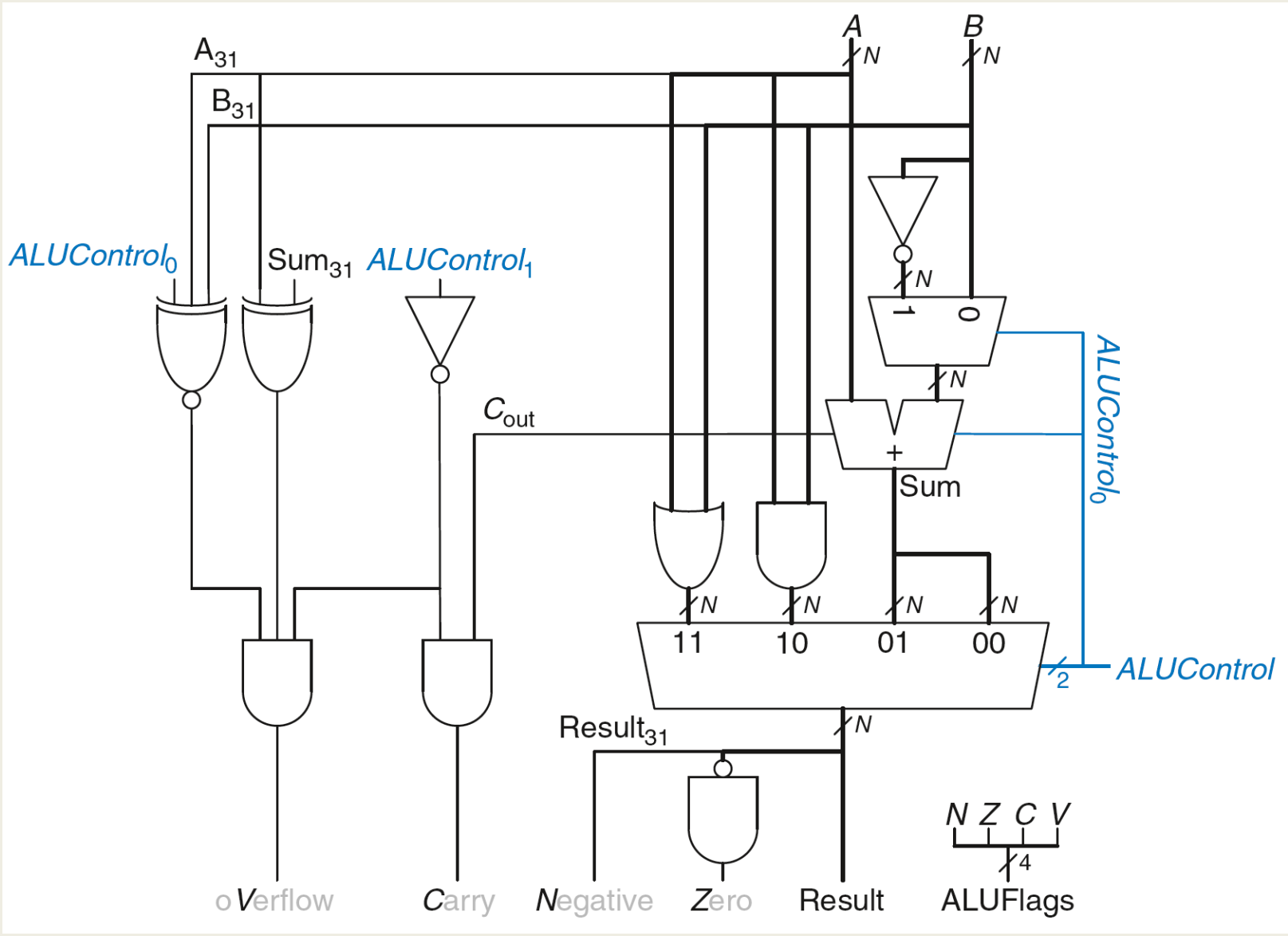
- V = 1, εάν το αποτέλεσμα της αριθμητικής πράξης ($ALUControl_1 = 0$) μεταξύ προσημασμένων αριθμών έχει υπερχειλίσει
  - *Η υπερχείλιση εμφανίζεται όταν δύο ομόσημοι προσημασμένοι αριθμοί παράγουν άθροισμα διαφορετικού προσήμου*
  - *Για να συμβεί υπερχείλιση ισχύουν οι συνθήκες:*
    - *Η μονάδα ALU εκτελεί πρόσθεση ή αφαίρεση* ($ALUControl_1 = 0$) **ΚΑΙ**
    - Α και Sum ετερόσημα ($A_{31} \neq Sum_{31}$) **ΚΑΙ**
    - Α και Β ομόσημα ($A_{31} = B_{31}$) στην πρόσθεση ($ALUControl_0 = 0$) ($A_{31}B_{31}Sum_{31} = 001, 110$)

      ή

      Α και Β ετερόσημα ($A_{31} \neq B_{31}$) στην αφαίρεση ($ALUControl_0 = 1$) ($A_{31}B_{31}Sum_{31} = 011, 100$)

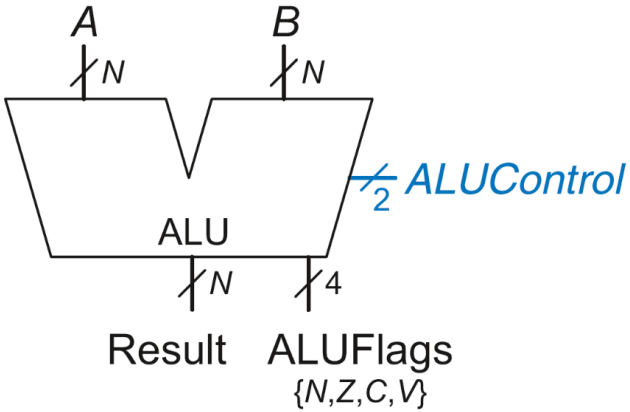


Σημαίες (ALU Status Flags)



Συγκριτής μεγέθους (A<B) των N bit

- Η σύγκριση μεγέθους για **προσημασμένους αριθμούς** συνήθως πραγματοποιείται με τον υπολογισμό του *A – B* στη μονάδα ALU και την εξέταση του **προσήμου N** (δηλαδή του περισσότερο σημαντικού bit) του αποτελέσματος (result)
  - *Αν το αποτέλεσμα είναι αρνητικό (N=1), τότε το A είναι μικρότερο από το B*
  - *Διαφορετικά, το A είναι μεγαλύτερο από ή ίσο με B (N=0)*
    - *Η ισότητα προκύπτει όταν το αποτέλεσμα είναι μηδέν (Z=1)*
  - *Προσοχή! Στην **υπερχείλιση (V=1)**. Αντιστρέφει το πρόσημο του αποτελέσματος και απαιτείται διόρθωση μετά την ανίχνευσή της*
    - *Αντί για N χρησιμοποιούμε το* N⊕V



Σημαία	Περιγραφή
N	Αρνητικό αποτέλεσμα (Negative)
Z	Μηδενικό αποτέλεσμα (Zero)
C	Ο αθροιστής παρήγαγε κρατούμενο εξόδου (Carry)
V	Ο αθροιστής υπερχείλισε (oVerflowed)

Συγκριτής Μεγέθους (A<B): Παράδειγμα*

- Θεωρείστε ότι οι προσημασμένοι αριθμοί είναι μεγέθους 4 bit
- Έστω A = 3, B = 5. Τότε $3_{10} - 5_{10}) = -2_{10}$ (χωρίς υπερχείλιση)
  - $3_{10} = 0011_2$ και $5_{10} = 0101_2$

```
   111 ← κρατούμενο εισόδου (Carry in)
    0011
+   1010 (αντεστραμμένο 0101=0x5)
    ----
    1110 = -2₁₀ => N=1, Z=0, C=0, V=0. (N⊕V)=1 => A<B
```

- Έστω A = –3, B = 6. Τότε $-3_{10} - 6_{10}) = -9_{10}$ **(με υπερχείλιση)**
  - $-3_{10} = 1101_2$ και $6_{10} = 0110_2$

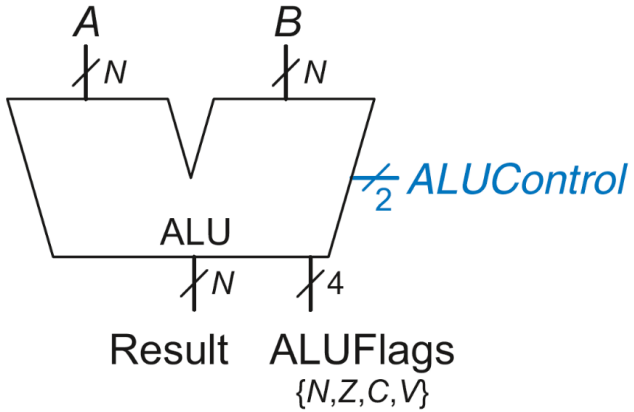
```
    11 ← κρατούμενο εισόδου (Carry in)
    1101
+   1001 (αντεστραμμένο 0110=0x6)
    ----
   ~1~0111 = 7₁₀ => N=0, Z=0, C=1, V=1. (N⊕V)=1 => A<B
```

Σημαία	Περιγραφή
N	Αρνητικό αποτέλεσμα (Negative)
Z	Μηδενικό αποτέλεσμα (Zero)
C	Ο αθροιστής παρήγαγε κρατούμενο εξόδου (Carry)
V	Ο αθροιστής υπερχείλισε (oVerflowed)

* Αντίστοιχο των ασκήσεων 5.9 και 5.10

Συγκριτής μεγέθους (A<B) των N bit

- Η σύγκριση μεγέθους για **μη προσημασμένους αριθμούς** συνήθως πραγματοποιείται με τον υπολογισμό του *A – B* στη μονάδα ALU και την εξέταση του **κρατούμενου εξόδου C**
  - *Αν η πράξη δεν παράγει κρατούμενο εξόδου (C=0), τότε το A είναι μικρότερο από το B*
  - *Διαφορετικά, το A είναι μεγαλύτερο από ή ίσο με B (C=1)*
    - *Η ισότητα προκύπτει όταν το αποτέλεσμα είναι μηδέν (Z=1)*
  - *Ισχύει ότι:*
    - $A - B = A + (-B) = A + \overline{B} + 1$
    - $B + \overline{B} = 2^N - 1 \Rightarrow \overline{B} + 1 = 2^N - B$
    - $A + (-B) = 2^N + A - B$
    - *C = 0, εάν* $A + (-B) < 2^N \Rightarrow A < B$



Σημαία	Περιγραφή
N	Αρνητικό αποτέλεσμα (Negative)
Z	Μηδενικό αποτέλεσμα (Zero)
C	Ο αθροιστής παρήγαγε κρατούμενο εξόδου (Carry)
V	Ο αθροιστής υπερχείλισε (oVerflowed)

Συγκριτής Μεγέθους (A<B): Παράδειγμα*

- Εστω δύο αριθμοί είναι μεγέθους 4 bit: A = 1111_2 και B = 0010_2
- Εξετάστε αν A<B, εάν οι αριθμοί είναι **προσημασμένοι**
  - *A = -1_{10}, B = 2_{10}. Τότε $-1_{10} - 2_{10}) = -3_{10}$ (χωρίς υπερχείλιση)*

```
   1111 ← κρατούμενο εισόδου (Carry in)
   1111
 + 1101 (αντεστραμμένο 0010=0x2)
 ----
  11101 = -3_10 => N=1, Z=0, C=1, V=0. (N⊕V)=1 => A<B
```

- Εξετάστε αν A<B, εάν οι αριθμοί είναι **μη προσημασμένοι**
  - *A = 15_{10}, B = 2_{10}. Τότε $15_{10} - 2_{10}) = 13_{10}$ (χωρίς υπερχείλιση)*

```
   1111 ← κρατούμενο εισόδου (Carry in)
   1111
 + 1101 (αντεστραμμένο 0010=0x2)
 ----
  11101 = 13_10 => N=1, Z=0, C=1, V=0. C=1, Z=0 => A>B
```

Σημαία	Περιγραφή
N	Αρνητικό αποτέλεσμα (Negative)
Z	Μηδενικό αποτέλεσμα (Zero)
C	Ο αθροιστής παρήγαγε κρατούμενο εξόδου (Carry)
V	Ο αθροιστής υπερχείλισε (oVerflowed)

* Αντίστοιχο του Παραδείγματος 5.3

Συγκρίσεις με βάση τις σημαίες

- Η σύγκριση γίνεται στην μονάδα ALU με αφαίρεση και εξέταση των ALU flags
- Είναι διαφορετική για προσημασμένους και μη προσημασμένους αριθμούς

Comparison	Signed	Unsigned
$=$	Z	Z
$\neq$	$\overline{Z}$	$\overline{Z}$
$<$	$N \oplus V$	$\overline{C}$
$\leq$	$Z + (N \oplus V)$	$Z + \overline{C}$
$>$	$\overline{Z} \bullet (\overline{N \oplus V})$	$\overline{Z} \bullet C$
$\geq$	$(\overline{N \oplus V})$	C

Σημαία	Περιγραφή
N	Αρνητικό αποτέλεσμα (Negative)
Z	Μηδενικό αποτέλεσμα (Zero)
C	Ο αθροιστής παρήγαγε κρατούμενο εξόδου (Carry)
V	Ο αθροιστής υπερχείλισε (oVerflowed)

Υλοποίηση της εντολής SLT στην ALU

- Σε μερικές αρχιτεκτονικές RISC υπολογιστών (MIPS, RISC-V) δεν αποθηκεύονται οι σημαίες N,Z,C,V, αλλά χρησιμοποιούνται εντολές, όπως η εντολή **SLT (Set if less than)** για περισσότερη ευελιξία
  - *Εκτελείται η **πράξη SLT** στην **μονάδα ALU με αφαίρεση***
    - **IF A<B then Result = 1 (N⊕V)=1, else Result =0 (N⊕V)=0**
  - *Το Result αποθηκεύεται στο αρχείο καταχωρητών.*
  - *Η σύγκριση γίνεται συνήθως με **προσημασμένους αριθμούς***
- Η υλοποίηση της πράξης SLT, όπως και κάθε άλλης πράξης (π.χ. XOR), απαιτεί επέκταση του μεγέθους του σήματος ελέγχου **ALUControl**

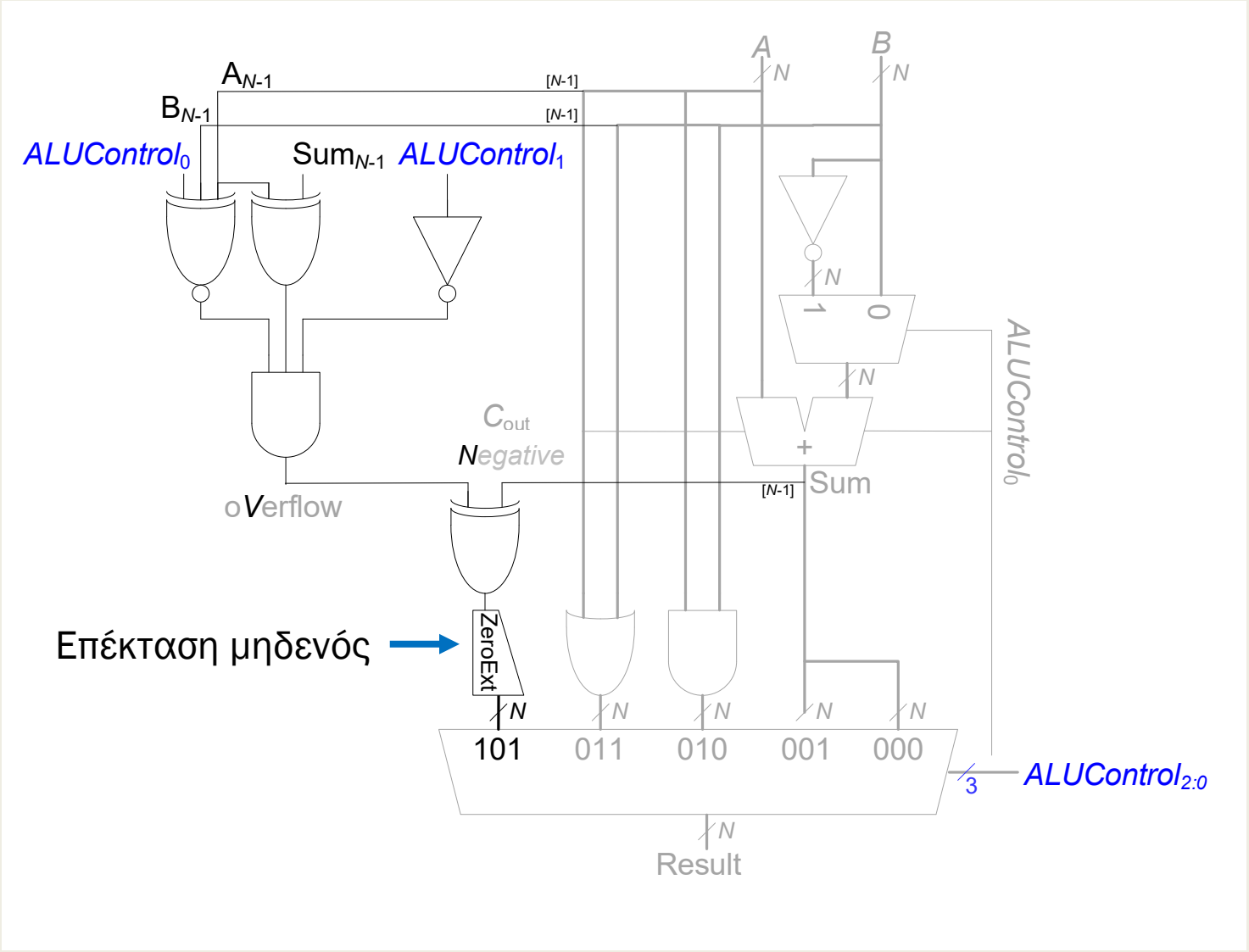
$ALUControl_{2:0}$	Πράξη
000	Add
001	Subtract
010	AND
011	OR
101	**SLT**

Το ALUControl2 = 1 δηλώνει την πράξη SLT

Το ALUControl0 = 1 απαιτείται για την αφαίρεση A-B

Η νέα μονάδα ALU

- Υποστηρίζει την πράξη SLT για προσημασμένους αριθμούς
- Διορθώνει και το λάθος λόγω υπερχείλισης, **(N⊕V) αντί N**



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)

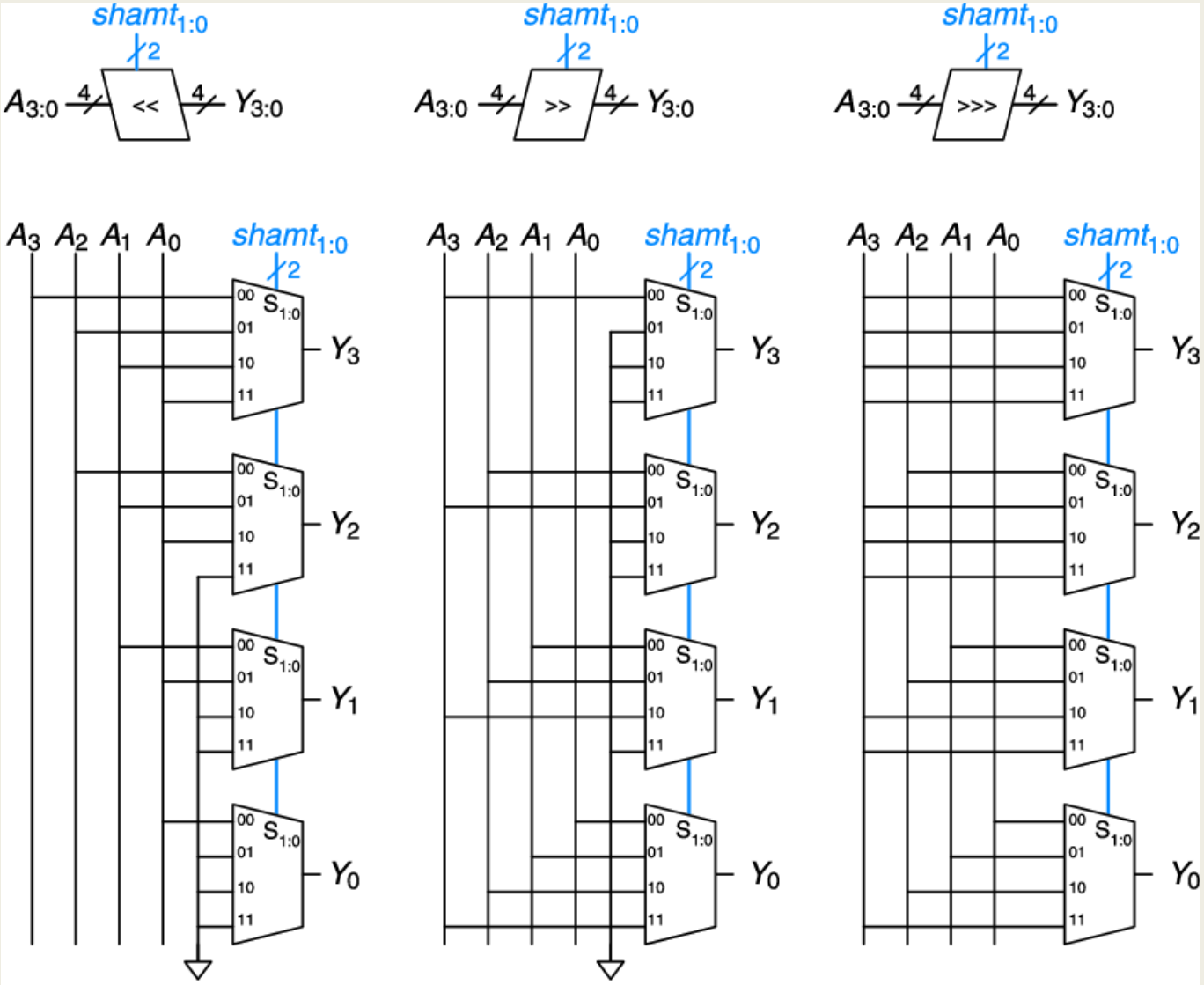
- **Λογικός ολισθητής**
  - *Ολισθαίνει τον αριθμό προς τα αριστερά (logical shift left, LSL, <<) και συμπληρώνει τις κενές θέσεις με **μηδενικά***
    - 11001 << 2 => 00100
  - *Ολισθαίνει τον αριθμό προς τα δεξιά (logical shift right, LSR, >>) και συμπληρώνει τις κενές θέσεις με **μηδενικά***
    - 11001 >> 2 => 00110
- **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 => 11110 (διαιρεί δια 4, από -8 σε -2)

$$A << N = A \times 2^N$$

$$A >>> N = A \div 2^N$$

Ολισθητές (shifters)

- Ένας ολισθητής των *N* bit υλοποιείται με *N* πολυπλέκτες Ν σε 1 (π.χ. Ν = 4)

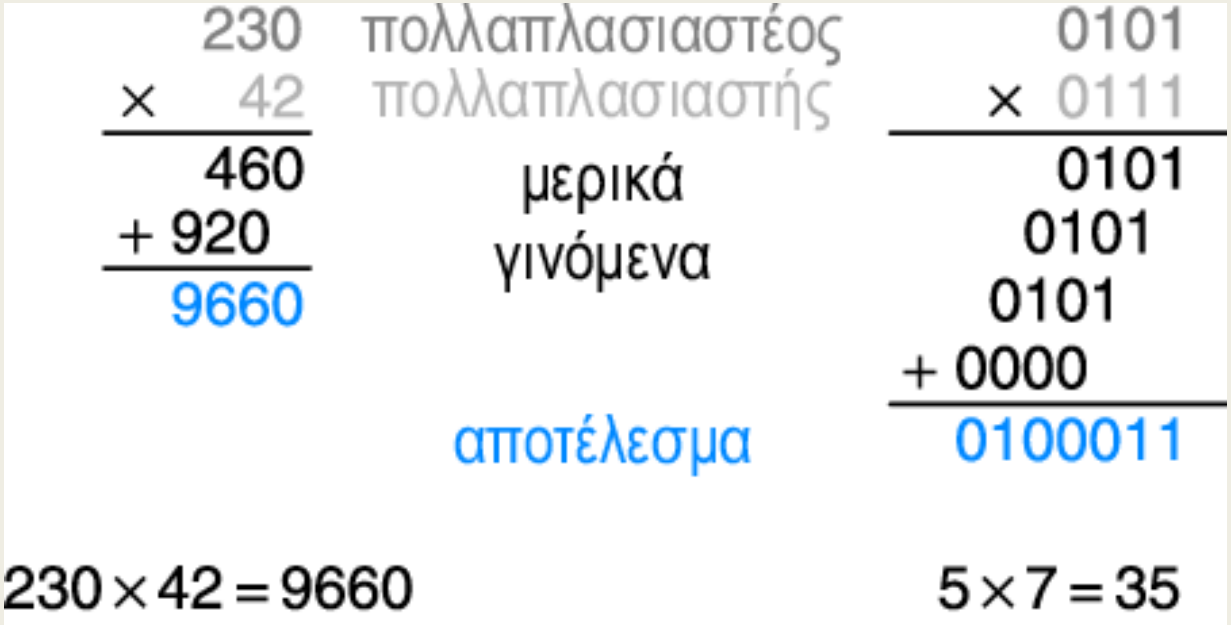


Περιστροφείς (rotators)

- **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

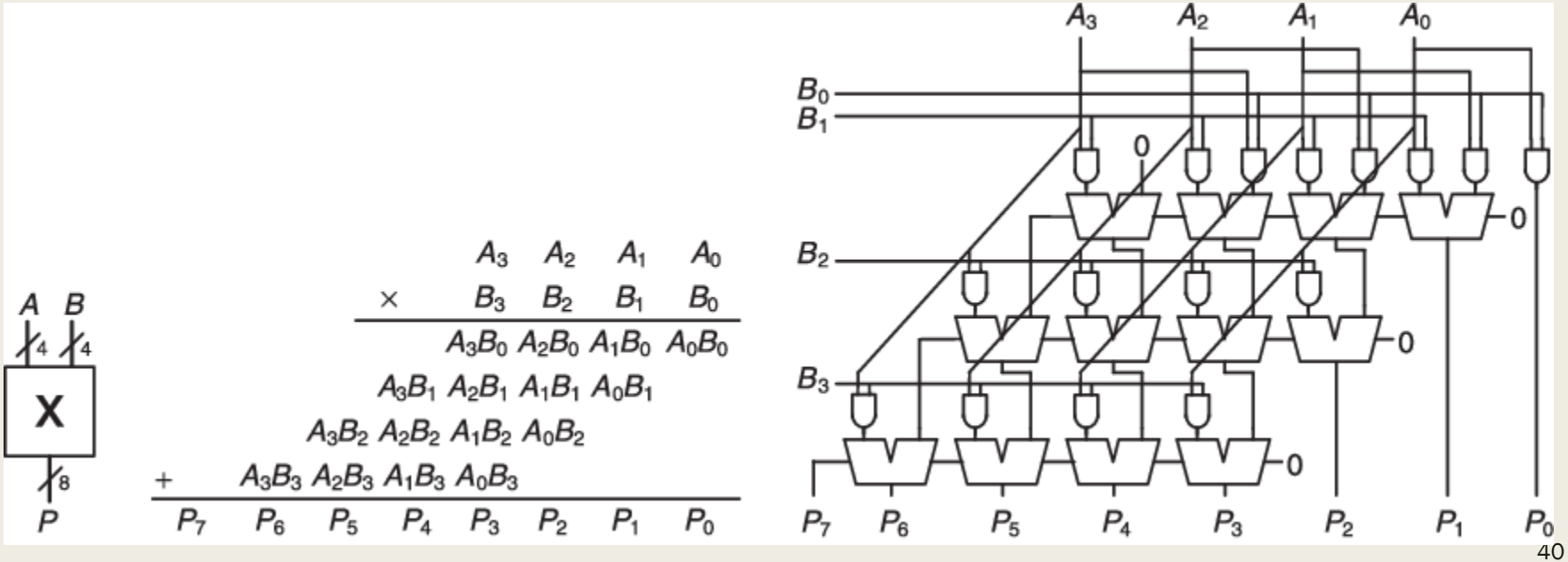
Δυαδικός πολλαπλασιασμός

- 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*



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 × 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 $\Rightarrow$ 0110.1100 =

    $0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 0\times2^{-4}$ =

    $4 + 2 + 0.50 + 0.25 = 6.75$
- Αναπαράσταση σταθερής υποδιαστολής **προσημασμένου πραγματικού** αριθμού σε αναπαράσταση συμπληρώματος ως προς δύο με 4 bit ακέραιο μέρος και 4 bit κλασματικό μέρος
  - 11011010 $\Rightarrow$ 1101.1010 =

    $1\times-2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 0\times2^{-4}$ =

    $-8 + 4 + 1 + 0.500 + 0.125 = -8 + 5.625 = -2.375$

Πραγματικοί αριθμοί σταθερής υποδιαστολής

- Συμβολισμός των **μη προσημασμένων αριθμών**: Ua.b
  - *a = μέγεθος ακεραίου μέρους*
  - *b = μέγεθος κλασματικού μέρους*
- Παράδειγμα: **6,75**
  - U4.4: 0110.1100 => 01101100
  - U3.5: 110.11000 => 11011000
  - U6.2: 000110.11 => 00011011

Το U8.8 χρησιμοποιείται σε δεδομένα αισθητήρων, στο audio και στο video

Το U16.16 χρησιμοποιείται στην επεξεργασίας σήματος με υψηλή ακρίβεια

Προσοχή στην υπερχείλιση, που π.χ. παράγει artifacts στο video:
ένα σκοτεινό pixel ανάμεσα σε φωτεινά pixel

Πραγματικοί αριθμοί σταθερής υποδιαστολής

- Συμβολισμός των **προσημασμένων αριθμών**: Qa.b
  - *a = μέγεθος ακεραίου μέρους*
  - *b = μέγεθος κλασματικού μέρους*
- Χρησιμοποιείται η αναπαράσταση συμπληρώματος ως προς δύο
- Εύρεση του **-6,75** στο Q4.4
  - **Αριθμός 6,75:** 0110.1100 => 01101100
  - **Αντιστροφή bit:** 0110.1100 => 1001.0011
  - **Πρόσθεσε το 1 στο LSB:** 1001.0011 => 1001.0100
  - **Αριθμός -6,75** 1001.0100 => 10010100

-7 +0,25

Το Q1.15 χρησιμοποιείται στην επεξεργασίας σήματος με υψηλή ακρίβεια για την περιοχή τιμών [-1,1]

Πραγματικοί αριθμοί σταθερής υποδιαστολής

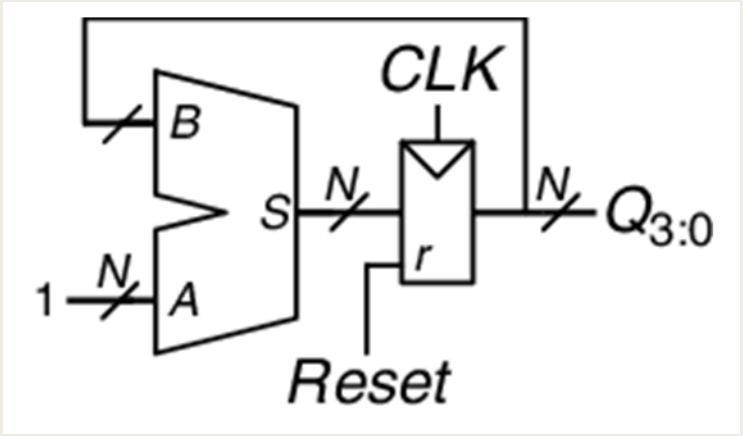
- Οι επεξεργαστές χειρίζονται τους πραγματικούς αριθμούς σταθερής υποδιαστολής όπως χειρίζονται τους ακεραίους
  - λόγω της απλότητας των πράξεων, οι *πραγματικοί αριθμοί σταθερής υποδιαστολής* χρησιμοποιούνται σε υλοποίηση αλγορίθμων στο υλικό, όπου απαιτούνται πραγματικοί αριθμοί
    - ιδιαίτερα σε 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
- Παράδειγμα πράξης: 0,750 – 0,625 στο Q4.4*

```
  1111 1  11 ← κρατούμενο εισόδου (Carry in)
  0000.1100 (0,750)
+ 1111.0101 (αντεστραμμένο 0000.1010=0,625)
-------------
 10000.0010 = 0.125_10 => N=0, Z=0, C=1, V=0
```

* Αντίστοιχο Παράδειγμα 5.5

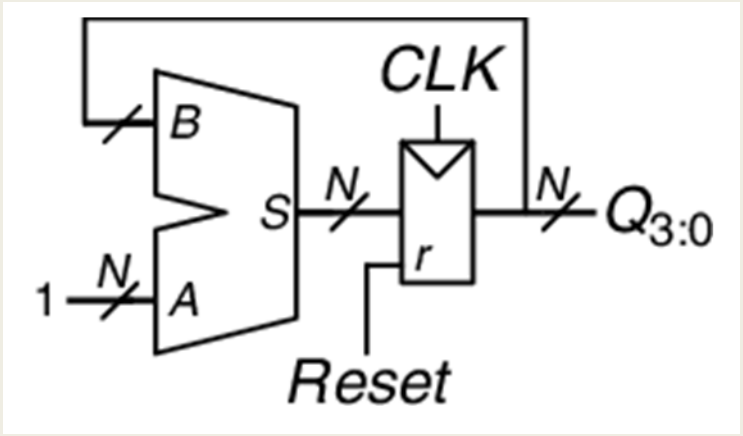
Μετρητής

- 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. N=3: 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)*



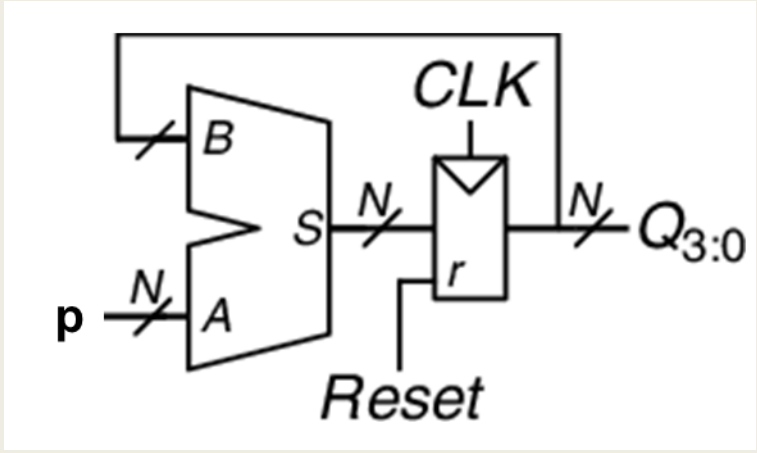
Μετρητής και διαίρεση συχνότητας

- Το MSB ενός **δυαδικού μετρητή των *N* bit** αλλάζει τιμή από 0 σε 1 και από 1 σε 0 κάθε 2^{N-1} κύκλους
  - *Στο MSB η συχνότητα του ρολογιού CLK μειώνεται κατά 2^N φορές*
  - *Ονομάζεται και **μετρητης διαίρεσης δια του 2^N (divide by 2^N counter)***
    - Π.χ. N=3: 000, 001, 010, 011, 100, 101, 110, 111, 000, …
    - Στο MSB η συχνότητα του ρολογιού CLK μειώνεται κατά 8 φορές
- Ο **μετρητής διαίρεσης δια του 2^N** χρησιμοποιείται για να επιβραδύνουμε γρήγορα σήματα ρολογιού CLK κατά **2^N φορές**
  - *Παράδειγμα: Πόσο μπορεί να μειωθεί η συχνότητα ενός σήματος CLK των 50 MHz με έναν μετρητή διαίρεσης των 24 bit?*
    - Κατά 2^{24} στο MSB του μετρητή
    - $f_{MSB} = f_{CLK} / 2^{24}$
    - $f_{MSB} = 50 \times 10^6 \text{ Hz} / 2^{24} = 2{,}98 \text{ Hz}$



Ψηφιακά ελεγχόμενος ταλαντωτής

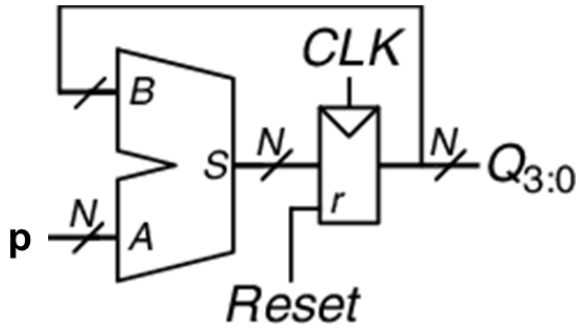
- Ο **ψηφιακά ελεγχόμενος ταλαντωτής (digitally controlled oscillator, DCO)** είναι μία γενίκευση του **μετρητή διαίρεσης δια του 2^N** προσθέτοντας την ποσότητα **p αντί για 1** σε κάθε κύκλο
  - *Υλοποιείται με αθροιστή των N bit με τη **μία είσοδο σταθερή στο p***
  - *Μειώνει την συχνότητα του ρολογιού CLK **κατά p / 2^N***
    - $f_{MSB} = f_{CLK} \times p / 2^N \Rightarrow p = f_{MSB} \times 2^N / f_{CLK}$
  - *Όσο μεγαλύτερο είναι το N στο μετρητή διαίρεσης δια του 2^N τόσο πιο ακριβής είναι η μείωση συχνότητας*
- Παράδειγμα 5.7
  - *Έχετε ένα ρολόι με συχνότητα 50 MHz και θέλετε να παράγετε έξοδο με συχνότητα 500 Hz.*
  - *Μπορείτε να χρησιμοποιήσετε έναν μετρητή διαίρεσης των 24 ή 32 bit.*
  - *Ποια τιμή πρέπει να επιλέξετε για το p, και πόσο κοντά μπορείτε να πλησιάσετε στα 500 Hz;*



Ψηφιακά ελεγχόμενος ταλαντωτής

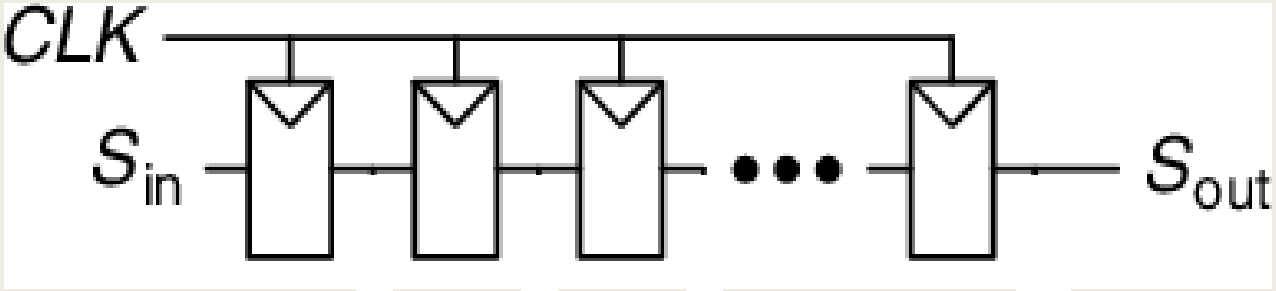
- Παράδειγμα 5.7
  - *Έχετε ένα ρολόι με συχνότητα 50 MHz και θέλετε να παράγετε έξοδο με συχνότητα 500 Hz.*
  - *Μπορείτε να χρησιμοποιήσετε έναν μετρητή διαίρεσης των 24 ή 32 bit.*
  - *Ποια τιμή πρέπει να επιλέξετε για το p, και πόσο κοντά μπορείτε να πλησιάσετε στα 500 Hz;*
- Λύση
  - $p_{24} = 500\ Hz \times 2^{24} / 50 \times 10^6\ Hz = 167{,}77 = 168$
  - $p_{32} = 500\ Hz \times 2^{32} / 50 \times 10^6\ Hz = 42949{,}67 = 42.950$
  - $f_{24} = 50 \times 10^6\ Hz \times 168 / 2^{24} = 500{,}679\ Hz$
  - $f_{32} = 50 \times 10^6\ Hz \times 42.950 / 2^{32} = 500{,}0038\ Hz$

$$f_{MSB} = f_{CLK} \times p / 2^N \Rightarrow p = f_{MSB} \times 2^N / f_{CLK}$$



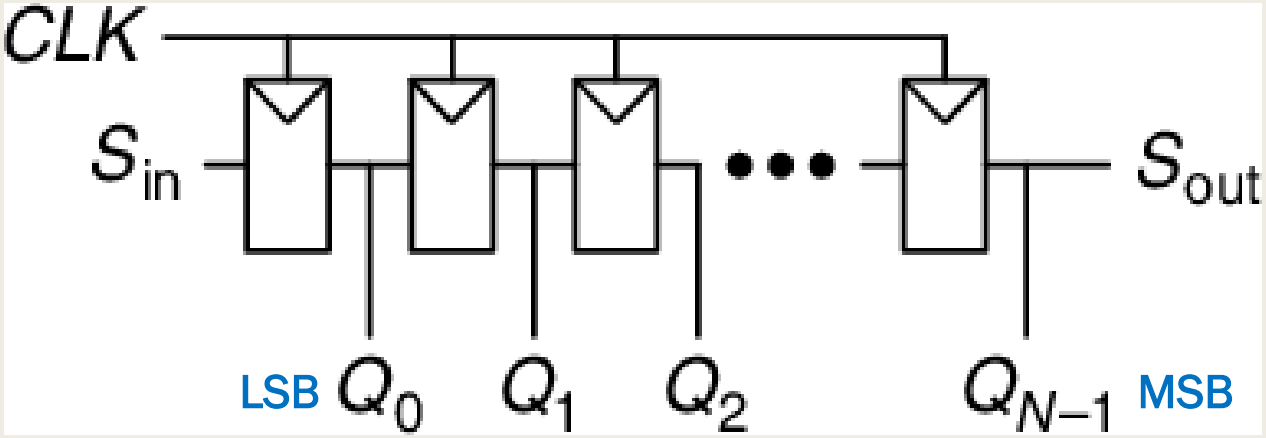
Καταχωρητής ολίσθησης

- 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



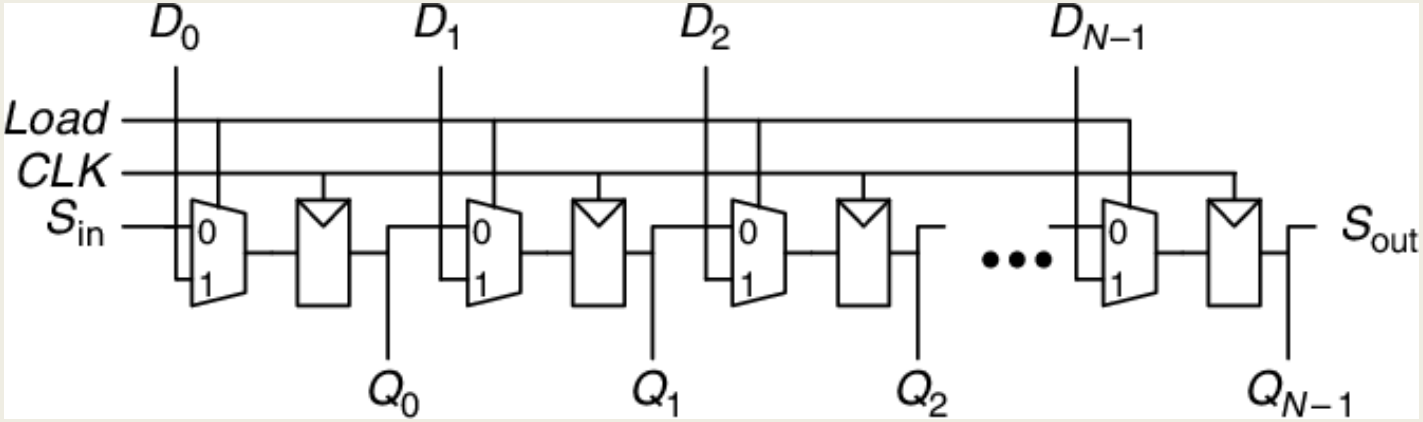
Καταχωρητής ολίσθησης

- 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 ολισθαίνει 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*

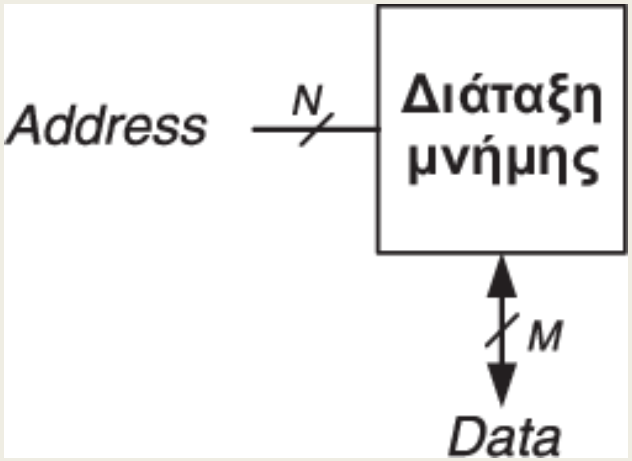


Διατάξεις μνήμης (memory arrays)

- 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)*

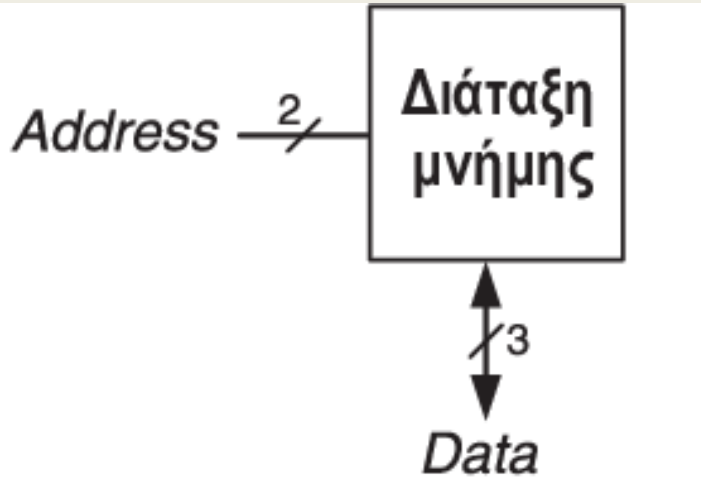
Διατάξεις μνήμης

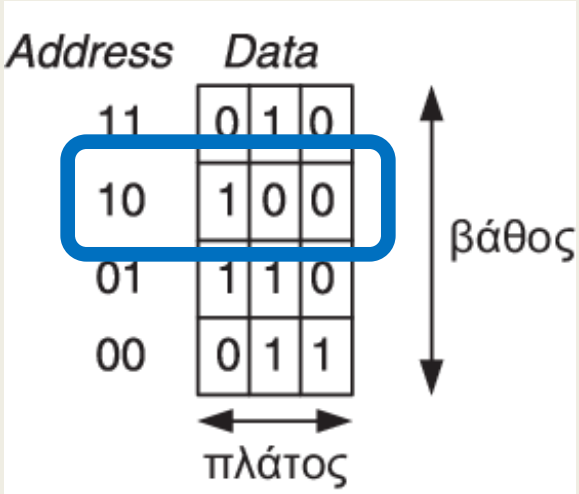
- Η μνήμη είναι οργανωμένη ως μια διδιάστατη **διάταξη κελιών μνήμης** (memory cells)
- Η μνήμη **διαβάζει** ή **εγγράφει** τα περιεχόμενα μίας από τις σειρές (rows) της διάταξης μνήμης, που ονομάζονται **δεδομένα** (data)
- Η σειρά της διάταξης μνήμης καθορίζεται από μια **διεύθυνση** (address)
- Μια 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

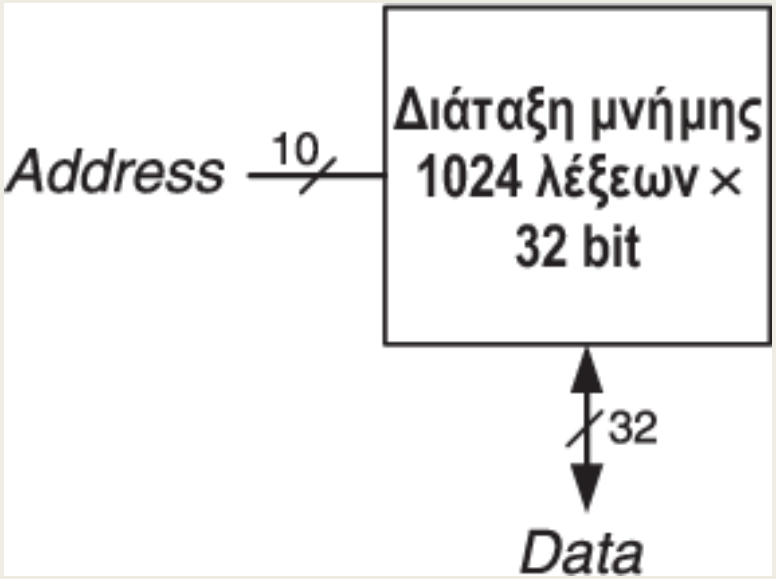
- Στην εικόνα μπορείτε να δείτε μια διάταξη μνήμης με δύο bit διεύθυνσης και τρία bit δεδομένων
  - *Τα δύο bit της διεύθυνσης (**N = 2**) καθορίζουν μία από τις τέσσερις σειρές (**$2^N = 4$**) της διάταξης μνήμης*
  - *Σε κάθε σειρά αποθηκεύεται μία λέξη δεδομένων που έχει μέγεθος **M = 3 bit***
  - *Παράδειγμα:* η λέξη των 3 bit που αποθηκεύεται στη διεύθυνση 10_2 είναι η 100_2
- Μέγεθος μιας διάταξης μνήμης (array size): **βάθος × πλάτος** (= 4 × 3 = 12)
  - ***Βάθος (depth)**: το πλήθος των σειρών (των λέξεων δεδομένων)*
  - ***Πλάτος (width)**: το πλήθος των στηλών (μέγεθος της λέξης δεδομένων)*





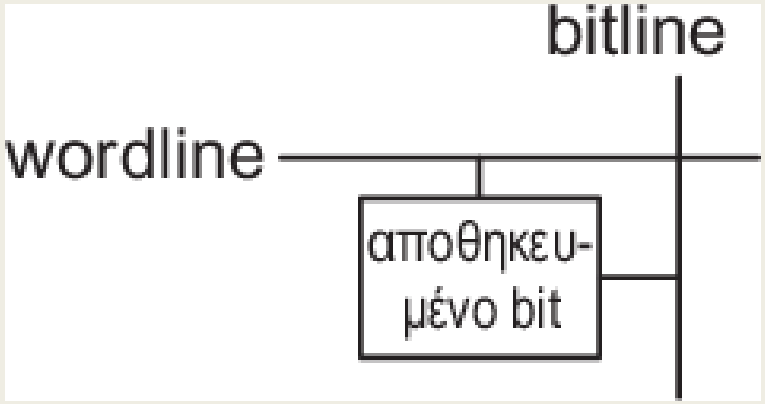
Παράδειγμα διάταξης μνήμης

- Στην εικόνα μπορείτε να δείτε το σύμβολο για μια διάταξη μνήμης 1024 λέξεων × 32 bit
- Το συνολικό μέγεθος αυτής της διάταξης μνήμης είναι ίσο με 32 kilobit (Kb)



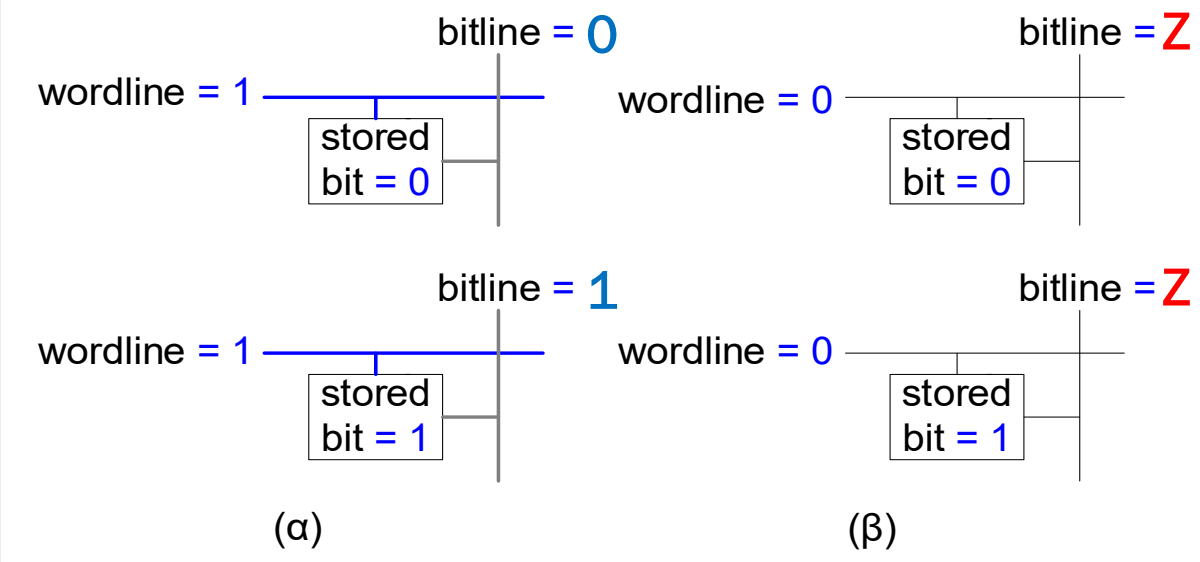
Κελιά μνήμης

- 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*



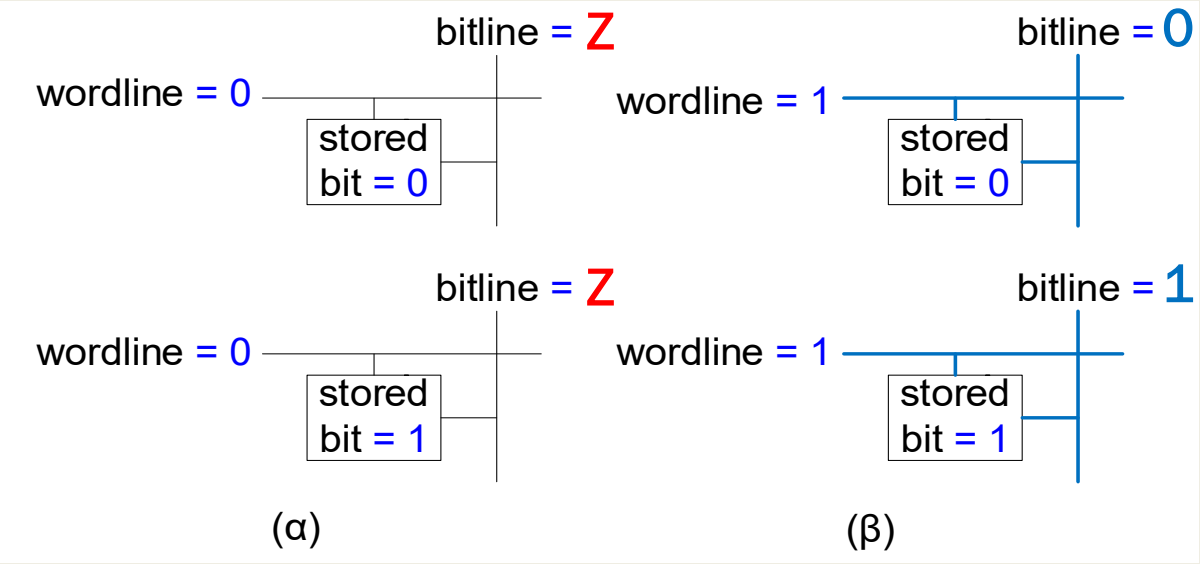
Κελιά μνήμης

- 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)



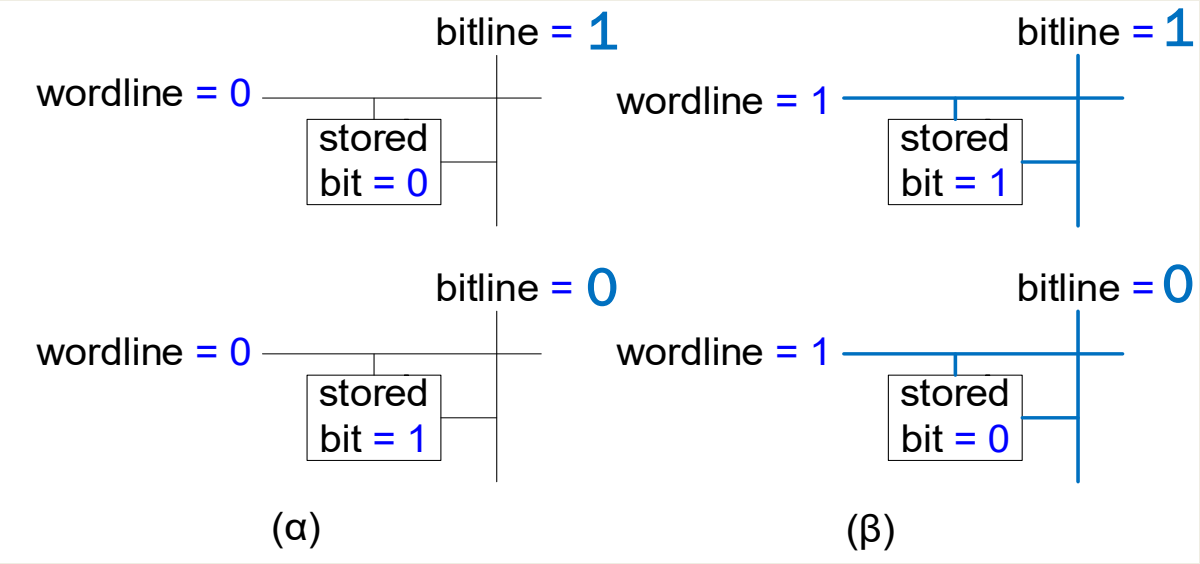
Κελιά μνήμης: Ανάγνωση

- (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*



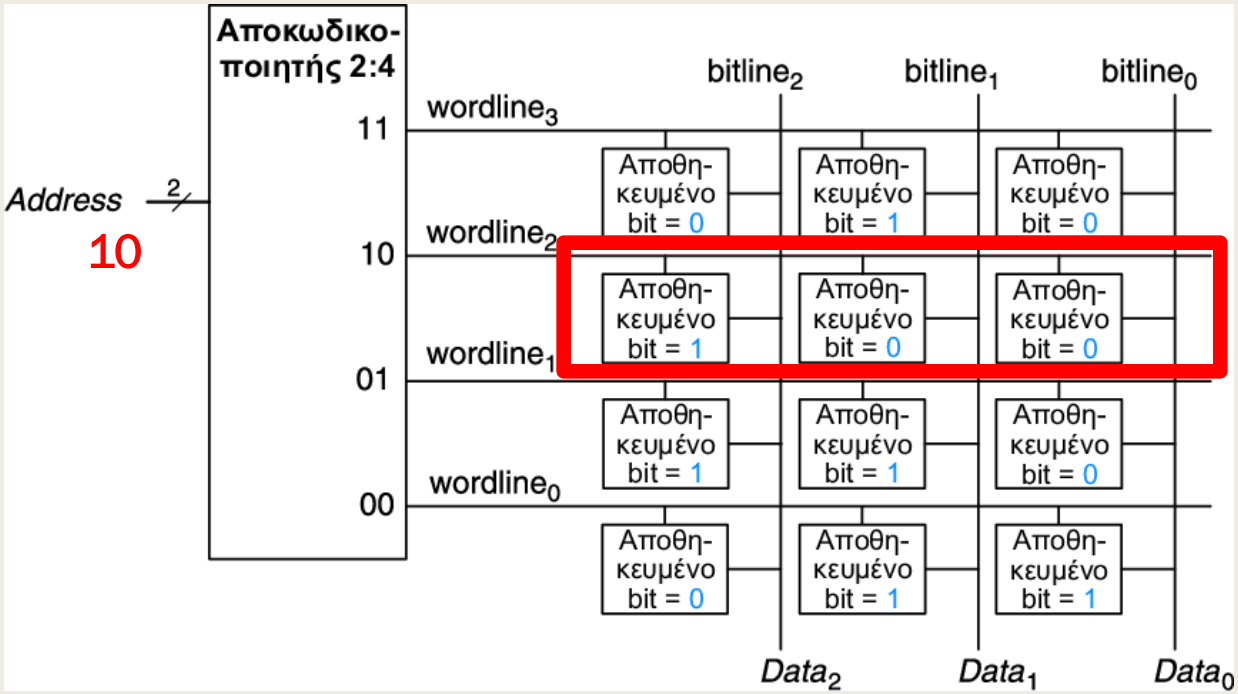
Κελιά μνήμης: Εγγραφή

- (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*



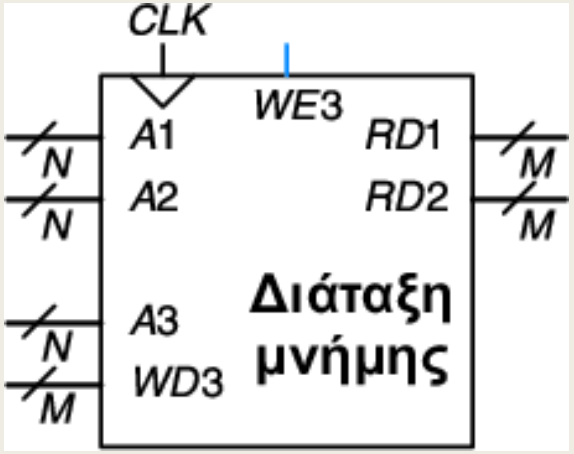
Οργάνωση μνήμης

- Η μνήμη απαρτίζεται από έναν **αποκωδικοποιητή Ν 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 **Μ κελιά μνήμης** της συγκεκριμένης σειράς της διάταξης μνήμης (**λέξης δεδομένων**)*
- Παράδειγμα: Διάταξη μνήμης 4 λέξεων × 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 κελια 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)
  - *χάνει τα δεδομένα της όταν απενεργοποιείται η τροφοδοσία ρεύματος*
- Η μνήμη ROM είναι ***μη πτητική*** (nonvolatile)
  - *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

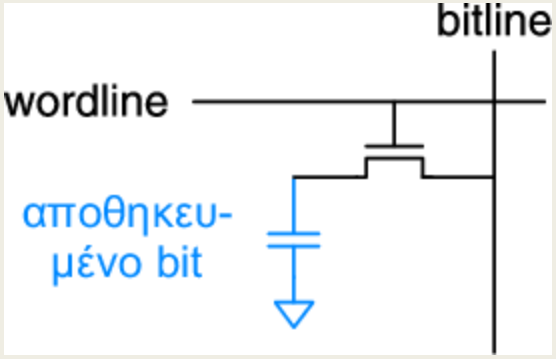
Η κύρια μνήμη στον υπολογιστή σας είναι RAM (DRAM)

Οι μνήμες Flash είναι ROM

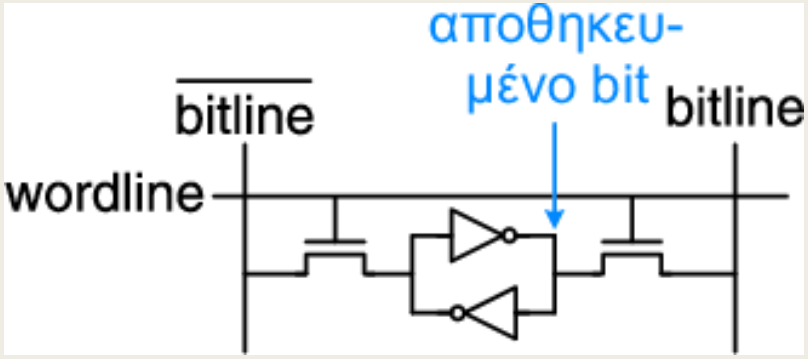
Τύποι μνήμης RAM

- 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***

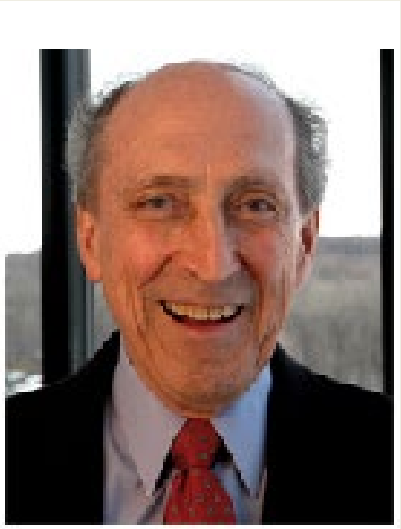
DRAM



SRAM



Robert Dennard, 1932 – 2024



- Εφηύρε τη μνήμη DRAM το 1966 στην εταιρεία IBM.
- Αν και πολλοί εξέφρασαν αρχικά τις αμφιβολίες τους σχετικά με τη βιωσιμότητα της ιδέας, έως τα μέσα της δεκαετίας του 1970 σχεδόν όλοι οι υπολογιστές διέθεταν μνήμη DRAM.
- Ο ίδιος ισχυρίζεται ότι το δημιουργικό του έργο ήταν ελάχιστο μέχρι τη στιγμή που, όταν βρέθηκε στην IBM, του έδωσαν ένα σημειωματάριο διπλωμάτων ευρεσιτεχνίας και του είπαν «γράψε όλες τις ιδέες σου εδώ».
- Από το 1965 και έπειτα έχει κατοχυρώσει 35 διπλώματα ευρεσιτεχνίας που αφορούν τους ημιαγωγούς (semiconductors) και τη μικροηλεκτρονική.

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 έχει 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

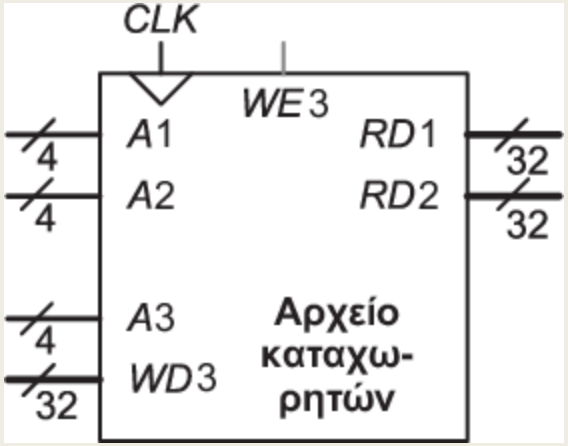
Τύπος μνήμης	Τρανζίστορ ανά κελί μνήμης	Λανθάνων χρόνος
D Flip-flop	~20	μικρός
SRAM	6	μέτριος
DRAM	1	μεγάλος

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 **αναζωογόνηση**
- 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, DDR4 και DDR5) αύξησαν την ταχύτητα του ρολογιού, με τη σχετική τιμή να ξεπερνάει το 4GHz το 2025*

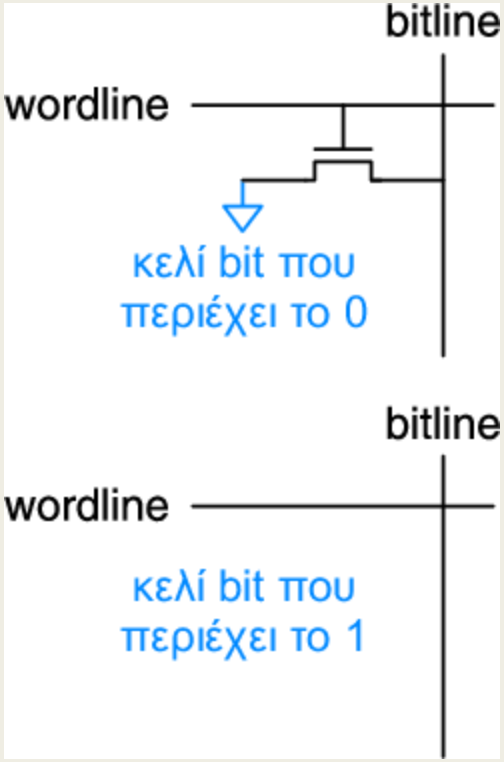
Αρχεία καταχωρητών

- 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*



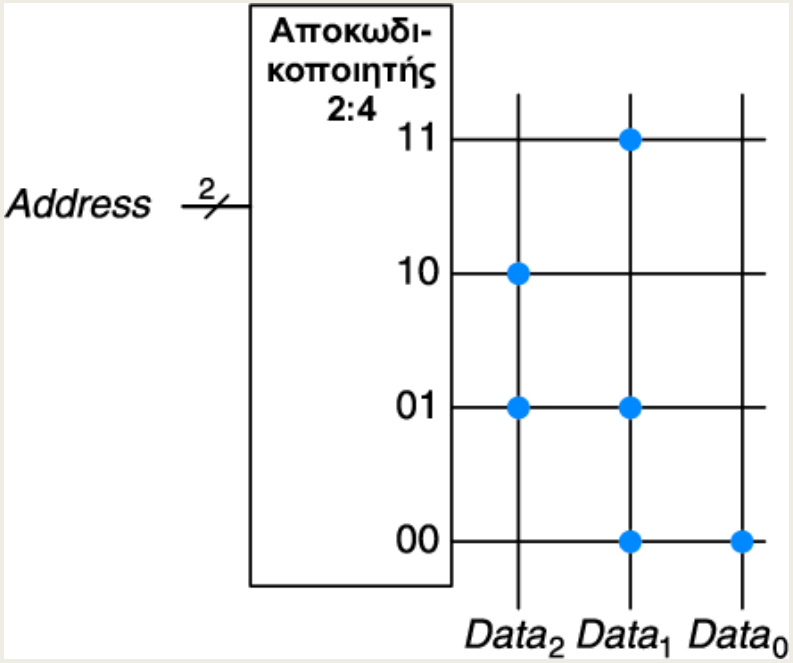
Μνήμη ROM

- 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 (**υπάρχει** το τρανζίστορ), αλλιώς έχει την τιμή 0 (**δεν υπάρχει** το τρανζίστορ)*
- Παράδειγμα: Διάταξη μνήμης ROM 4 λέξεων × 3 bit (N =2, M=3) που υλοποιεί τις συναρτήσεις $Data_0$, $Data_1$ και $Data_2$

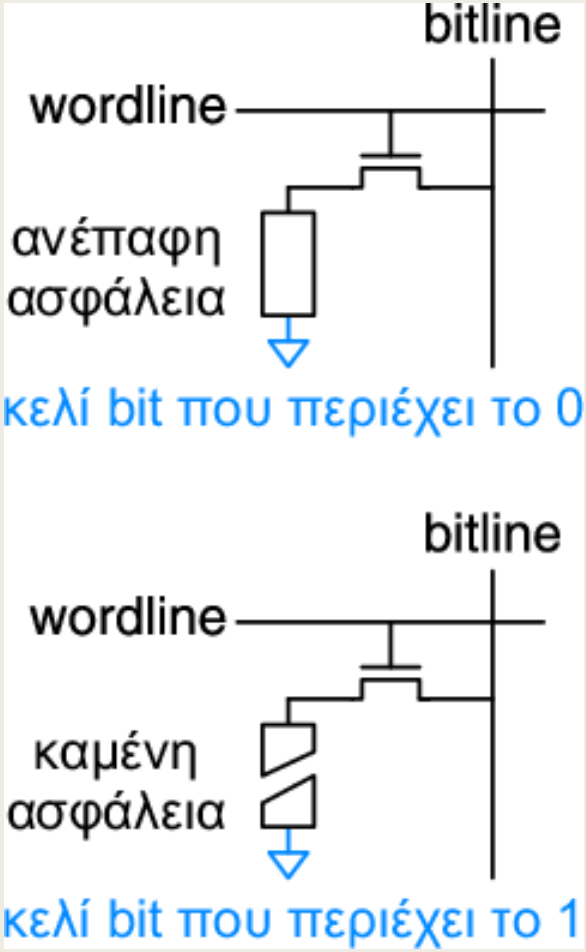


Address	Data		
11	0	1	0
10	1	0	0
01	1	1	0
00	0	1	1

$$Data_2 = A_1 \oplus A_0 \quad Data_1 = \overline{A_1} + A_0 \quad Data_0 = \overline{A_1}\,\overline{A_0}$$

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)
  - *Χρησιμοποιούνται στο διάστημα*



Μνήμες EPROM, EEPROM και Flash

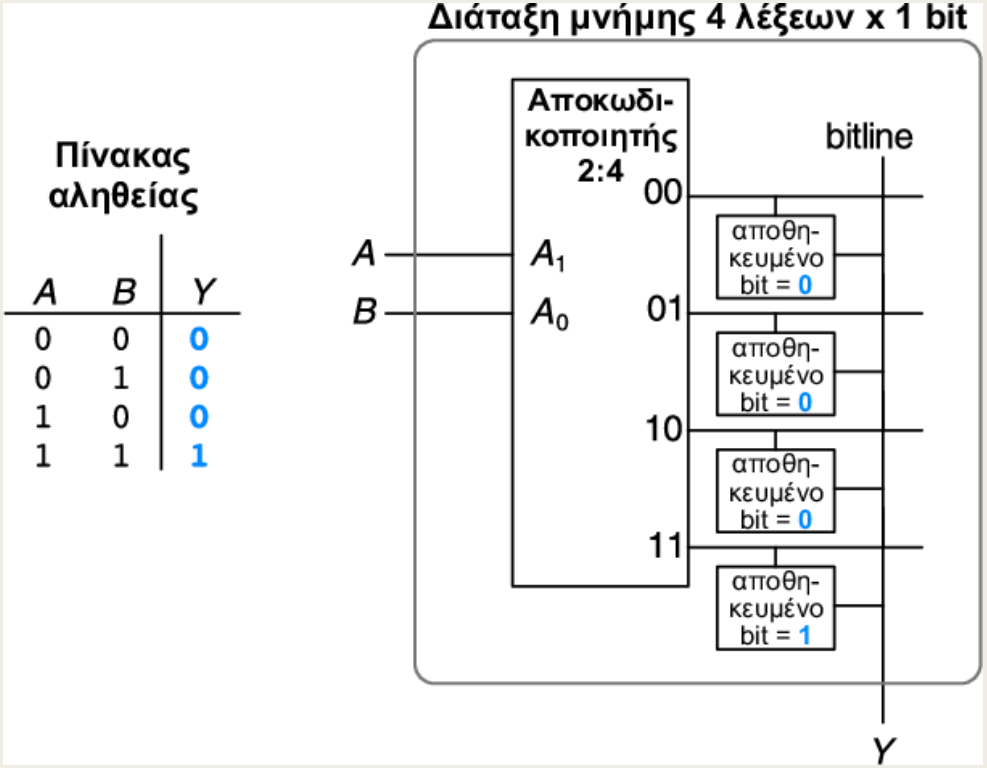
- Οι επαναπρογραμματιζόμενες μνήμες PROM διαθέτουν αντιστρέψιμο μηχανισμό για τη σύνδεση ή την αποσύνδεση του τρανζίστορ με/από τη γείωση (GND)
- Οι **απαλείψιμες μνήμες PROM** (Erasable PROM, EPROM) αντικαθιστούν το τρανζίστορ και την ασφάλεια με ένα **τρανζίστορ μετέωρης πύλης**
  - *Με κατάλληλα υψηλές 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
  - *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
    - Οι μνήμες flash με USB (Universal Serial Bus) έχουν αντικαταστήσει τις δισκέτες και τους δίσκους CD στην αποθήκευση αρχείων.

Fujio Masuoka, 1943 -

- Κάτοχος διδακτορικού τίτλου σπουδών (Ph.D.) από τη Σχολή Ηλεκτρολόγων Μηχανολόγων του Πανεπιστημίου Tohoku στην Ιαπωνία.
- Από το 1971 έως το 1994 ανέπτυξε μνήμες και κυκλώματα υψηλών ταχυτήτων για την εταιρεία Toshiba.
- Εφηύρε τη **μνήμη flash** στα πλαίσια ενός ανεπίσημου έργου το οποίο ολοκλήρωσε στα τέλη της δεκαετίας του 1970 αφιερώνοντας όλο τον ελεύθερο χρόνο που διέθετε τις νύχτες και τα Σαββατοκύριακα.
  - *Η μνήμη flash ονομάστηκε έτσι επειδή η διαδικασία απαλοιφής της μνήμης θυμίζει το φλας των φωτογραφικών μηχανών.*
- Η Toshiba άργησε να υλοποιήσει εμπορικά την ιδέα· η Intel ήταν εκείνη που διέθεσε πρώτη εμπορικά προϊόντα στην αγορά το 1988.
- Η μνήμη flash έχει εξελιχθεί σε μια αγορά με τζίρο 170 Δισ. δολαρίων ετησίως και αυξάνει κατά 25 Δισ. δολάρια ετησίως.
- Αργότερα ο Fujio Masuoka έγινε μέλος του διδακτικού προσωπικού στο Πανεπιστήμιο Tohoku και έχει επικεντρώσει τις ερευνητικές προσπάθειές του στην ανάπτυξη τριδιάστατων τρανζίστορ.

Πίνακες αναζήτησης (LUT)

- Οι **διατάξεις μνήμης** που χρησιμοποιούνται για την υλοποίηση **συνδυαστικής λογικής** ονομάζονται ***πίνακες αναζήτησης*** (lookup tables, LUT)
  - *Υλοποιείται ο πίνακας αλήθειας*
- Παράδειγμα: Διάταξη μνήμης 4 λέξεων × 1 bit η οποία χρησιμοποιείται ως πίνακας αναζήτησης για την υλοποίηση της συνάρτησης $Y = AB$
  - *Χρησιμοποιούνται στις διατάξεις FPGA*

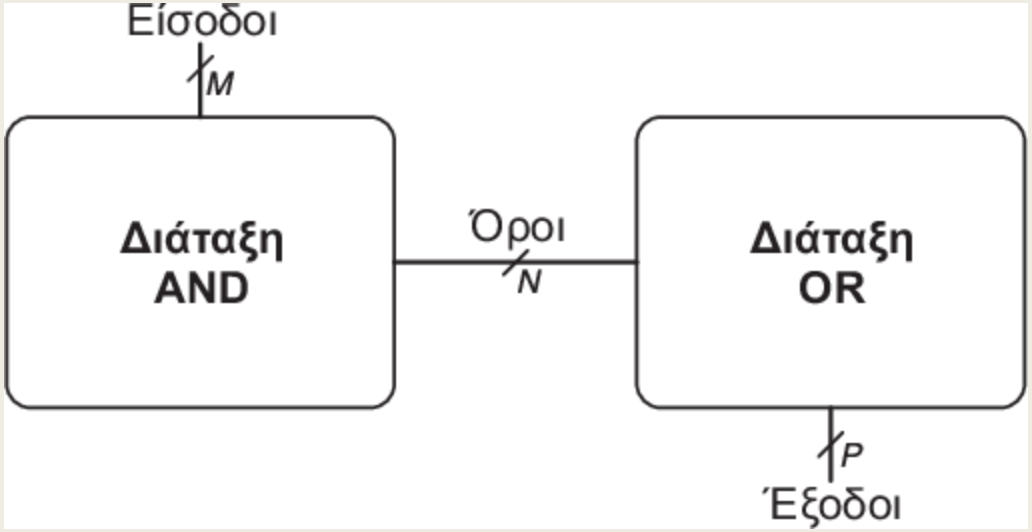


Διατάξεις λογικής (Logic arrays)

- **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)*

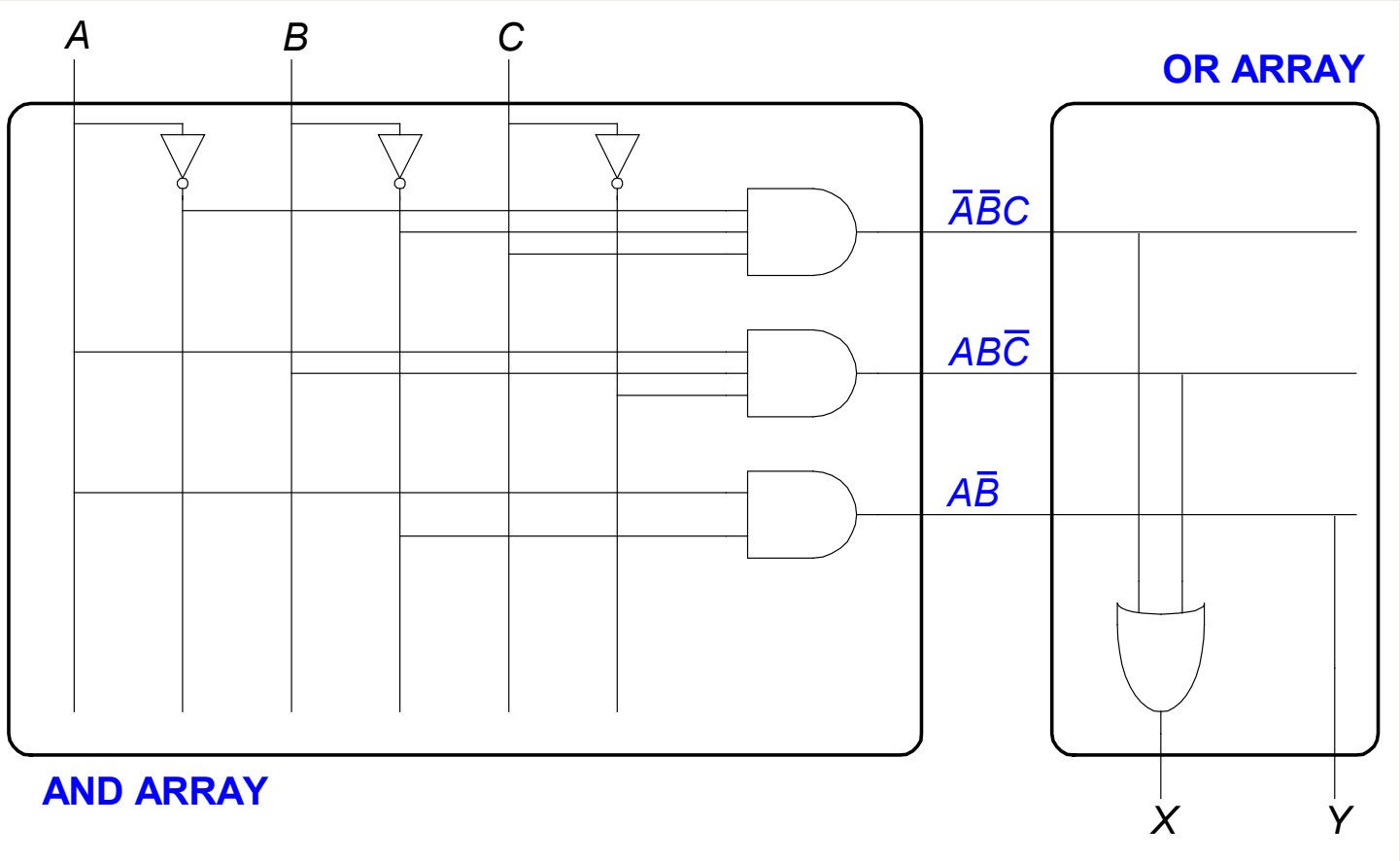
Προγραμματιζόμενες διατάξεις λογικής (PLA)

- 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 × N × P bit διαθέτει M εισόδους, N όρους και P εξόδους*



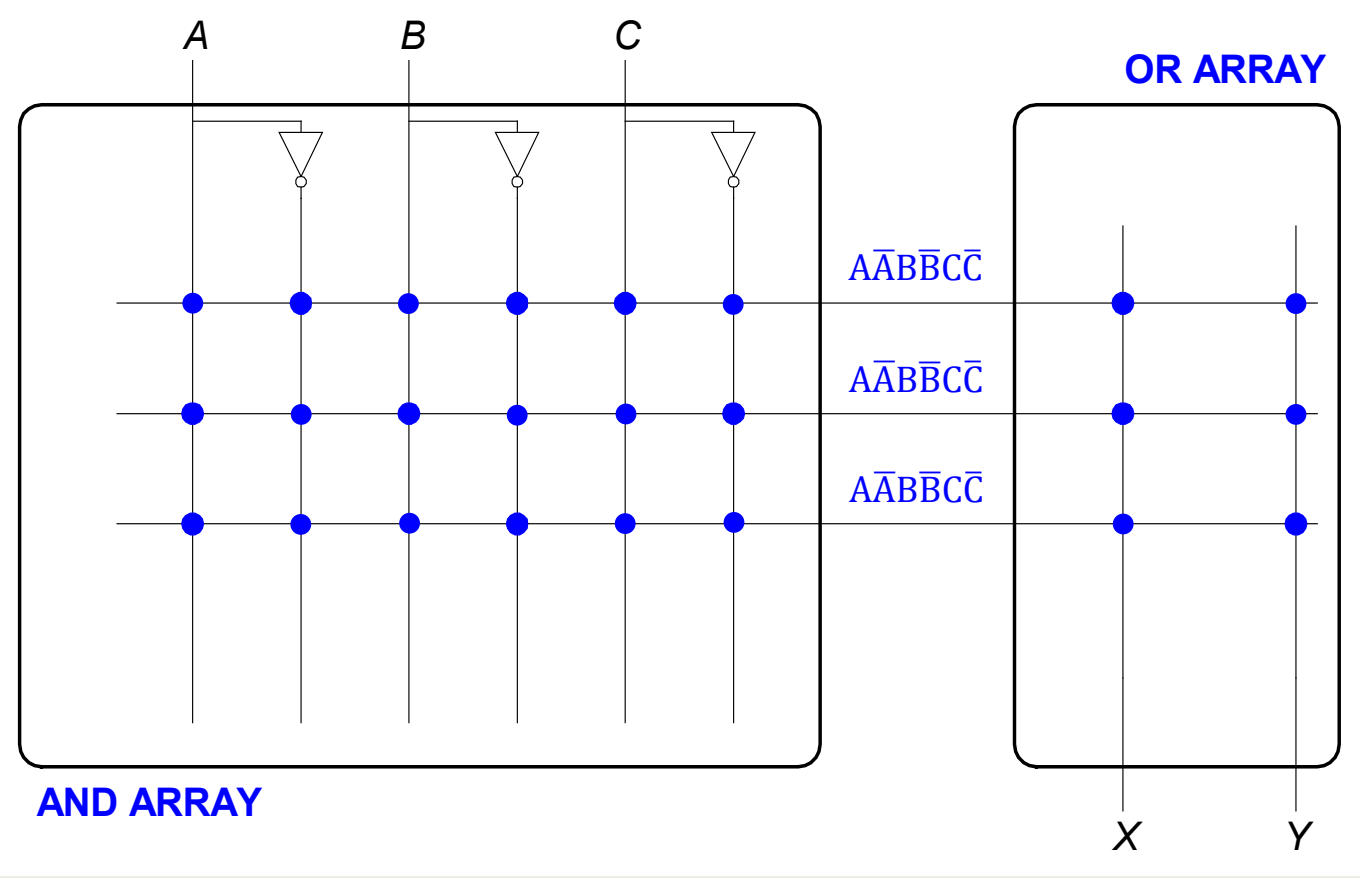
Προγραμματιζόμενες διατάξεις λογικής (PLA)

- Παράδειγμα: Διάταξη PLA με διαστάσεις 3 × 2 × 3 bit που διαθέτει M=3 εισόδους, N=3 όρους και P=2 εξόδους
  - *Υλοποίηση των λογικών συναρτήσεων* $\mathbf{X = \bar{A}\,\bar{B}C + AB\bar{C}}$ *και* $\mathbf{Y = A\bar{B}}$ με χρήση λογικής δύο επιπέδων



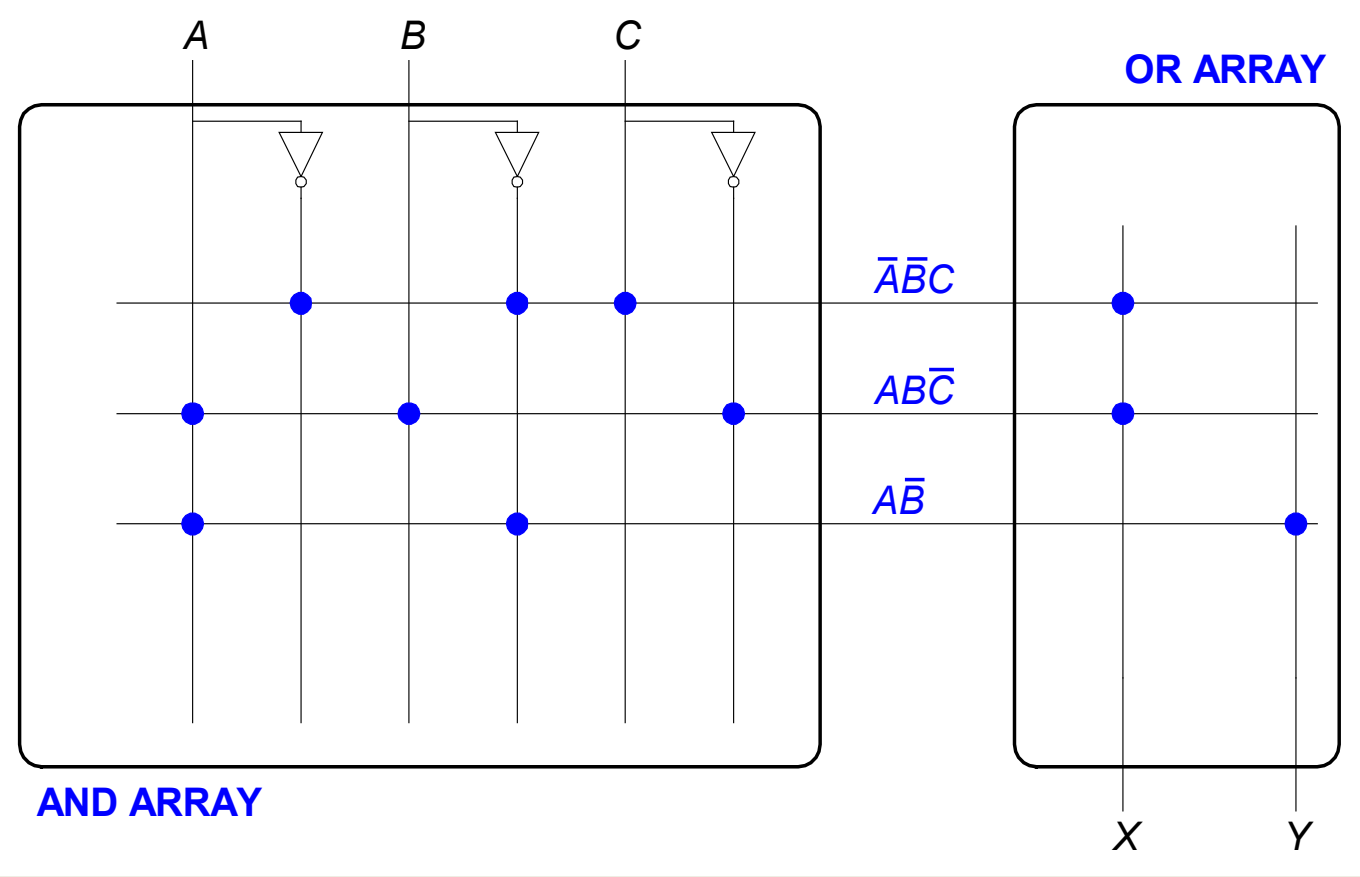
Προγραμματιζόμενες διατάξεις λογικής (PLA)

- Παράδειγμα: Διάταξη PLA με διαστάσεις 3 × 2 × 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 × 2 × 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* $\mathbf{X = \bar{A}\,\bar{B}C + AB\bar{C}}$ *και* $\mathbf{Y = A\bar{B}}$
  - *Για τις συνδέσεις μπορεί να χρησιμοποιηθεί ο* ***συμβολισμός με τελείες***

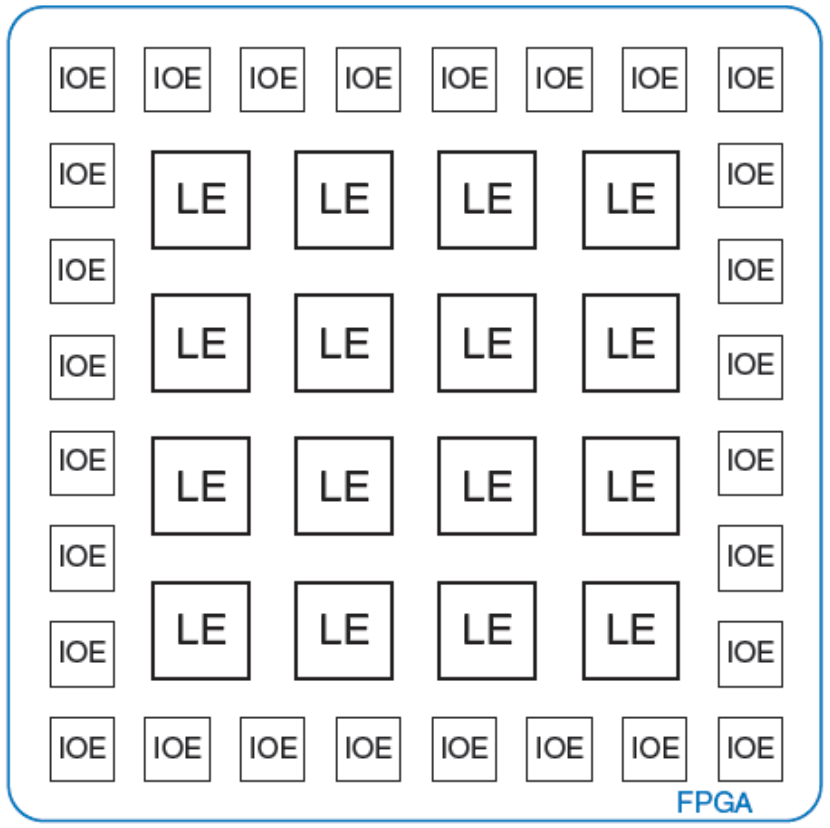


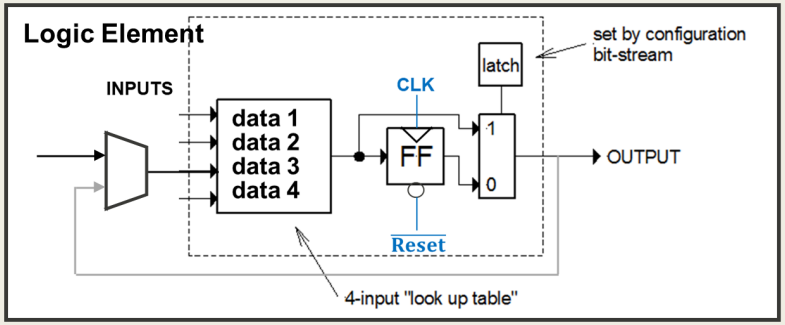
Προγραμματιζόμενες από τον χρήστη διατάξεις πυλών (Field Programmable Gate Arrays - FPGA)

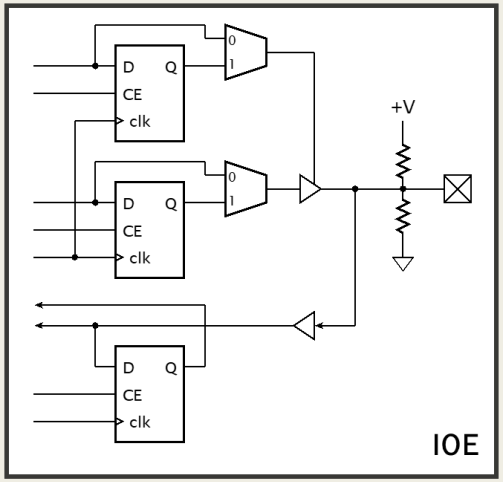
- Οδηγούν τις εξελίξεις στην υλοποίηση αλγορίθμων στο υλικό
- Παρέχουν δυνατότητα σχεδίασης **VLSI κυκλωμάτων χαμηλού κόστους** στο πεδίο, με τη χρήση σχετικά φθηνών εργαλείων λογισμικού (CAD)
  - *κόστος ανεκτό από μικρές εταιρείες που δραστηριοποιούνται στην ανάπτυξη επιταχυντών υλικού και γενικότερα 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 ή μηχανές AI** σε νέες τεχνολογίες
- 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)

- Ένα απλοποιημένο FPGA συνίσταται από:
  - ***LEs (Logic elements)***: *για υλοποίηση λογικής*
  - ***IOEs (Input/output elements)***: *για την διασύνδεση με τον εξωτερικό κόσμο*
  - ***Programmable interconnection***: *συνδέουν LEs και IOEs*

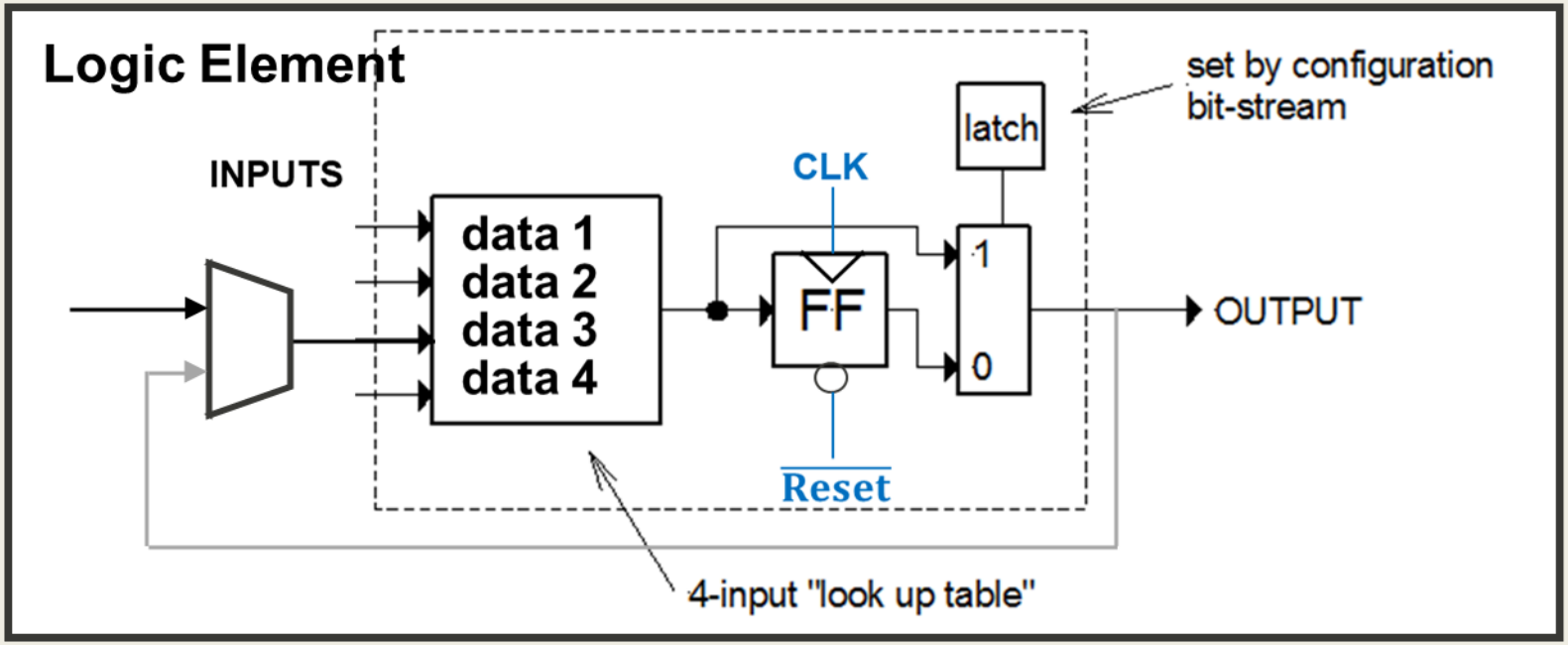






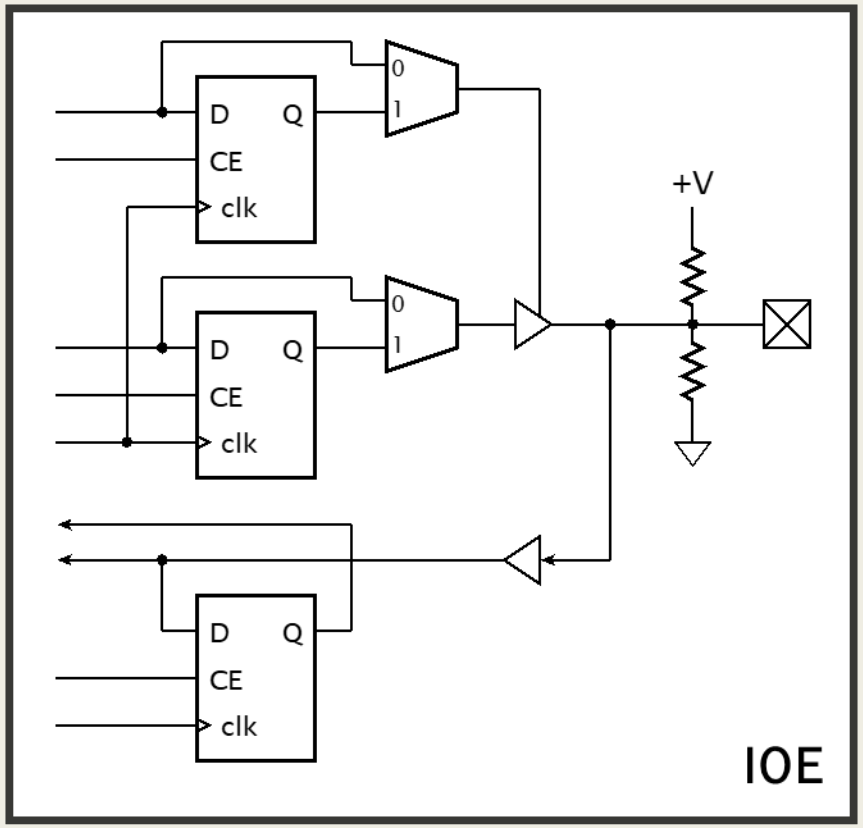
Ένα «απλοποιημένο» FPGA Logic Element (LE)

- **Πίνακες αναζήτησης (Look-Up Table, LUT)** των 4 εισόδων
  - *είναι διατάξεις μνήμης που υλοποιούν τον πίνακα αλήθειας συνδυαστικής λογικής*
  - *οι νέες τεχνολογίες FPGA έχουν LUT των 6 εισόδων*
- **Flip-flop με Reset active low**
  - *μπορεί να αποθηκεύει την έξοδο του LUT για έξοδο*
- **Πολυπλέκτες**
  - *Για επιλογή μεταξύ συνδυαστικής και ακολουθιακής (μέσω καταχωρητή) εξόδου*
  - *Για επιλογή σύνδεσης της εισόδου data 3 με την έξοδο (ανάδραση)*



IOEs (Input/output elements)

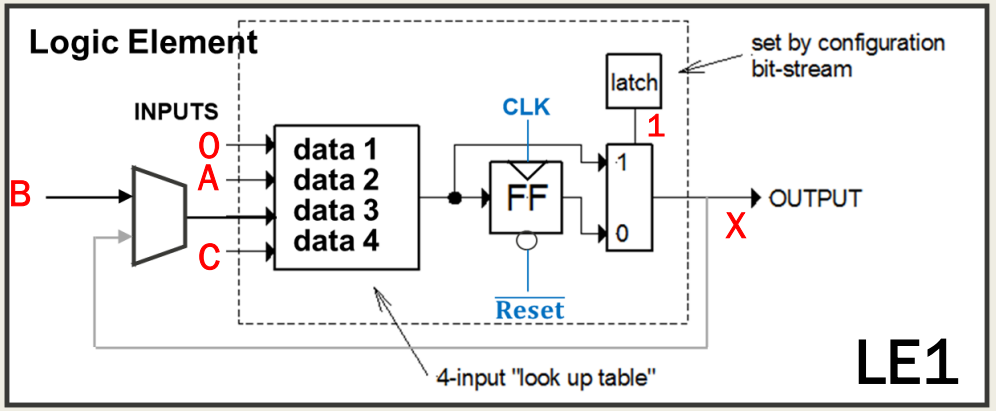
- Συνήθως, υποστηρίζουν συνδυαστική είσοδο/έξοδο ή μέσω καταχωρητή ή/και απομονωτή τριών καταστάσεων
  - *Προγραμματιζόμενα επίπεδα λογικής, ρυθμός μεταβολής, κατώφλι εισόδου, …*
  - *Αμφίδρομη επικοινωνία*



Δημιουργία συναρτήσεων με χρήση LE*

- Υλοποίηση των λογικών συναρτήσεων $\mathbf{X = \bar{A}\,\bar{B}C + AB\bar{C}}$ και $\mathbf{Y = A\bar{B}}$

0	A	B	C	X
data 1	data 2	data 3	data 4	Output
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
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



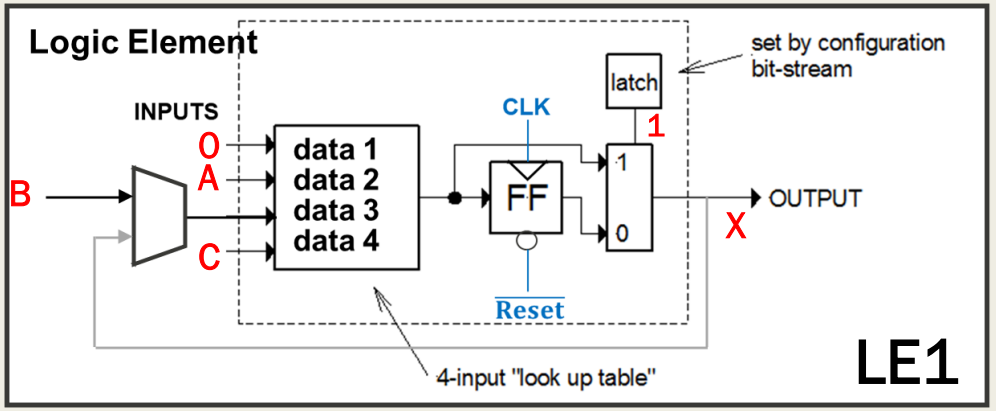
Απαιτούνται δύο Logic Elements, παράλληλα: LE1 για το X και LE2 για το Y

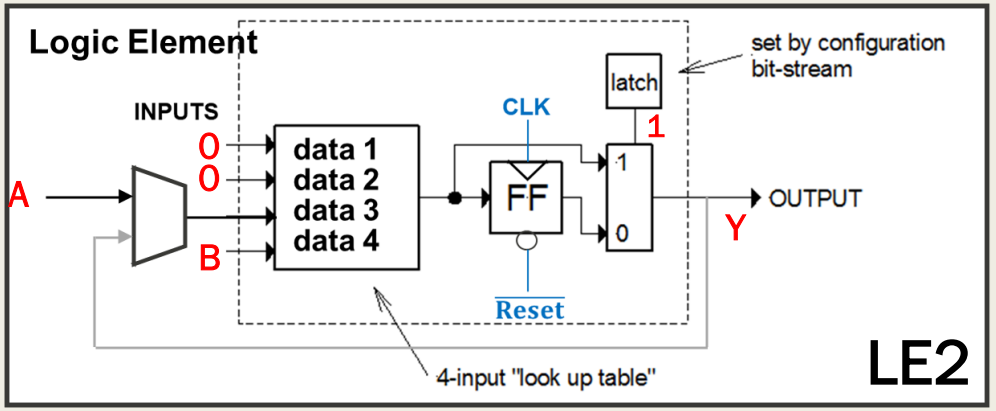
* Αντίστοιχο του Παραδείγματος 5.8

Δημιουργία συναρτήσεων με χρήση LE*

- Υλοποίηση των λογικών συναρτήσεων $\mathbf{X = \bar{A}\,\bar{B}C + AB\bar{C}}$ και $\mathbf{Y = A\bar{B}}$

0	0	A	B	Y
data 1	data 2	data 3	data 4	Output
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
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





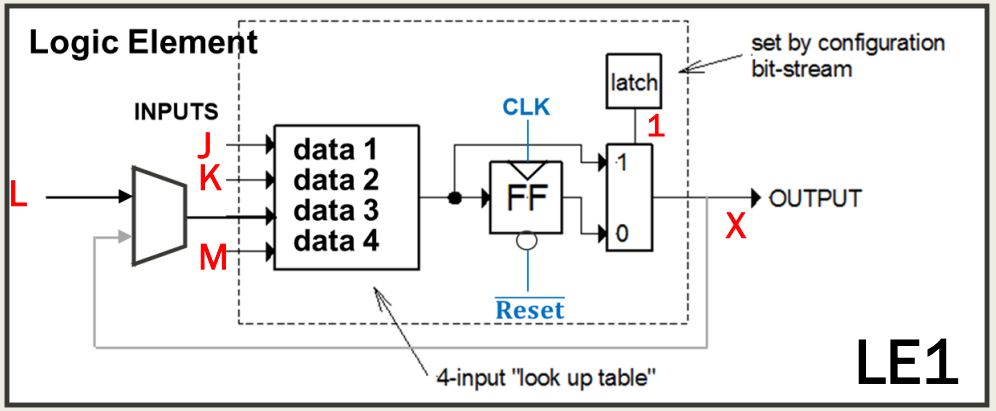
Απαιτούνται δύο Logic Elements, παράλληλα: LE1 για το X και LE2 για το Y

* Αντίστοιχο του Παραδείγματος 5.8

Δημιουργία συναρτήσεων με χρήση LE*

- Υλοποίηση της λογικής συνάρτησης $\Upsilon = JKLMPQR$

J	K	L	M	X
data 1	data 2	data 3	data 4	Output
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	1



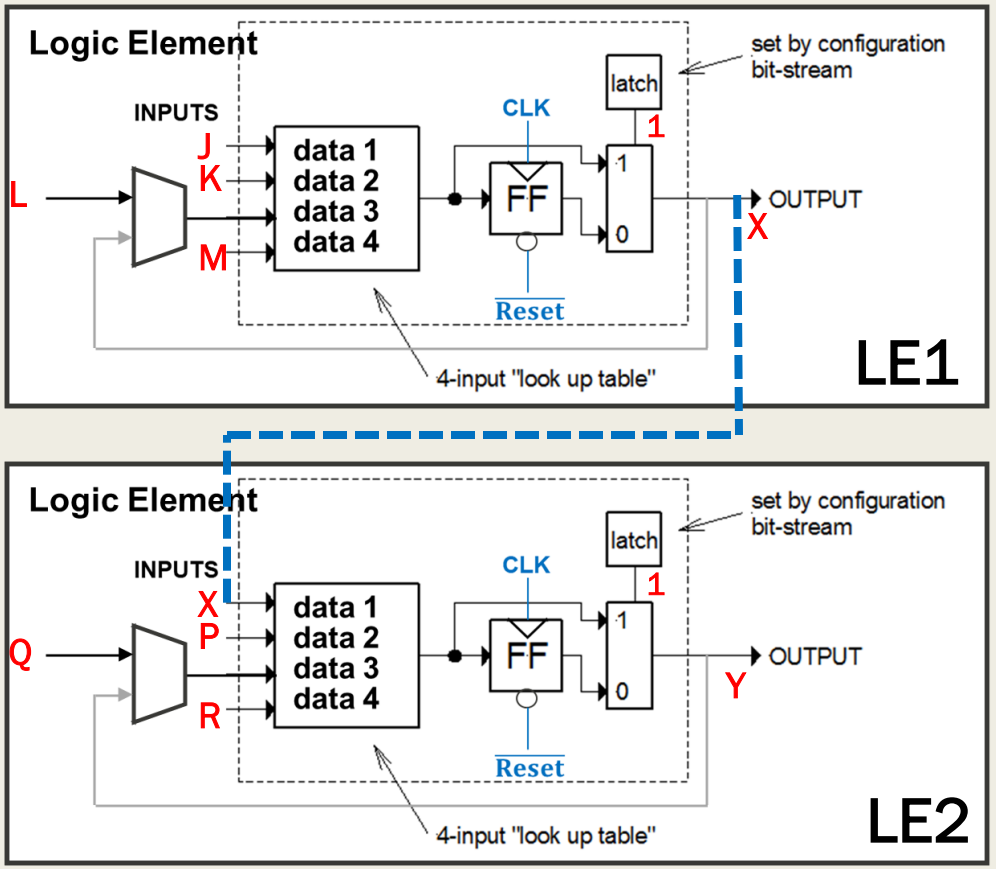
Απαιτούνται δύο Logic Elements σε σειρά:
LE1 για το X=JKLM και LE2 για το Y=XPQR

* Αντίστοιχο του Παραδείγματος 5.8

Δημιουργία συναρτήσεων με χρήση LE*

- Υλοποίηση της λογικής συνάρτησης $\Upsilon = JKLMPQR$

X	P	Q	R	Y
data 1	data 2	data 3	data 4	Output
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	1



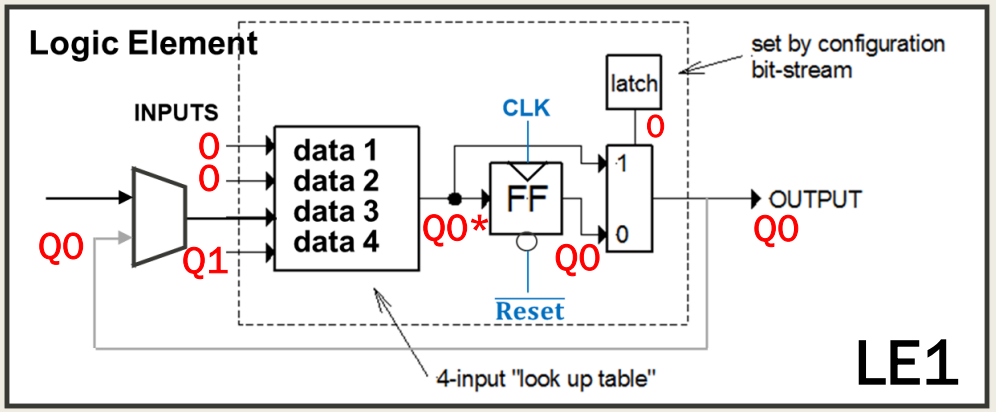
Απαιτούνται δύο Logic Elements σε σειρά:
LE1 για το X=JKLM και LE2 για το Y=XPQR

* Αντίστοιχο του Παραδείγματος 5.8

Δημιουργία συναρτήσεων με χρήση LE*

- Υλοποίηση ενός μετρητή διαίρεσης δια του 3

0	0	Q0	Q1	Q0*
data 1	data 2	data 3	data 4	Output
0	0	0	0	1
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



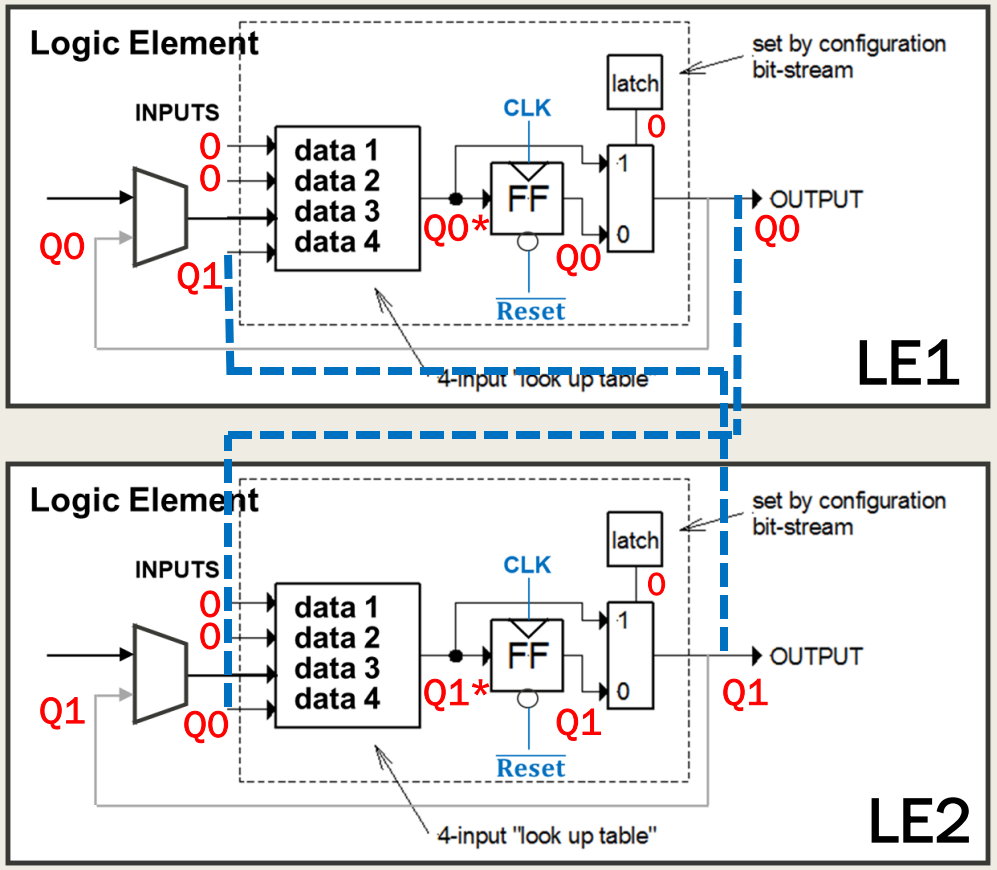
3 Logic Elements, LE1, LE2 παράλληλα. LE3 σε σειρά:
LE1 για το Q0*/Q0, LE2 για το Q1*/Q1 και LE3 για το Y

* Αντίστοιχο του Παραδείγματος 5.8 $Q1^* = \overline{Q1}\,Q0$ $Q0^* = \overline{Q1}\,\overline{Q0}$ $Y = \overline{Q1}\,\overline{Q0}$

Δημιουργία συναρτήσεων με χρήση LE*

- Υλοποίηση ενός μετρητή διαίρεσης δια του 3

0	0	Q1	Q0	Q1*
data 1	data 2	data 3	data 4	Output
0	0	0	0	0
0	0	0	1	1
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



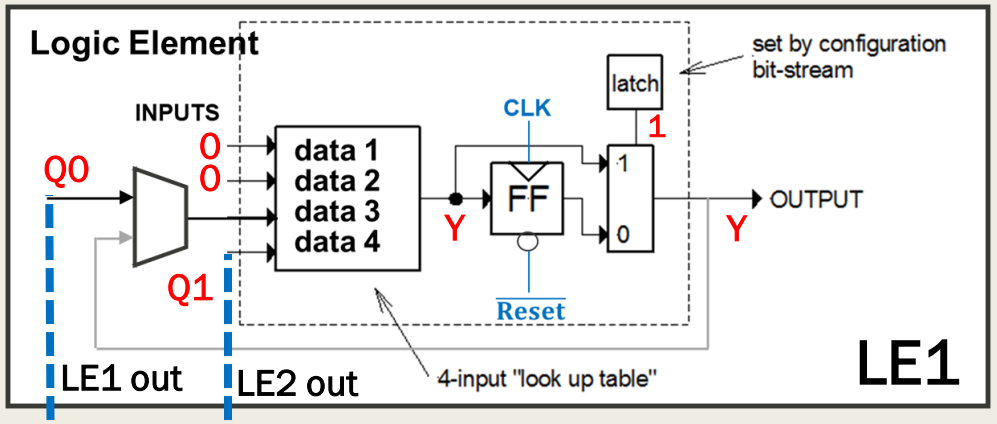
3 Logic Elements, LE1, LE2 παράλληλα. LE3 σε σειρά:
LE1 για το Q0*/Q0, LE2 για το Q1*/Q1 και LE3 για το Y

* Αντίστοιχο του Παραδείγματος 5.8 $Q1^* = \overline{Q1}\,Q0$ $Q0^* = \overline{Q1}\,\overline{Q0}$ $Y = \overline{Q1}\,\overline{Q0}$

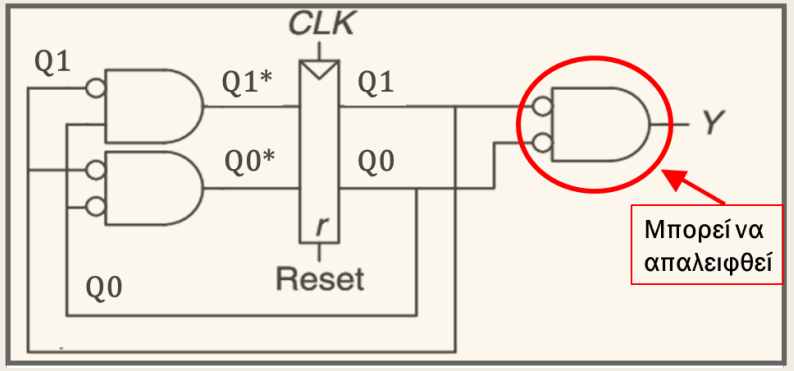
Δημιουργία συναρτήσεων με χρήση LE*

- Υλοποίηση ενός μετρητή διαίρεσης δια του 3

0	0	Q0	Q1	Y
data 1	data 2	data 3	data 4	Output
0	0	0	0	1
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



Εάν το LE είχε δύο διακριτές εξόδους μία συνδυαστική και μία ακολουθιακή λογική θα χρησιμοποιούσαμε 2 LE (LE1,LE3 κοινό)



3 Logic Elements, LE1, LE2 παράλληλα. LE3 σε σειρά: LE1 για το Q0*/Q0, LE2 για το Q1*/Q1 και LE3 για το Y

* Αντίστοιχο του Παραδείγματος 5.8 $Q1^* = \overline{Q1}\,Q0$ $Q0^* = \overline{Q1}\,\overline{Q0}$ $Y = \overline{Q1}\,\overline{Q0}$

Καθυστέρηση διάδοσης στα LE*

- Ας υποθέσουμε ότι το απλοποιημένο LE έχει τα χαρακτηριστικά χρονισμού της διάταξης Cyclone IV της Intel και θέλουμε να κατασκευάσουμε ένα ακολουθιακό κύκλωμα που να τρέχει στα **200 MHz**
- Πόσα επίπεδα στοιχείων LE (N) το πολύ μπορούμε να χρησιμοποιήσουμε μεταξύ δύο καταχωρητών;

$T_c = 1 / f = 1 / 200\ \text{MHz} = 5.000\ \text{ps}$

$t_{pd} = t_{LE} + t_{wire} = 627\ ps$

$T_c \geq t_{pcq} + t_{pd} + t_{setup} \Rightarrow T_c \geq t_{pcq} + N\ t_{pd} + t_{setup} \Rightarrow$

$N \leq [T_c - (t_{pcq} + t_{setup})] / t_{pd} \Rightarrow$

$N \leq [5.000 - (199 + 76)]\ \text{ps} / 627\ \text{ps} \Rightarrow$

$N \leq 7{,}54 \Rightarrow N = 7$

Χαρ/κά χρονισμού

Καταχωρητή:

t_{pcq} = 199 ps

t_{setup} = 76 ps

t_{hold} = 0 ps

Logic Element-Wire:

t_{LE} = 381 ps

t_{wire} = 246 ps*

* Η καθυστέρηση διάδοσης των συρμάτων μεταξύ των στοιχείων LE

* Αντίστοιχο του Παραδείγματος 5.10