Ψηφιακή Σχεδίαση

Ενότητα 5: Συνδυαστικά Κυκλώματα και Ακολουθιακά κυκλώματα

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών
http://arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σκοπός της ενότητας

- Να αναλυθούν τα συνδυαστικά και ακολουθιακά κυκλώματα ως προς τις αρχές σχεδιασμού και τον ιεραρχικό σχεδιασμό.
- Να δειχθούν μέθοδοι ως προς την σχεδίαση.
- Να δειχθεί η υλοποίηση αθροιστών και ημι-αθροιστών.
- Να αναλυθεί ο σχεδιασμός CLA.

Υπάρχουν 2 τύποι κυκλωμάτων

- Συνδυαστικά Κυκλώματα.
- Ακολουθιακά κυκλώματα.

Συνδυαστικά Κυκλώματα (1)

- Ένα συνδυαστικό κύκλωμα αποτελείται από λογικές πύλες των οποίων οι έξοδοι, ανά πάσα στιγμή, καθορίζονται από κάποιο λογικό συνδυασμό των τιμών στις εισόδους του κυκλώματος.
- Για n μεταβλητές εισόδου, υπάρχουν 2^n πιθανοί (δυαδικοί) συνδυασμοί τιμών εισόδου.
- Για κάθε τέτοιο συνδυασμό, υπάρχει μια πιθανή δυαδική τιμή στην κάθε έξοδο.

Συνδυαστικά Κυκλώματα (2)

Ένα συνδυαστικό κύκλωμα με n εισόδους και m εξόδους, μπορεί να περιγραφεί από:

1. Ένα πίνακα αληθείας που περιέχει τις τιμές στις εξόδους του κυκλώματος για κάθε πιθανό συνδυασμό στις εισόδους του κυκλώματος, ή
2. m δυαδικές συναρτήσεις, 1 για κάθε μεταβλητή εξόδου.

Ακολουθιακά Κυκλώματα

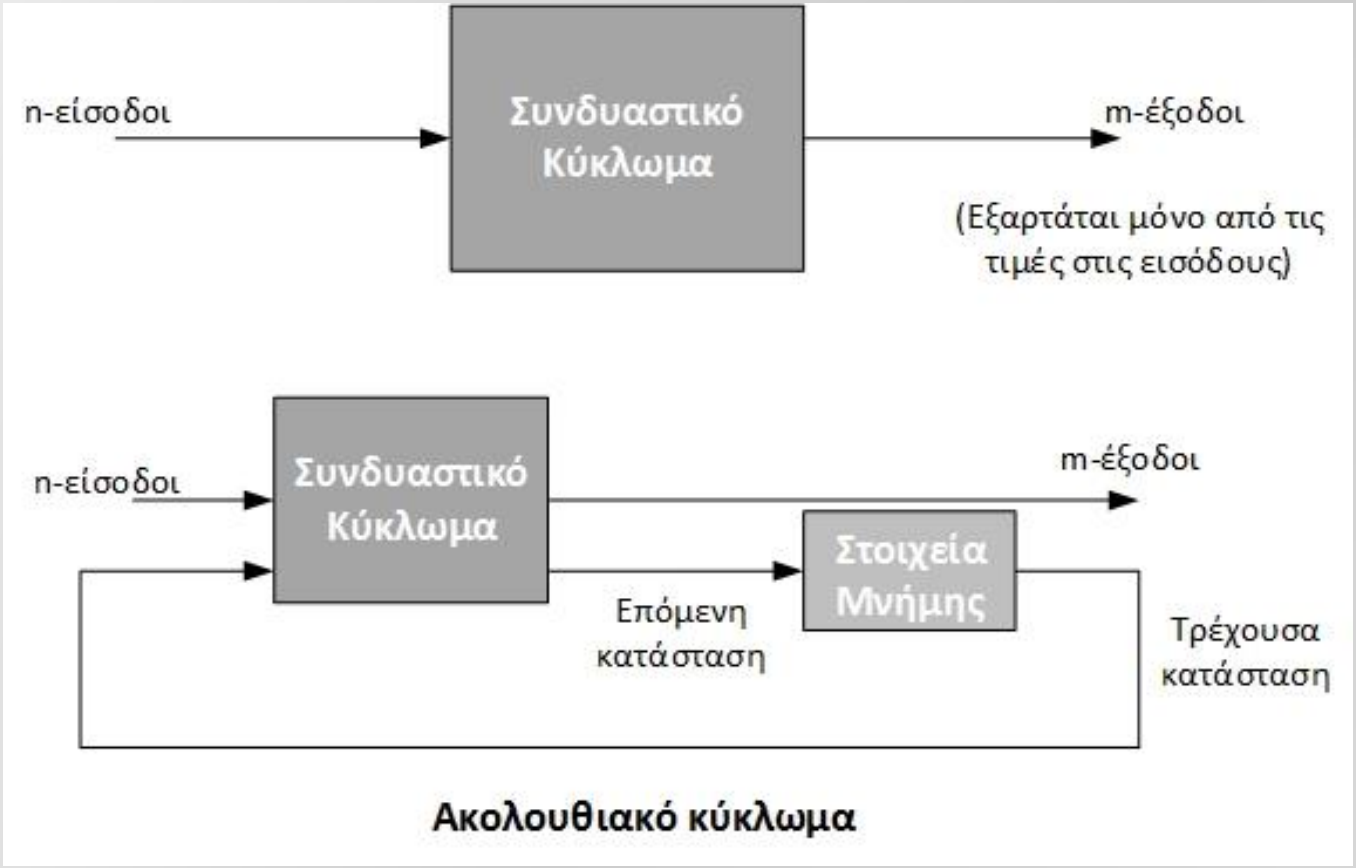
- Τα συνδυαστικά κυκλώματα δεν έχουν μνήμη.

Οι τιμές στις εξόδους τους εξαρτώνται ΜΟΝΟ από τις τρέχουσες τιμές στις εισόδους.

- Τα ακολουθιακά κυκλώματα αποτελούνται από συνδυαστική λογική και από στοιχεία μνήμης (χρησιμοποιούνται για αποθήκευση κάποιων καταστάσεων του κυκλώματος).

→ Οι τιμές στις εξόδους εξαρτώνται από τις τρέχουσες τιμές στις εισόδους και τις τιμές προηγούμενων εισόδων (που έχουν αποθηκευτεί στα στοιχεία μνήμης).

Συνδυαστικά vs. Ακολουθιακά Κυκλώματα



Ακολουθιακό κύκλωμα

Αρχές Σχεδιασμού

- Ο σύγχρονος ψηφιακός σχεδιασμός ασχολείται με διάφορες μεθόδους και εργαλεία που χρησιμοποιούνται για τον σχεδιασμό , την προσομοίωση και την επαλήθευση πολύπλοκων κυκλωμάτων και συστημάτων.
- Σημαντικές Αρχές Σχεδιασμού
  - Ιεραρχικός σχεδιασμός.
  - Σχεδιασμός από πάνω προς τα κάτω (Top-Down).
  - Σχεδιασμός με την βοήθεια του υπολογιστή (CAD).
  - Γλώσσες προγραμματισμού / περιγραφής υλικού (VHDL, Verilog).
  - Λογική Σύνθεση.

Ιεραρχικό Σχεδιασμός

- Η τεχνική **“Διαίρει και Βασίλευε”** χρησιμοποιείται για την επιτυχή αντιμετώπιση των αυξανόμενων απαιτήσεων του σχεδιασμού πολύπλοκων συστημάτων.

(πολλές φορές της τάξεως των εκατομμυρίων πυλών).

- Το κύκλωμα τεμαχίζεται σε **κομμάτια (blocks)**, επαναληπτικά , μέχρι που καταλήγει σε **πρωταρχικά / βασικά (primitive / common)** στοιχεία.
- Πρόκληση: Επιβεβαίωση ότι το κύκλωμα λειτουργεί ορθά σε κάθε επίπεδο της ιεραρχίας!

Ιεραρχικός Σχεδιασμός – Παράδειγμα (1)

Παράδειγμα: Περιττή συνάρτηση 9εισόδων

- Ανώτατο Επίπεδο:

9 είσοδοι, 1 έξοδος.

- 2ο Επίπεδο:

4 όμοια συνιστώσα περιττών συναρτήσεων των 3ων bit, σε δύο επίπεδα.

- 3ο Επίπεδο:

Συνάρτηση XOR 3ων bit, για κάθε συνιστώσα του 2ου επιπέδου.

- 4ο Επίπεδο (τελευταίο):

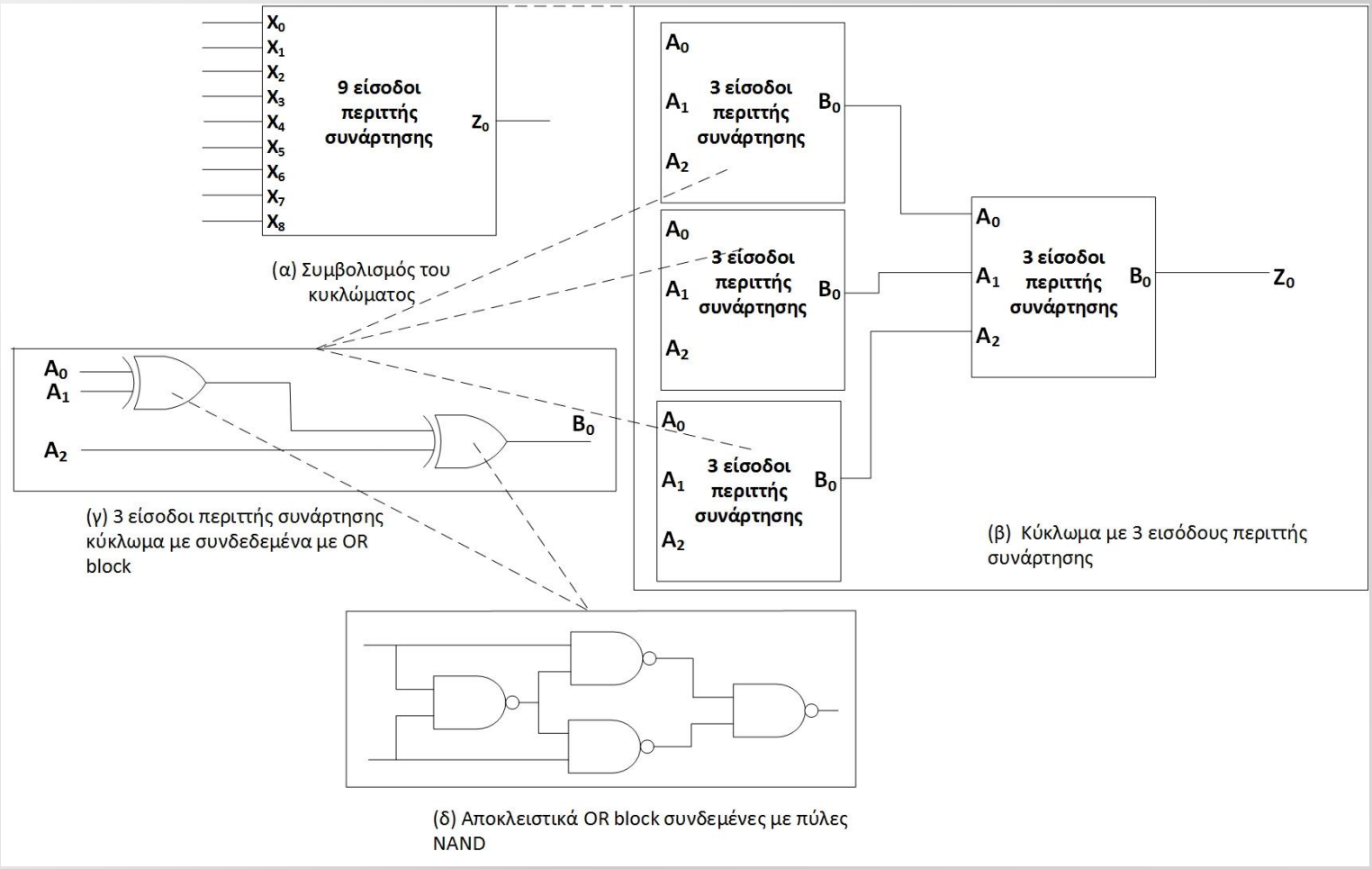
Υλοποίηση XOR με 4 NAND 2-εισόδων, για κάθε XOR του 3ου επιπέδου.

- Πρωταρχικό στοιχείο:

NAND

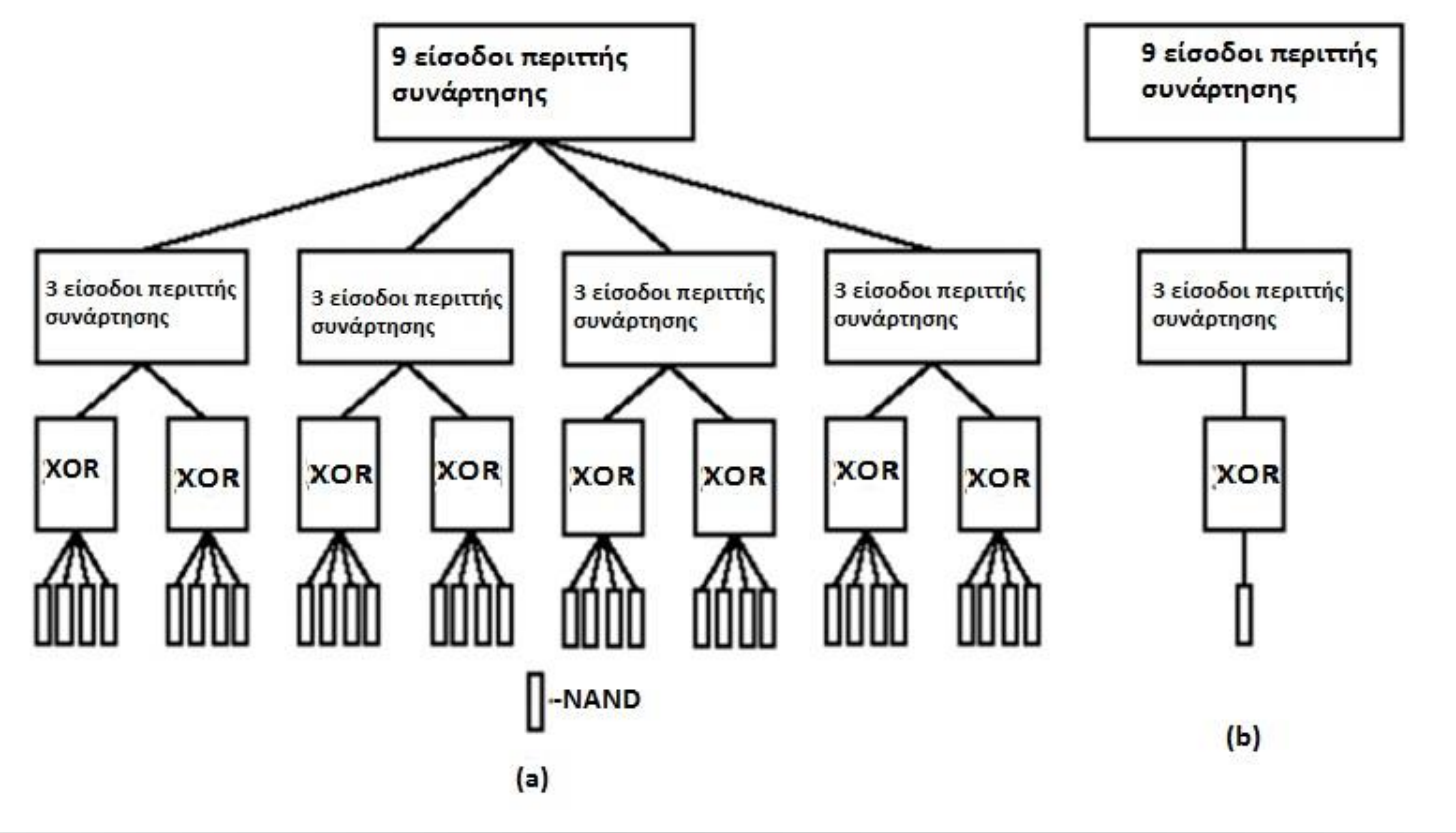
- Ο σχεδιασμός απαιτεί 4 x 2 x 4 = 32 πύλες NAND 2-εισόδων.

Ιεραρχικός Σχεδιασμός – Παράδειγμα (2)



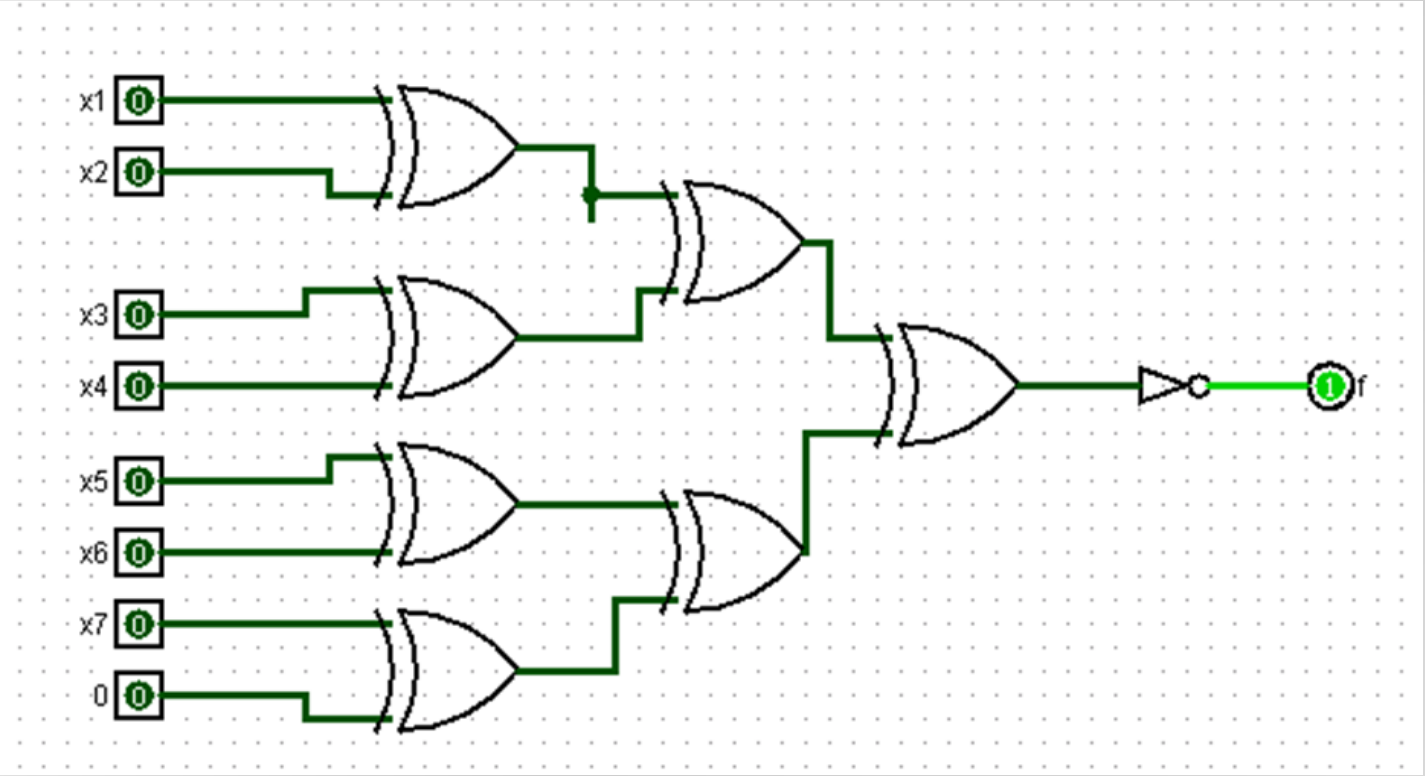
Ιεραρχικός Σχεδιασμός – Παράδειγμα (3)

- Παράδειγμα: Περιττή συνάρτηση 9 εισόδων



Παράδειγμα με Logisim (1)

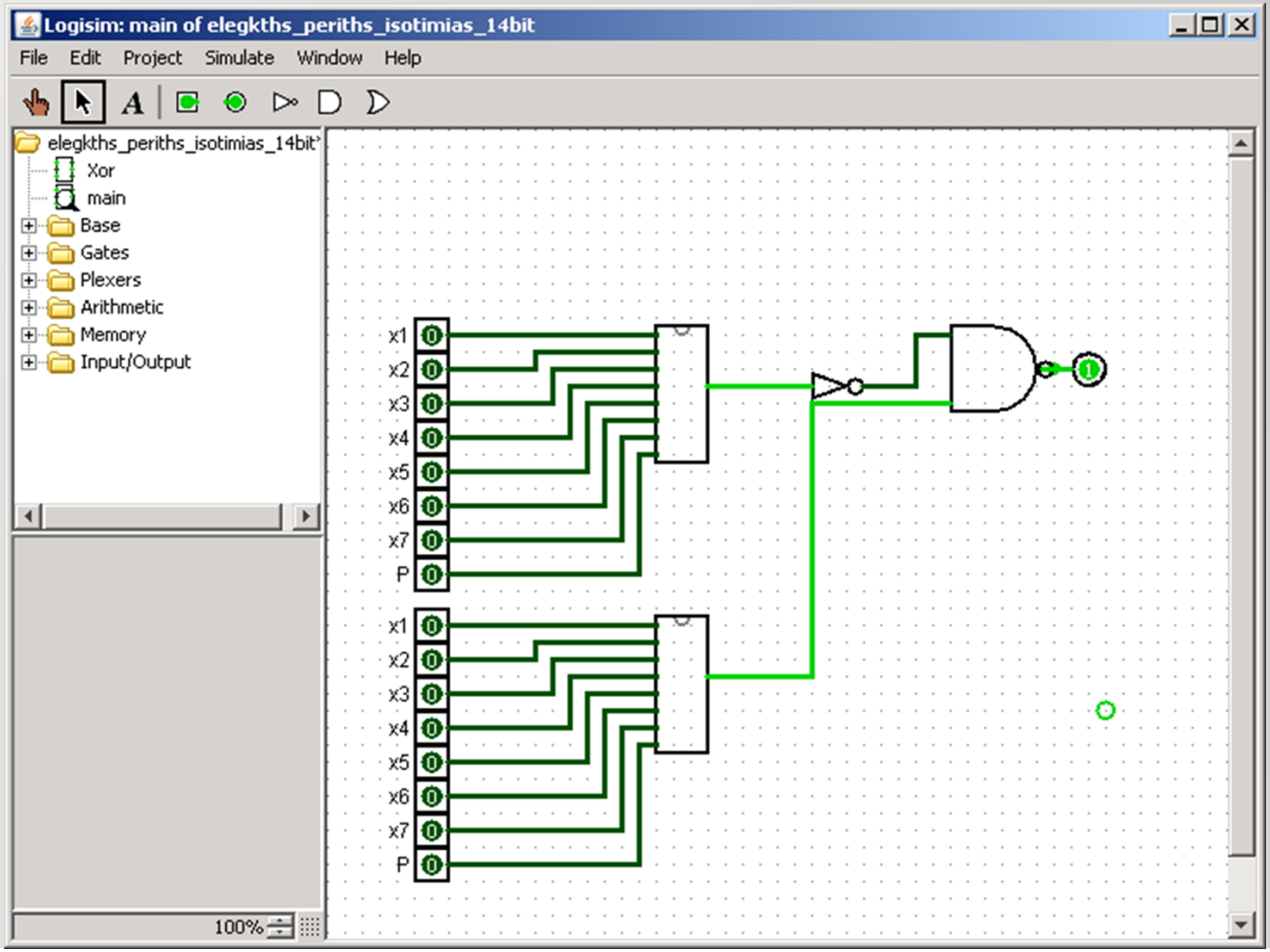
- Κατασκευάζουμε ένα κύκλωμα περιττής ισοτιμίας των 7 bit και το αποθηκεύουμε ως XOR



Παράδειγμα με Logisim (2)

- Από το menu Project → Add circuit προσθέτουμε το main.
- Επιλέγουμε το main και από το menu Project → Set as main circuit.
- Σύρουμε από το αριστερό πάνελ το κύκλωμα xor και το τοποθετούμε στο παράθυρο main.

Παράδειγμα με Logisim (3)

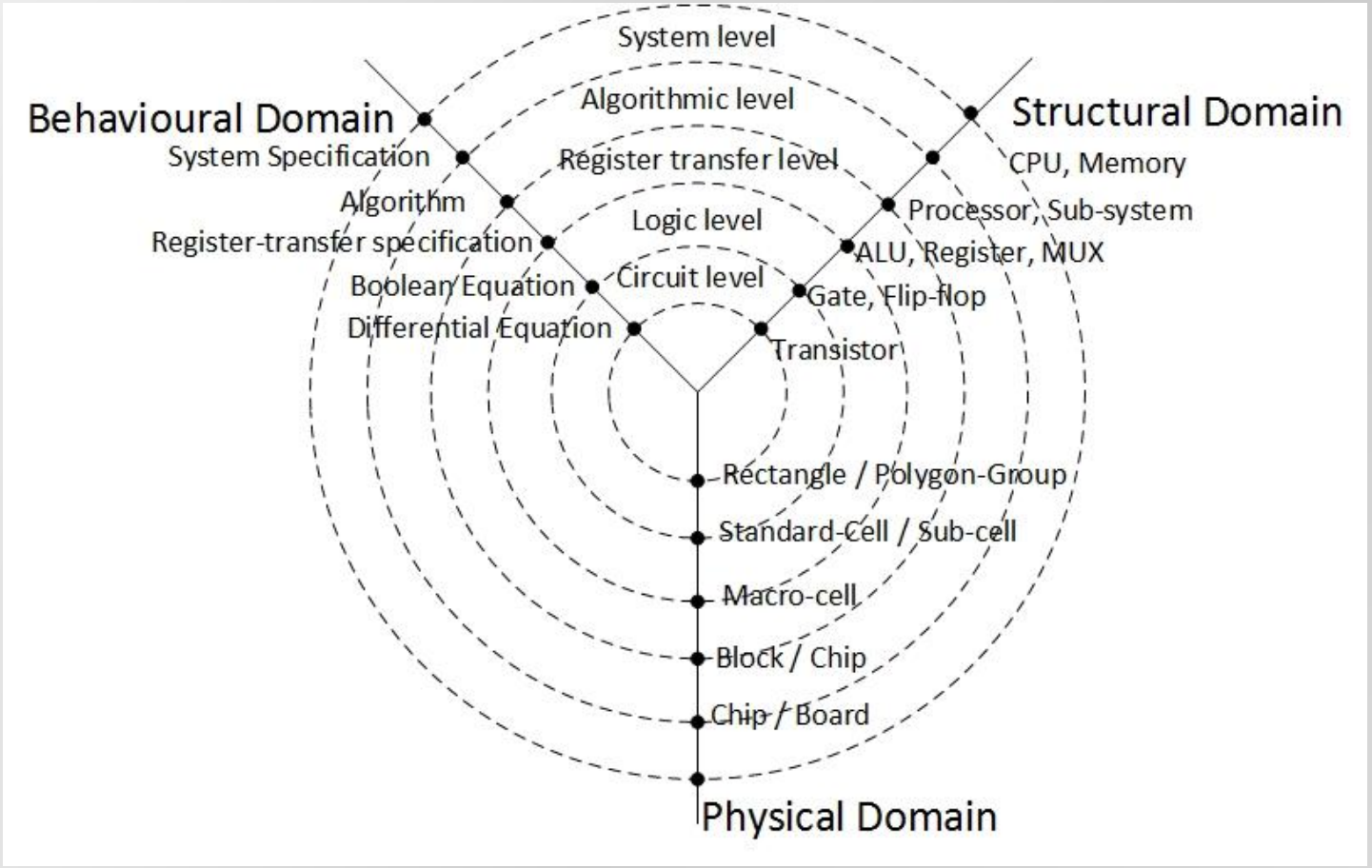


Γιατί είναι χρήσιμη η ιεράρχηση (1)

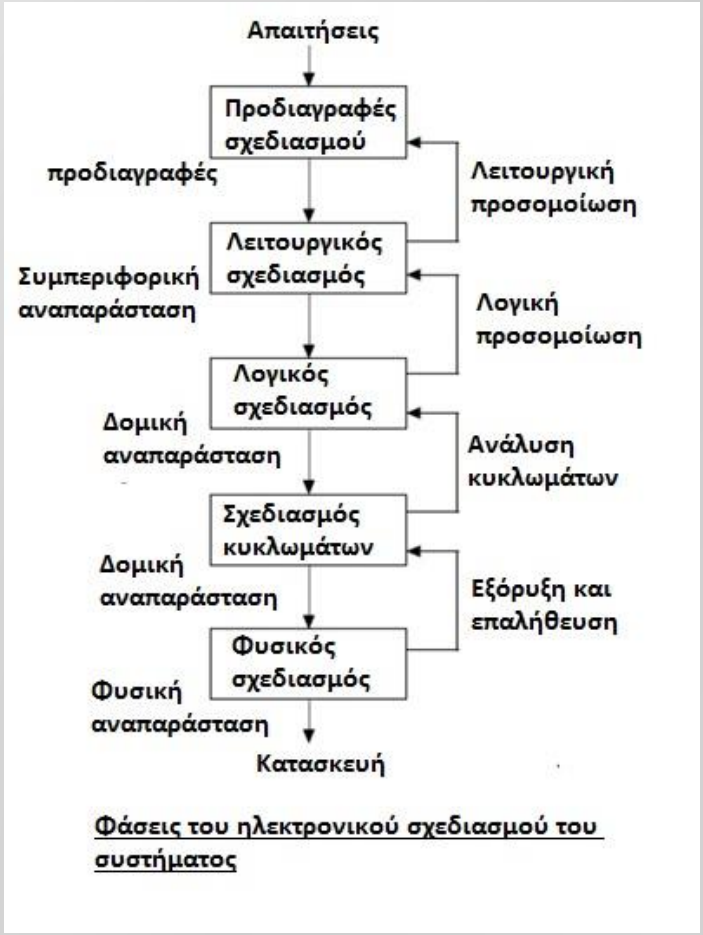
- Επιτρέπει την επαναχρησιμοποίηση (reuse) ήδη σχεδιασμένων μπλοκ.
  - Πανομοιότυπα κομμάτια μπορούν να χρησιμοποιηθούν σε διάφορα σημεία του σχεδιασμού ή και ακόμη σε διαφορετικούς σχεδιασμούς.
  - Όπου είναι δυνατόν, προσπαθούμε να αποσυνθέσουμε (decompose) έναν πολύπλοκο σχεδιασμό σε βασικά επαναχρησιμοποιήσιμα μπλοκ (reusable blocks).
  - Επαναχρησιμοποιήσιμα μπλοκ:
    - Έχουν ήδη επαληθευτεί και τεκμηριωθεί.
    - Αποθηκεύονται σε βιβλιοθήκες για γενική χρήση.

→ Μειώνει την πολυπλοκότητα του σχεδιασμού και της αντιπροσώπευσης του συνολικού σχεδίου (schematic) του κυκλώματος.

Γιατί είναι χρήσιμη η ιεράρχηση (2)

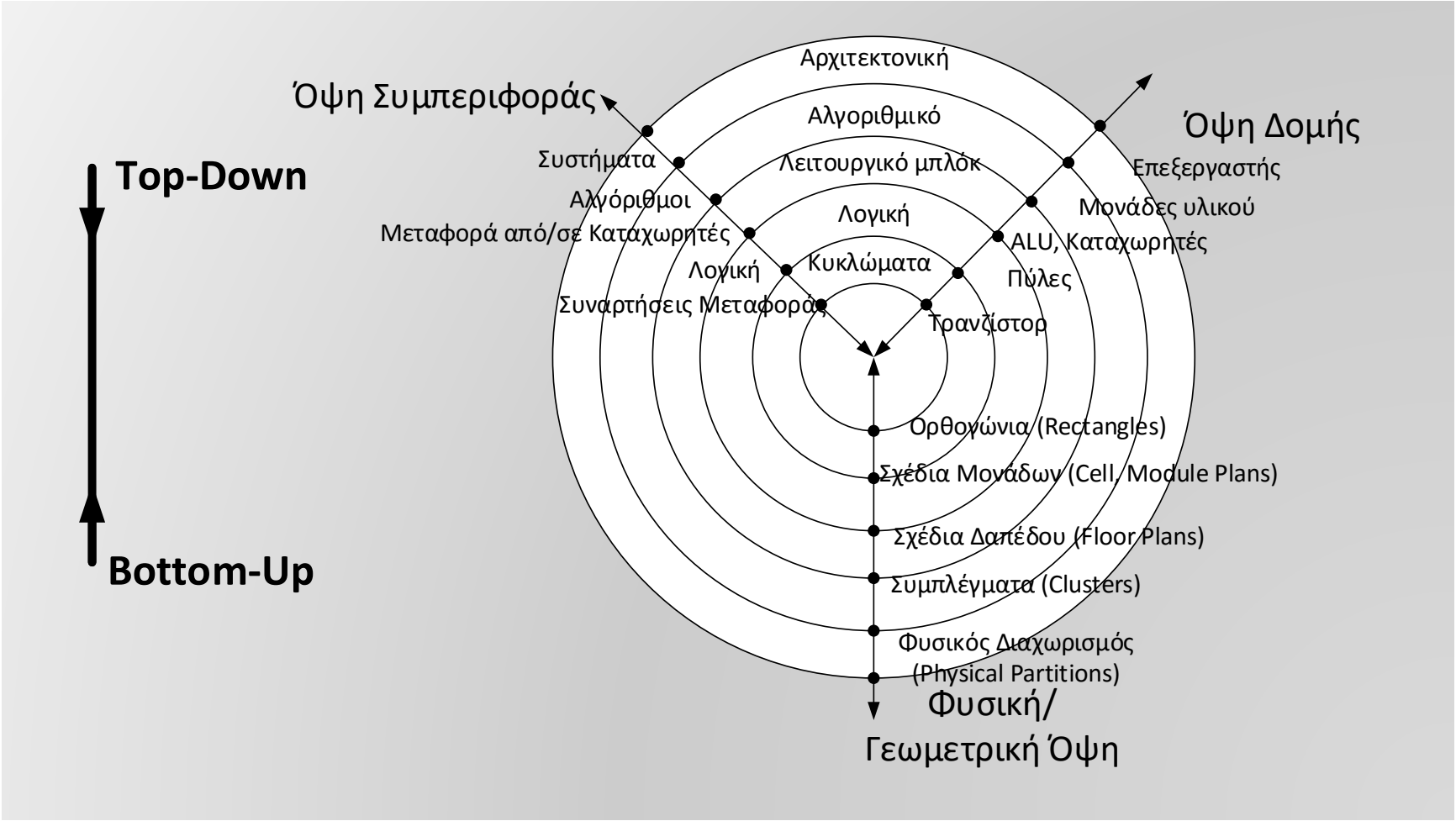


Γιατί είναι χρήσιμη η ιεράρχηση (3)



Φάσεις του ηλεκτρονικού σχεδιασμού του συστήματος

Χάρτης Y


Όψη Συμπεριφοράς
Όψη Δομής
Top-Down
Bottom-Up
Αρχιτεκτονική
Αλγοριθμικό
Λειτουργικό μπλόκ
Λογική
Κυκλώματα
Συστήματα
Αλγόριθμοι
Μεταφορά από/σε Καταχωρητές
Λογική
Συναρτήσεις Μεταφοράς
Επεξεργαστής
Μονάδες υλικού
ALU, Καταχωρητές
Πύλες
Τρανζίστορ
Ορθογώνια (Rectangles)
Σχέδια Μονάδων (Cell, Module Plans)
Σχέδια Δαπέδου (Floor Plans)
Συμπλέγματα (Clusters)
Φυσικός Διαχωρισμός
(Physical Partitions)
Φυσική/
Γεωμετρική Όψη

Top-Down vs Bottom-Up

- Top-Down: Προχωρά από μια περιγραφή υψηλού επιπέδου προς μια πιο λεπτομερή περιγραφή, με κατάλληλη εφαρμογή της λειτουργίας της αποσύνθεσης.
- Bottom-Up: Ξεκινά από βασικά / πρωταρχικά μπλοκ και τα συνδυάζει για να δημιουργήσει μεγαλύτερα και πιο πολύπλοκα μπλοκ.
- Ο σχεδιασμός πολύπλοκων κυκλωμάτων γίνεται συνήθως και με τους δυο τρόπους, συγχρόνως:
  - Top-Down: επικεντρώνεται στο «Τι σχεδιάζουμε;»
  → Ελέγχει την πολυπλοκότητα.
  - Bottom-Up: επικεντρώνεται στο «Πως το σχεδιάζουμε;»
  → Συγκεντρώνεται στις λεπτομέρειες.

Computer Aided Design (Σχεδιασμός με την βοήθεια Υ/ή)

- Επιτρέπει την αυτοματοποίηση του σχεδιασμού πολύπλοκων κυκλωμάτων και συστημάτων.
- Παραδείγματα εργαλείων CAD:
  - Γραφικοί επεξεργαστές (Graphic / Schematic Editors)
  - Προσομοιωτές λογικής (Logic simulators)
  - Προσομοιωτές χρονισμού (Timing simulators)
  - Σύνθετες λογικής (Logic synthesizers): βελτιστοποίηση χώρου, καθυστέρησης, ισχύος, κτλ.
  - Γλώσσες Προγραμματισμού Υλικού (Hardware Description Languages – HDLs)
  - VHDL, Verilog, ABEL
  - και πολλά άλλα…

Γλώσσες Προγραμματισμού Υλικού

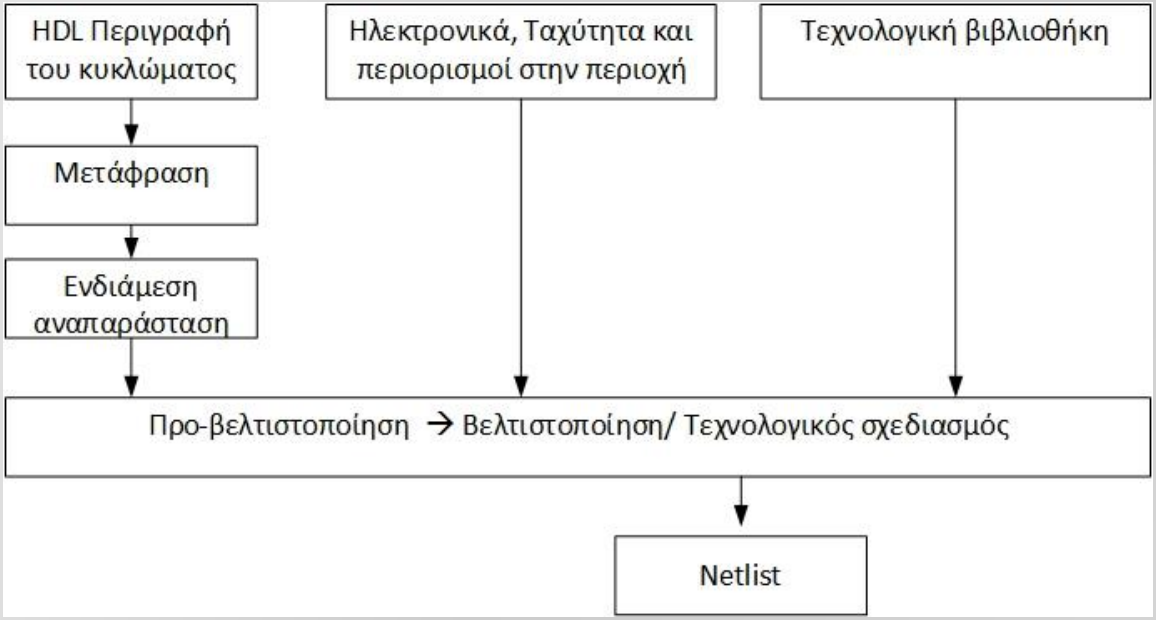
- Γλώσσες προγραμματισμού που έχουν ως κύριο στόχο την περιγραφή της συμπεριφοράς και της δομής υλικού.
- Επιτρέπουν και παράλληλη εκτέλεση ενώ οι γλώσσες προγραμματισμού υψηλού επιπέδου (π.χ. C, C++) επιτρέπουν μόνο σειριακή εκτέλεση.
- VHDL, Verilog
- Χρήση:
  - Εναλλακτική αναπαράσταση από αυτή των γραφικών / σχηματικών περιγραφών (περιγραφή δομής).
  - Χρησιμοποιείται για την αναπαράσταση δυαδικών εξισώσεων, πινάκων αληθείας, k-χάρτες, κτλ. (περιγραφή συμπεριφοράς).
  - Διευκολύνει την διαδικασία της λογικής σύνθεσης.
  - Μπορεί να χρησιμοποιηθεί από διάφορα εργαλεία CAD (σε αντίθεση με τα σχηματικά εργαλεία που διατίθενται από συγκεκριμένους κατασκευαστές).

Λογική Σύνθεση (Logic Design)

- Λογικός συνθέτης:
- Μετατρέπει μια RTL (Register - Transfer Logic) περιγραφή του κυκλώματος σε μια βελτιστοποιημένη περιγραφή πυλών (netlist = περιγραφή κυκλώματος σε επίπεδο-πύλης).

- Ένα netlist μπορεί να μετατραπεί σε μια διάταξη ολοκληρώμενου (IC layout), χρησιμοποιώντας ειδικά εργαλεία σχεδιασμού.

Εργαλεία Λογικής Σύνθεσης

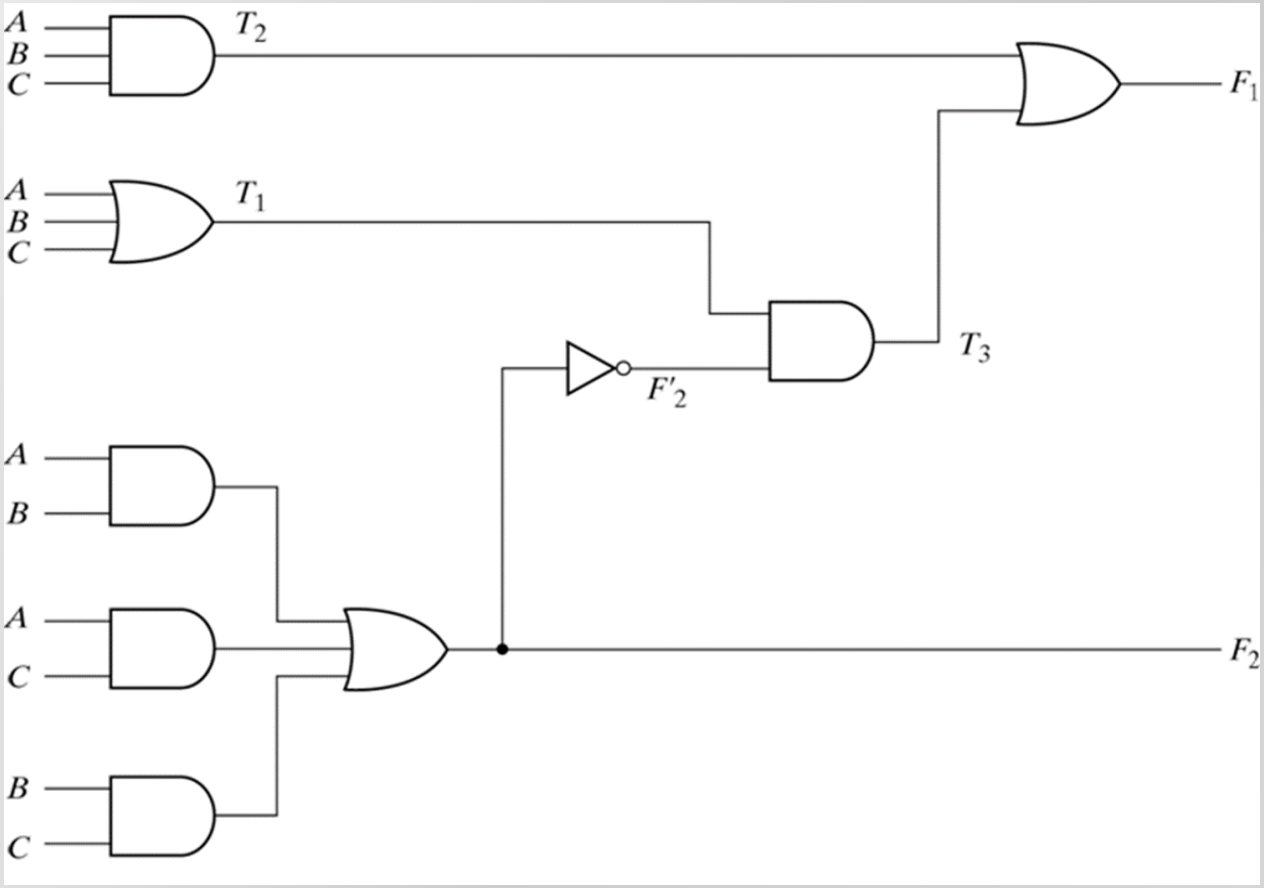


Ανάλυση Συνδυαστικών Κυκλωμάτων

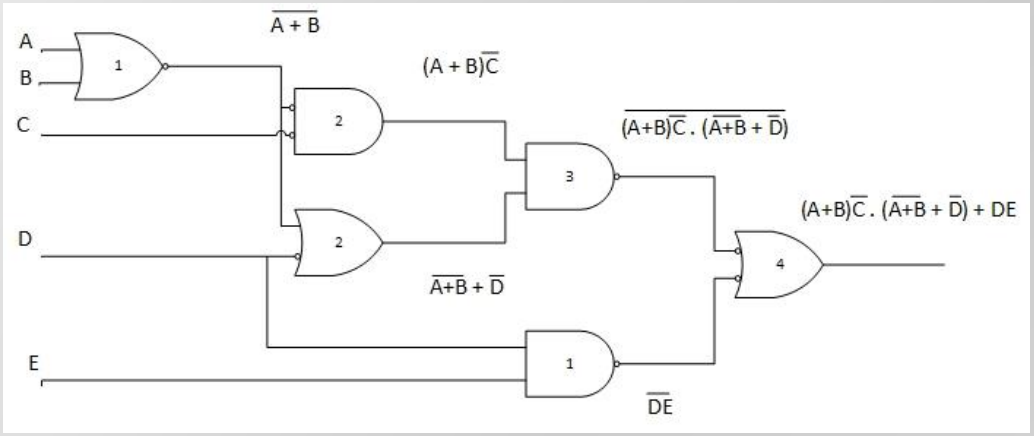
- Ανάλυση = ο καθορισμός και η επαλήθευση των λογικών συναρτήσεων που υλοποιεί το κύκλωμα.

- Η ανάλυση ξεκινά δεδομένου ενός διαγράμματος λογικού κυκλώματος ή περιγραφή συμπεριφοράς, και τελειώνει με:
- Ένα σύνολο από λογικές συναρτήσεις ή
- έναν πίνακα αληθείας.

Παράδειγμα Ανάλυσης



Παραγωγή συναρτήσεων -Παράδειγμα



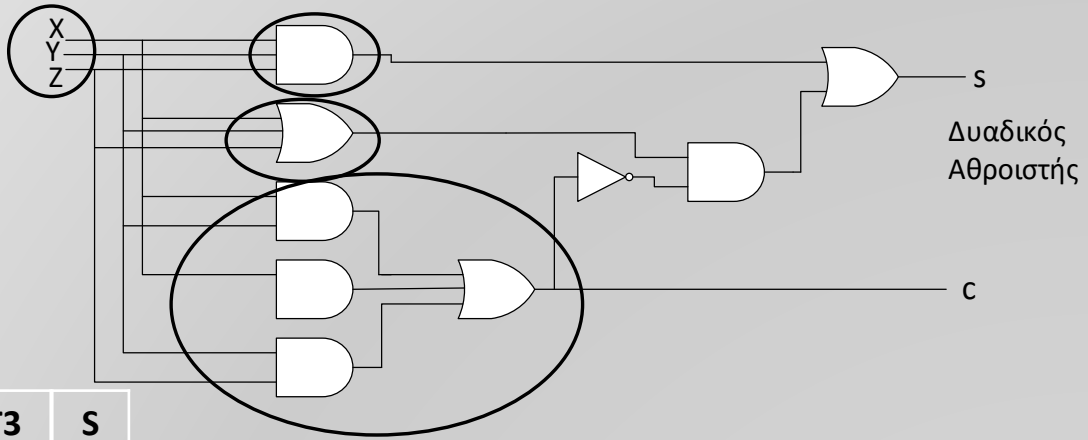
- Ξεκινούμε από τις εισόδους και προχωρούμε προς τις εξόδους (συστηματικά, από επίπεδο-σε-επίπεδο).
- Ονομάζουμε τις ενδιάμεσες συναρτήσεις.
- Ελέγχουμε ξανά τις αντιστροφές.
- Απλοποιούμε (χρησιμοποιώντας το θεώρημα DeMorgan), όπου είναι δυνατόν.

Παραγωγή Πίνακα Αληθείας (1)

- Καθορίζουμε τον # των γραμμών (2^n όπου n είναι ο αριθμός των εισόδων) και τοποθετούμε τους δυαδικούς αριθμούς 0.. 2^n - 1 στον πίνακα.
- Σπάζουμε το κύκλωμα σε κυκλώματα μιας εξόδου και τα ονομάζουμε.
- Βρίσκουμε τον πίνακα αληθείας για κομμάτια που καθοδηγούνται ΜΟΝΟ από εισόδους (ή μικρά / απλά κομμάτια).
- Βρίσκουμε τον πίνακα αληθείας για κομμάτια που καθοδηγούνται από εισόδους ή κομμάτια που ο πίνακας αληθείας τους έχει ήδη υπολογιστεί.

  ΕΠΑΝΑΛΑΜΒΑΝΟΥΜΕ μέχρι να υπολογιστούν οι πίνακες αληθείας για όλες τις εξόδους.

Παραγωγή Πίνακα Αληθείας (2)



X	Y	Z	C	C’	T1	T2	T3	S
0	0	0	0	1	0	0	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	1	0	0	1	0	0
1	1	0	1	0	0	1	0	0
1	1	1	1	0	1	1	0	1

Προσομοίωση (1)

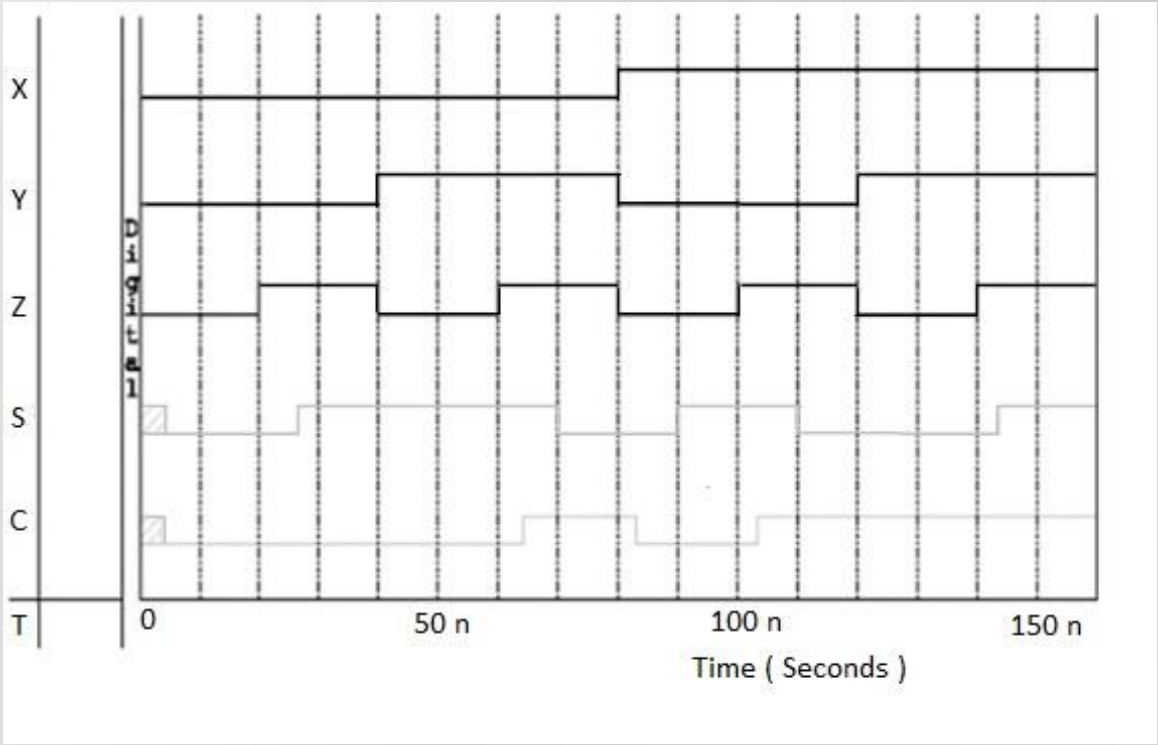
- Χρησιμοποιείται για την ακριβή προσομοίωση και επαλήθευση ενός κυκλώματος. Δεν παράγει συναρτήσεις.
- Το κύκλωμα πρέπει να περιγράφει με τέτοιο τρόπο έτσι ώστε ο προσομοιωτής να μπορεί να το “διαβάσει”:
- Netlist: περιγραφή κυκλώματος σε επίπεδο πυλών (σε μορφή κειμένου).
- HDL περιγραφή.
- Σχηματικά (Schematic) παράγεται από ένα σχηματικό εργαλείο (πχ. Graphic Editor του Max + Plus II).

Προσομοίωση (2)

- Για να γίνει μια προσομοίωση πρέπει ο χρήστης να δώσει τιμές στις εισόδους του κυκλώματος:
- Μέσω του πληκτρολογίου (Interactively – διαδραστικά).
- Μέσω ενός αρχείου (οι τιμές βρίσκονται στο αρχείο).

- Για **πλήρη επαλήθευση** της λειτουργικότητας ενός κυκλώματος χρειάζεται να προσομοιώσουμε όλους τους πιθανούς συνδυασμούς στις εισόδους. Αυτό απαιτεί πολύ χρόνο για κυκλώματα πέρα των 20 (περίπου) εισόδων.

Πλήρη Προσομοίωση - Παράδειγμα Δυαδικού Αθροιστή



Διαδικασία Σχεδιασμού –Συνδυαστικών Κυκλωμάτων

- **Σχεδιασμός:** η ανάπτυξή ενός κυκλώματος δεδομένης μιας περιγραφής της λειτουργίας του.

- Ξεκινά με δεδομένες προδιαγραφές και παράγει ένα επαληθευμένο βέλτιστο λογικό σχεδιασμό συγκεκριμένης τεχνολογίας (συμπεριλαμβάνει το στάδιο της ανάλυσης).

Στάδια διαδικασίας σχεδιασμού

- Προδιαγραφή (Specification)
  - Δίνεται ή ορίζεται (χρησιμοποιείται ως είσοδος ή ως έξοδος).
- Διατύπωση (Formulation)
  - Παραγωγή πίνακα αληθείας ή αρχικών δυαδικών συναρτήσεων που ορίζουν την απαιτούμενη σχέση λειτουργίας εισόδων-εξόδων.
- Βελτιστοποίηση (Optimization)
  - Διεπίπεδη και πολλών επιπέδων.
  - Σχεδιασμός λογικού διαγράμματος ή netlist του κυκλώματος χρησιμοποιώντας πρωταρχικές (primitive) πύλες (AND, OR, NOT).
- Αντιστοίχιση Τεχνολογίας (Technology Mapping)
  - Επιλογή τεχνολογίας υλοποίησης και αντικατάσταση πρωταρχικών πυλών.
- Επαλήθευση (Verification)
  - Επαλήθευση ορθότητας λειτουργίας και χρονισμού με βάση τις αρχικές προδιαγραφές.

Διαδικασία Σχεδιασμού

- Καθορίστε τον απαιτούμενο αριθμό εισόδων και εξόδων.
- Βρείτε τον πίνακα αληθείας που ορίζει τη σχέση λειτουργιάς μεταξύ εισόδων-εξόδων.
- Καθορίστε και ελαχιστοποιήστε τις δυαδικές συναρτήσεις που υλοποιούνται στις εξόδους (K-χάρτες, αλγεβρικοί χειρισμοί, εργαλεία CAD, …). Θεωρήστε πιθανούς περιορισμούς σχεδιασμού (χώρος, καθυστέρηση, ισχύ, διαθέσιμες βιβλιοθήκες, κ.α.)
- Σχεδιάστε το λογικό διάγραμμα.
- Αντιστοίχιση τεχνολογίας (αν δεν έχει ληφθεί υπόψη στο 3).
- Επαληθεύεστε την ορθότητα του σχεδιασμού.

Σχεδιασμός - Παράδειγμα

Σχεδιάστε ένα συνδυαστικό κύκλωμα με 4 εισόδους το οποίο παράγει 1 όταν ο αριθμός των 1 στις εισόδους είναι ίσες με τον αριθμό των 0.

Χρησιμοποιήστε ΜΟΝΟ πύλες NOR 2-εισόδων.

…άσκηση για το σπίτι

Άλλο παράδειγμα – Μετατροπέας Κώδικα

Ένας μετατροπέας κώδικα μετατρέπει από έναν κώδικα σε έναν άλλο, π.χ.:

- BCD-to-Excess-3
- BCD-to-Seven-Segment

Μετατροπέας Κώδικα BCD-to-EXCESS-3 (1)

Προδιαγραφή:

- Σχεδιάστε ένα κύκλωμα που να μετατρέπει ένα κώδικα BCD (Binary - Coded Decimal) στον ανάλογο Excess-3 κώδικα. Η τεχνολογία υλοποίησης υποστηρίζει μόνο πύλες (2NAND, 2NOR και 2-2 AOI).

- Κώδικας Excess-3: Δεδομένου ενός δεκαδικού ψηφίου n, ο ανάλογος excess-3 κώδικας είναι (n + 3) 2.

  Παράδειγμα:

  n=5 → n+3=8 = → 1000excess-3

  n=0 → n+3=3 = → 0011excess-3

  Επιθυμητός κώδικας για δεκαδική αφαίρεση…

Μετατροπέας Κώδικα BCD-to-EXCESS-3 (2)

Input BCD					Output Excess-3			
Decimal Digit	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	1	0
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

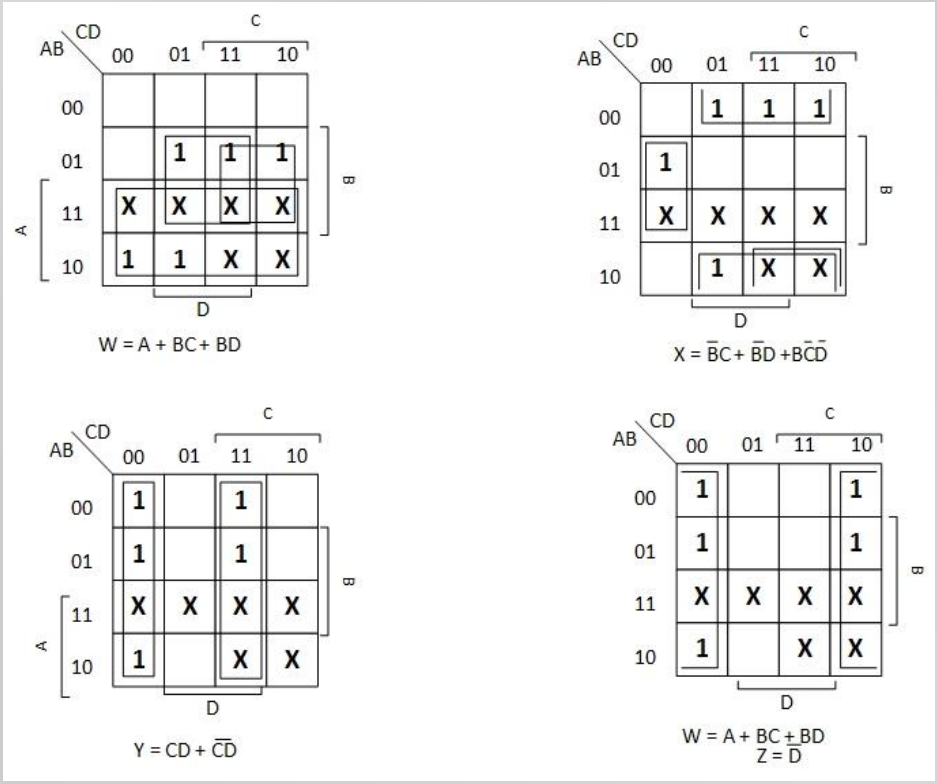
- Οι τιμές των εξόδων από την τιμή εισόδου 1010 εώς 1111 έχουν τιμές αδιαφορίας (και δεν φαίνονται εδώ).

Μετατροπέας Κώδικα BCD-to-EXCESS-3
Βελτιστοποίηση με Κ-χάρτη

- Οι Κ-χάρτες κατασκευάζονται

Χρησιμοποιώντας τους όρους της αδιαφορίας.

→ Βελτιστοποίηση Ατομικής συνάρτησης (διεπίπεδη).



Μετατροπέας Κώδικα BCD-to-EXCESS-3 Βελτιστοποίηση

- Από τους K-χάρτες έχουμε:

$W = A + BC + BD$ $\qquad\qquad$ $W = A + B(C + D)$

$X = B'C + B'D + BC'D'$ Μετασχηματισμοί → $X = B'(C + D) + BC'D'$

$Y = CD + C'D'$ $\qquad\qquad$ $Y = (C D)'$

$Z = D'$ $\qquad\qquad$ $Z = D'$

$f = C + D$ $\qquad\qquad$ $= 1$ OR

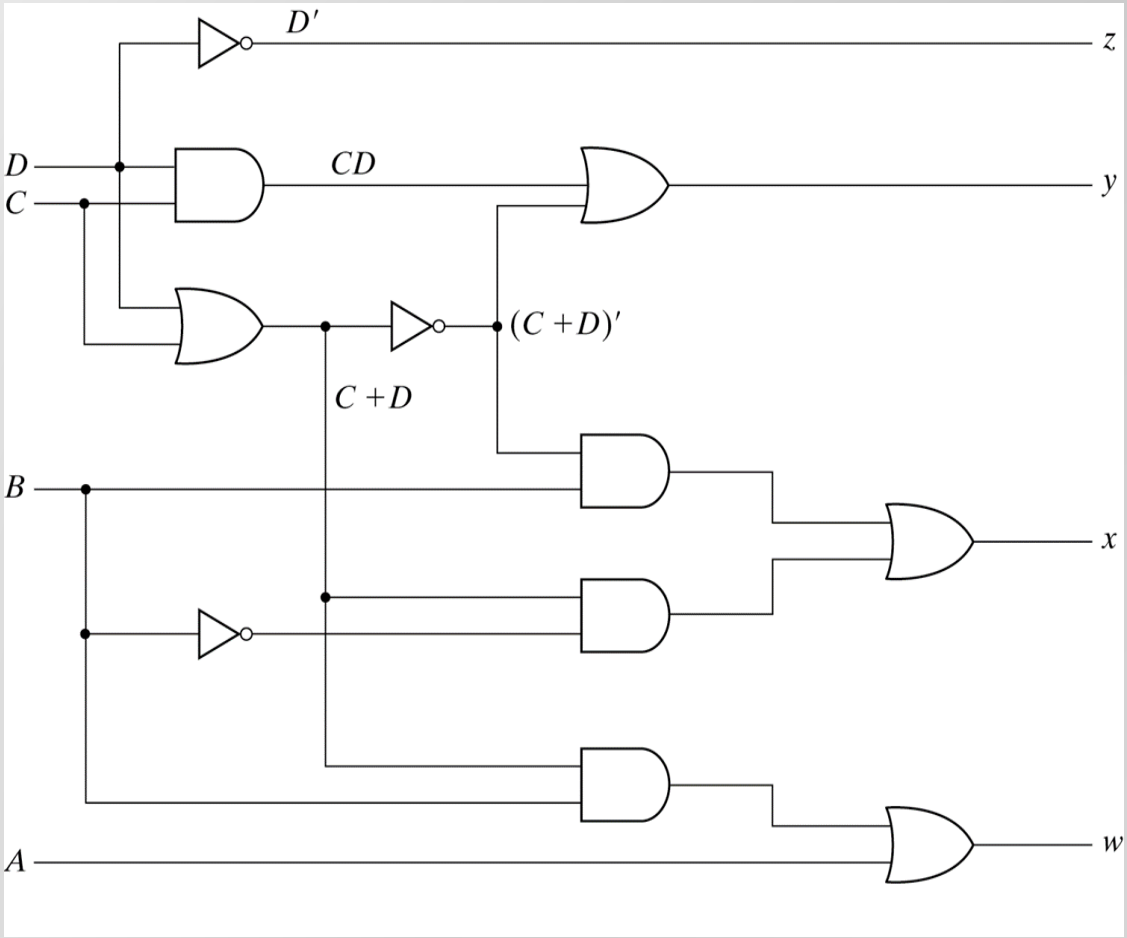
$W = A + Bf$ $\qquad\qquad$ $= 1$ AND, 1 OR

$X = B'f + BC'D'$ $\qquad\qquad$ $= 3$ NOT, 2 AND , 1 OR

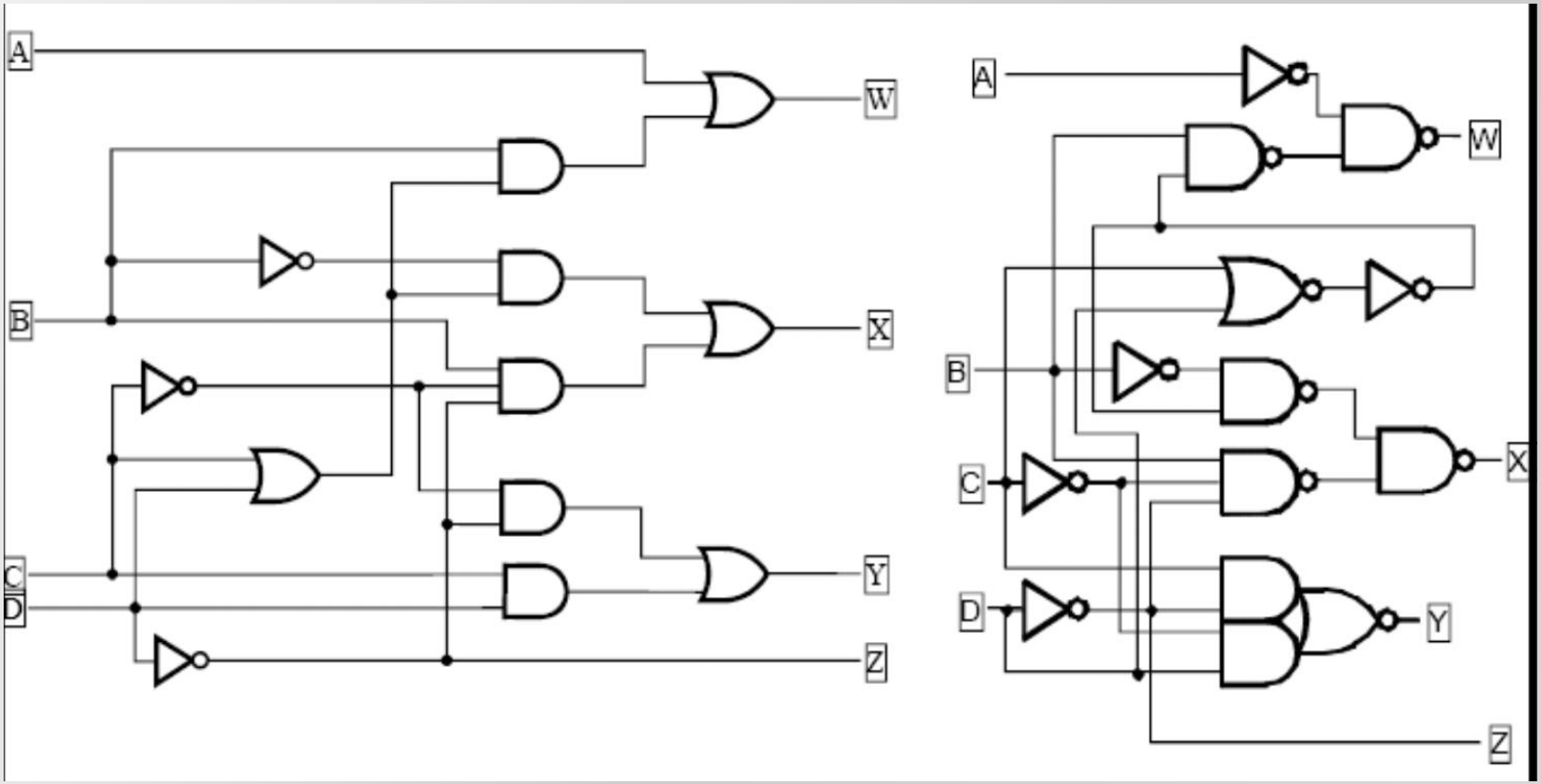
$Y = (C D)'$ $\qquad\qquad$ $= 1$ XNOR

$Z = D'$

Μετατροπέας Κώδικα
BCD-to-EXCESS-3 Βελτιστοποίηση – Σχηματικό Δ.



Μετατροπέας Κώδικα
BCD-to-EXCESS-3 (αντιστοίχιση τεχνολογίας)

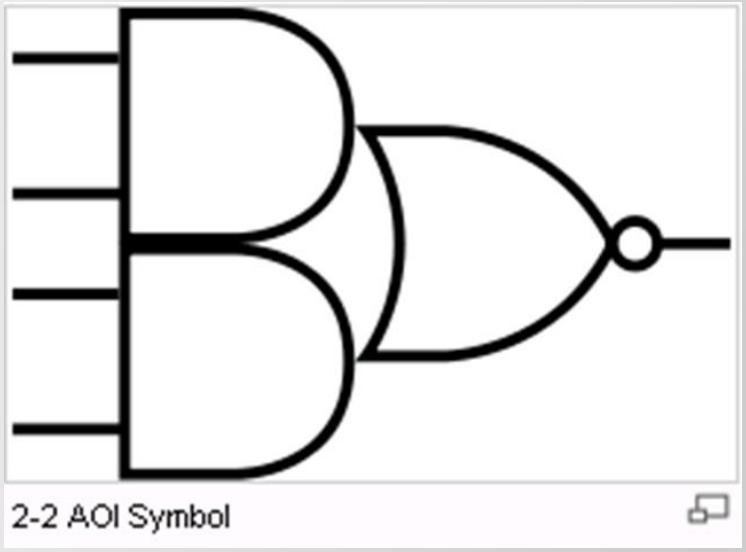


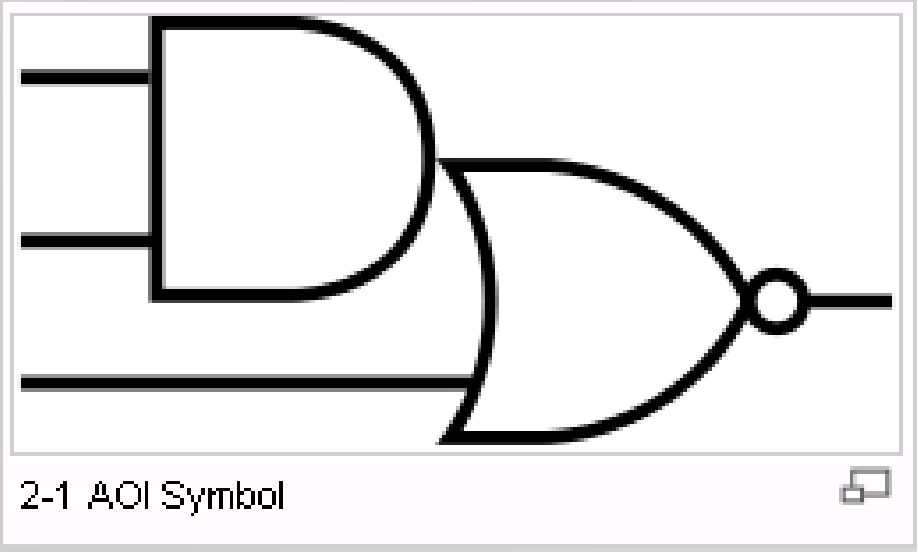
Αντιστοίχιση Τεχνολογίας

- Σχεδιαστικό στυλ για Ολοκληρωμένα.
- Στοιχειά (cells) και βιβλιοθήκες στοιχείων.
- Μεθοδολογίες Αντιστοίχισης
  - Πύλες NAND (βλέπε προηγούμενη διάλεξη).
  - Πύλες NOR (βλέπε προηγούμενη διάλεξη).
  - Πολλαπλών τύπων πύλες.
  - Προγραμματιζόμενες λογικές διατάξεις (βλέπε επόμενες διαλέξεις).

(Τί είναι 2-2 AOI ή 2-1 AOI…;)

- AOI = AND – OR – INVERT
- Complex gates





Σχεδιαστικό Στυλ για IC

- **Πλήρως Προσαρμοσμένο (Full Custom):** Γίνεται όλος ο σχεδιασμός του ολοκληρωμένου, μέχρι την πιο μικρή Λεπτομέρεια:
  - Ακριβό.
  - Δικαιολογείται μόνο για πυκνά (dense), γρήγορα ολοκληρωμένα, με ευρεία χρήση (υψηλές πωλήσεις).
- **Τυποποιημένα Στοιχεία (Standard cell):** Κομμάτια που έχουν σχεδιαστεί από πριν ή είναι κομμάτια προηγούμενων σχεδιασμών.
  - Μεσαίο κόστος.
  - Λιγότερο πυκνά και γρήγορα σε σχέση με τα πλήρη προσαρμοσμένα.
- **Διάταξη Πυλών (Gate Array):** Κανονικές, επαναλαμβανόμενες διατάξεις (regular patterns), οι οποίες μπορούν να χρησιμοποιηθούν σε διαφορετικούς σχεδιασμούς μόνο οι διασυνδέσεις μεταξύ των πυλών προσαρμόζονται για τον κάθε σχεδιασμό.
  - Χαμηλότερο κόστος.
  - Λιγότερη πυκνότητα σε σχέση με τα άλλα 2.

Βιβλιοθήκες Στοιχείων

- Στοιχειό (Cell) - προσχεδιασμένο πρωταρχικό μπλοκ
- Βιβλιοθήκη Στοιχείων – σύνολο στοιχείων που μπορούν να χρησιμοποιηθούν για μια συγκεκριμένη τεχνολογία υλοποίησης.
- Χαρακτηρισμός Στοιχείων (Cell characterization) – λεπτομερής προδιαγραφή των στοιχείων – συχνά βάση του συγκεκριμένου σχεδιασμού, της υλοποίησης του και άλλων μετρίσιμων τιμών.
- Τα στοιχειά χρησιμοποιούνται για σχεδιασμούς gate array, standard cell, και σε κάποιες περιπτώσεις, για full custom.

Επαλήθευση

- Επίδειξη ότι το τελικό κύκλωμα συμπεριφέρεται ακριβώς με τον ίδιο τρόπο που ορίζεται στις προδιαγραφές και τη διατύπωση του κυκλώματος.

- Παραδείγματα διατύπωσης:
  - Πίνακες αληθείας.
  - Δυαδικές συναρτήσεις.
  - Κώδικα-HDL.
- Αν τα πιο πάνω παραδείγματα διατύπωσης δεν είναι μέρος των αρχικών προδιαγραφών, είναι απαραίτητο να επαληθευτεί και η διαδικασία διατύπωσης!

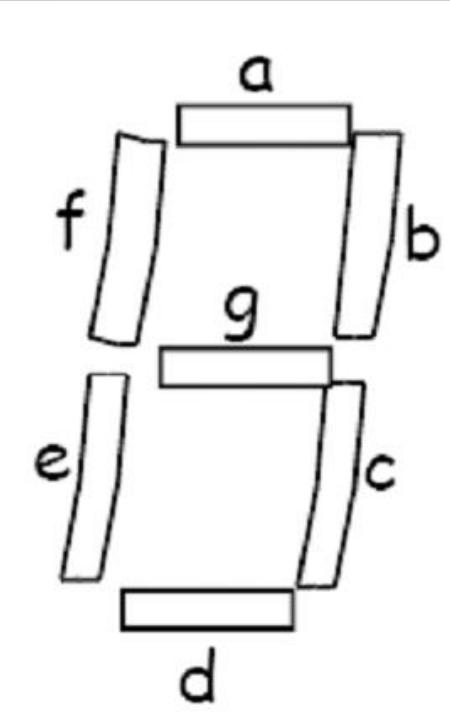
Βασικές Μέθοδοι Επαλήθευσης

- **Θεωρητική Επαλήθευση (Formal Verification)**
  - Παράγουμε τον πίνακα αληθείας ή τις δυαδικές του τελικού κυκλώματος
  - Συγκρίνουμε με τον πίνακα αληθείας ή τις δυαδικές συναρτήσεις των προδιαγραφών / διατύπωσης.
- **Μη Προσομοίωση (Simulation-based Verification)**
  - Προσομοιώνουμε το τελικό κύκλωμα (ή το netlist του, το οποίο συχνά δίνεται σε μορφή HDL των προδιαγραφών), χρησιμοποιώντας κατάλληλες τιμές εισόδων οι οποίες επιβεβαιώνουν πλήρως την ορθότητα του κυκλώματος.
  - Οι κατάλληλες τιμές για ένα συνδυαστικό κύκλωμα είναι όλοι οι πιθανοί συνδυασμοί (εκτός των συνδυασμών αδιαφορίας) – δεν είναι εφικτό για μεγάλα κυκλώματα!

Άλλο παράδειγμα μετατροπέα κώδικα BCD-to-Seven-Segment (1)

Εμφάνιση Seven-Segment:

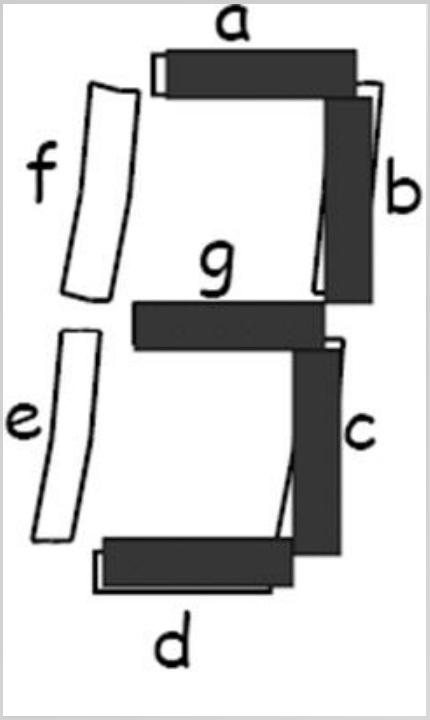
- 7 LED (Light Emitting Diodes), κάθε μια ελέγχεται από μία είσοδο
- Το 1 σημαίνει “ on ” , και το 0 “ off ”



Άλλο παράδειγμα μετατροπέα κώδικα BCD-to-Seven-Segment (2)

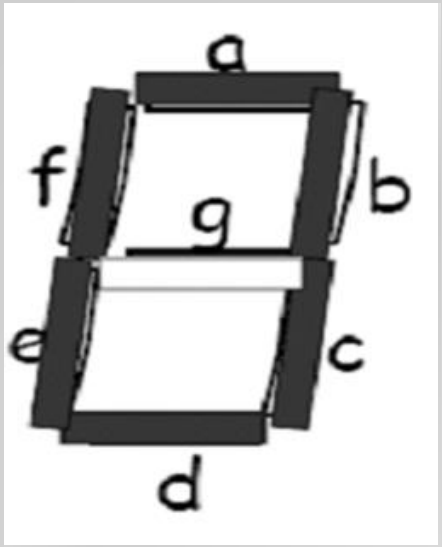
Εμφάνιση Seven-Segment:

- 7 LED (Light Emitting Diodes), κάθε μια ελέγχεται από μία είσοδο.
- Το 1 σημαίνει “ on ” , και το 0 “ off ”.
- Για την εμφάνιση του ψηφίου “ 3 ”.
  - Θέστε a, b, c, d, g σε 1.
  - Θέστε e, f σε 0.



Άλλο παράδειγμα μετατροπέα κώδικα BCD-to-Seven-Segment (3)

- Η είσοδος είναι ένας κώδικας BCD 4ων bit → 4 είσοδοι (w, x, y, z).
- Η έξοδος είναι ένας κώδικας με 7 bits (a, b, c, d, e, f, g), που επιτρέπει στο αντίστοιχο δεκαδικό να εμφανιστεί.
- Παράδειγμα:
  - Είσοδος: 0000_{BCD}
  - Έξοδος : 1111110_{SSD}
  - (a = b = c = d = e = f = 1, g = 0)



Άλλο παράδειγμα μετατροπέα κώδικα BCD-to-Seven-Segment (4)

Ψηφίο	wxyz	abcdefg
0	0000	1111110
1	0001	0110000
2	0010	1101101
3	0011	1111001
4	0100	0110011
5	0101	1011011
6	0110	X011111
7	0111	11100X0

Ψηφίο	wxyz	abcdefg
8	1000	1111111
9	1001	111X011
	1010	XXXXXXX
	1011	XXXXXXX
	1100	XXXXXXX
	1101	XXXXXXX
	1110	XXXXXXX
	1111	XXXXXXX

Άλλο παράδειγμα

- Σχεδιάστε ένα «Συγκριτή Ισοτιμίας» $4^{ων}$ bit
  - Είσοδοι: 2 αριθμοί , 4-bit ο κάθε ένας

    $A(3:0) = A_3\ A_2\ A_1\ A_0$ και $B(3:0) = B_3\ B_2\ B_1\ B_0$
  - Έξοδος : F (1 όταν A=B, 0 στις άλλες περιπτώσεις).
- Άμεσος τρόπος σχεδιασμού (εύκολη σκέψη):

Πίνακας αληθείας με 8 μεταβλητές εισόδων → 128 γραμμές και Quine-McCluskey για βελτιστοποίηση.

- Έμμεσος τρόπος σχεδιασμού: χρήση Ιεραρχίας

$E_i = (A_i)B_i + A_i\ (B_i)' = A_i \oplus B_i$ ($E_i = 1$ εάν $A_i \neq B_i$)

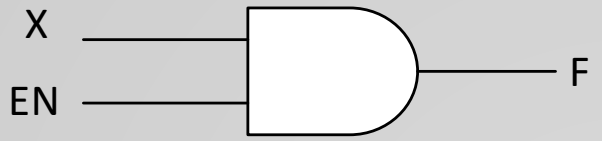
$E = (E_1 + E_2 + E_3 + E_4)'$

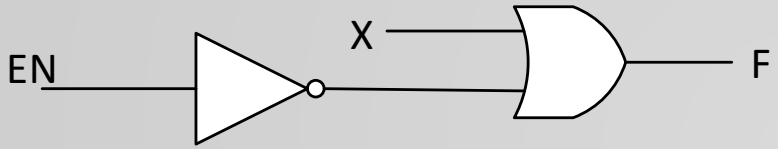
Συναρτήσεις και Λειτουργικές Μονάδες

- Εξετάζουμε βασικές συναρτήσεις που χρησιμεύουν στο σχεδιασμό ψηφιακών κυκλωμάτων.
- Σε κάθε συνάρτηση αντιστοιχεί μια υλοποίηση συνδυαστικού κυκλώματος που αναφέρετε ως λειτουργική μονάδα.
- Στο παρελθόν, πολλές λειτουργικές μονάδες υλοποιούνταν ως κυκλώματα τεχνολογίας SSI, MSI και LSI.
- Σήμερα, συχνά, είναι μέρος (κομμάτια) των κυκλωμάτων τεχνολογίας VLSI.

Συνάρτηση Ενεργοποίησης

- Ενεργοποίηση: επιτρέπει ένα σήμα εισόδου να περάσει στην έξοδο.
- Απενεργοποιήση: εμποδίζει ένα σήμα εισόδου να περάσει στην έξοδο, αντικαθιστώντας το με μια σταθερή τιμή.
- Η τιμή μιας απενεργοποιημένης εξόδου μπορεί να είναι Hi-Z (όπως σε tri-state buffers και πύλες μετάδοσης), 0 ή 1, αναλόγως της σύμβασης.
- Όταν EN = 0, F = 0
- Όταν EN = 0 , F = 1





Αθροιστής 1bit

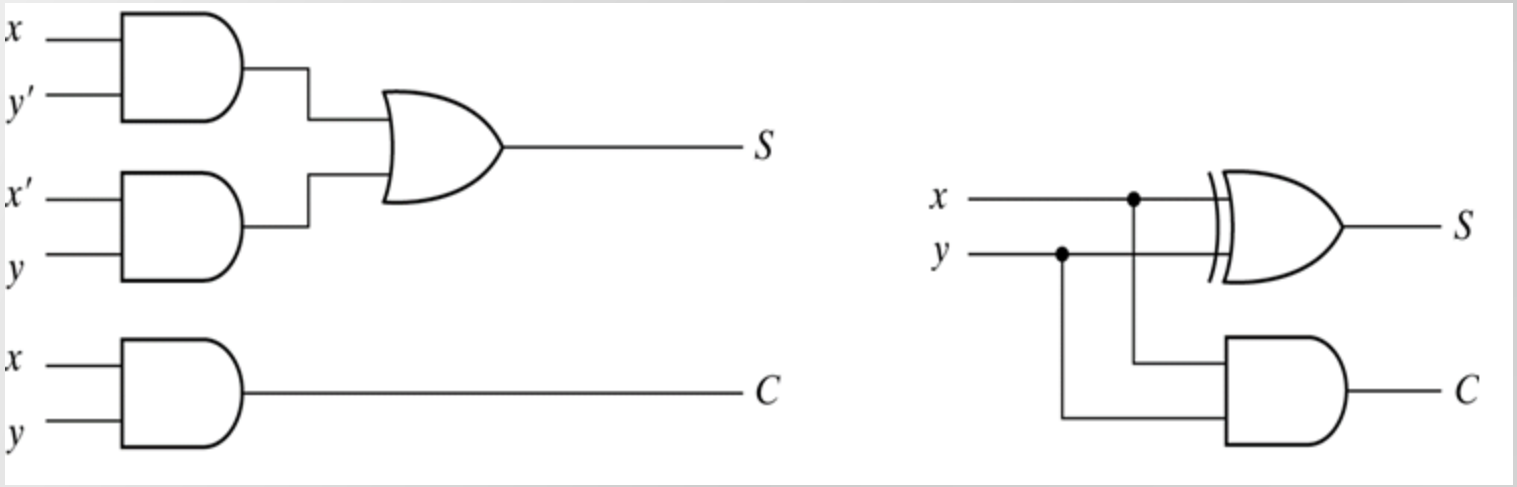
- Εκτελεί πρόσθεση μεταξύ δυο bits.
- Τέσσερις πιθανές πράξεις:
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 10
- Η υλοποίηση του κυκλώματος απαιτεί 2 εξόδους, η μια για το **άθροισμα (sum)** και η άλλη για το **κρατούμενο (carry)**.

Ημι-αθροιστής (1)

- Εκτελεί πρόσθεση μεταξύ δυο bit.
- Είσοδοι: A_0, B_0
- Έξοδοι: S_0 , C_1
- Ο δείκτης υποδεικνύει σημαντικότητα, 0 για LSB και 1 για το επόμενο σημαντικό bit.
- Δυαδικές συναρτήσεις:
  - $S_0 = A_0 B_0' + A_0' B_0 = A_0 \oplus B_0$
  - $C_1 = A_0 B_0$

A_0	B_0	S_0	C_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ημι-αθροιστής (2)



(a) S = xy’ + x’y

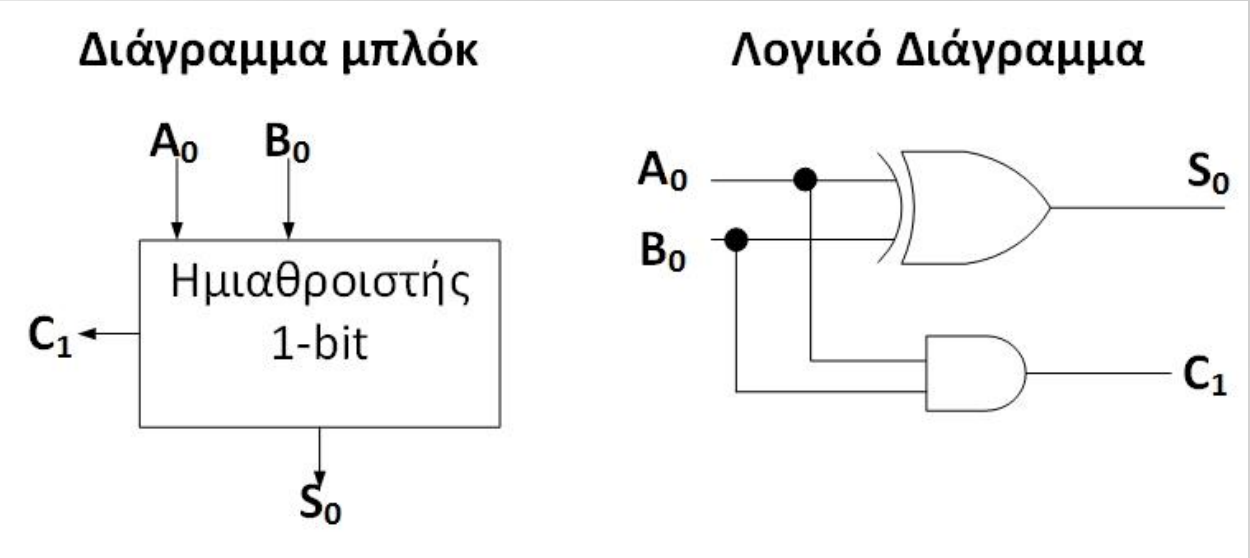
C = xy

(b) S = x ⊕ y

C = xy

Ημι-αθροιστής (3)

- $S_0 = A_0 B_0' + A_0' B_0 = A_0 \oplus B_0$
- $C_1 = A_0 B_0$



Πρόσθεση n-bit

- Σχεδιάστε ένα δυαδικό αθροιστή ο όποιος προσθέτει δυο n-bit δυαδικούς αριθμούς και παράγει ένα άθροισμα (sum) με n-bit και ένα κρατούμενο εξόδου (carry out) με 1-bit.

- Παράδειγμα: Θεωρήστε n = 4

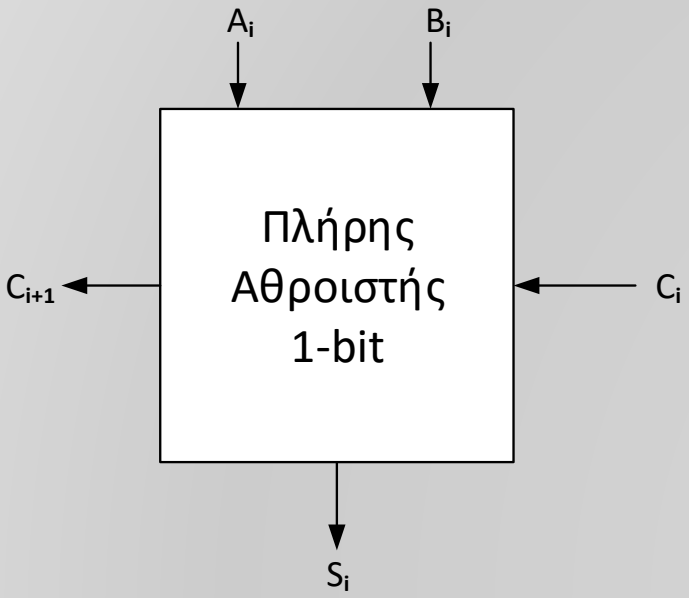
C_{out} C_3	C_2	C_1	C_0	
A_3	A_2	A_1	A_0	
+ B_3	B_2	B_1	B_0	
S_3	S_2	S_1	S_0	

1	1	0	1	0
	1	1	0	1
+	1	1	0	1
	1	0	1	0

- Αυτό απαιτεί πρόσθεση $3^{ων}$-bit!

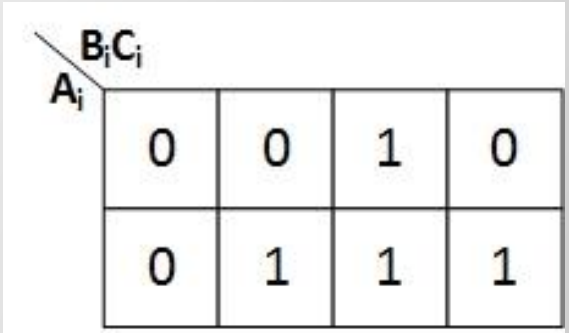
Πλήρης αθροιστής 1-bit (1)

- Συνδυαστικό κύκλωμα που διεκπεραιώνει την πρόσθεση μεταξύ 3ων (2bits προσθετέων και 1 bit για κρατούμενο εισόδου – carry-in).

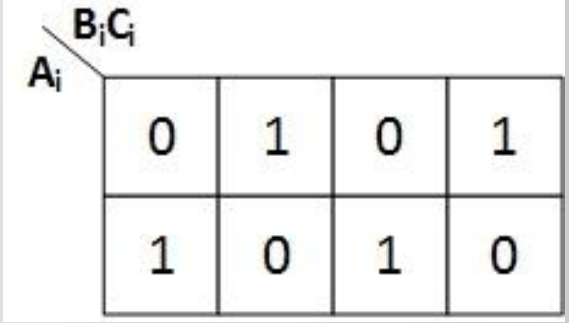


Πλήρης αθροιστής 1-bit (2)

- Οι K-χάρτες για:
- C_{i+1} :



- S_i :

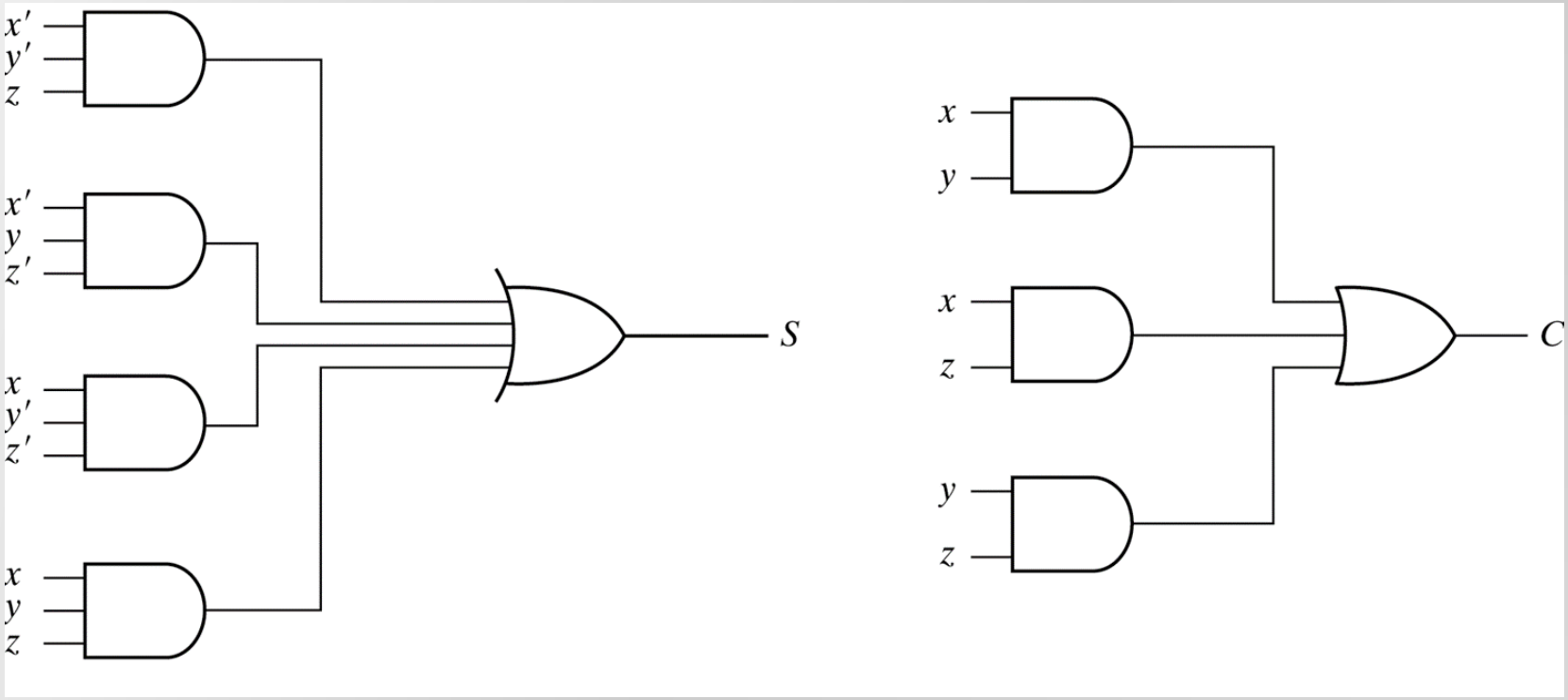


A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Πλήρης αθροιστής 1-bit (3)

- Δυαδικές συναρτήσεις:
- $C_{i+1} = A_i B_i + A_i C_i + B_i C_i$
- $S_i = A_i B_i' C_i' + A_i 'B_i' C_i + A_i ' B_i C_i' + A_i B_i C_i =$
  $= A_i \oplus B_i \oplus C_i$
- Μπορείτε να σχεδιάσετε ένα πλήρη αθροιστή <u>άμεσα</u> από τις πάνω συναρτήσεις. (απαιτούνται 3 πύλες AND και 1 πύλη OR για το $C_{i+1,}$ και 2 πύλες XOR για το S_i).
- Υπάρχει καλύτερη υλοποίηση;

Υλοποίηση πλήρους αθροιστή σε μορφή αθροίσματος γινομένων

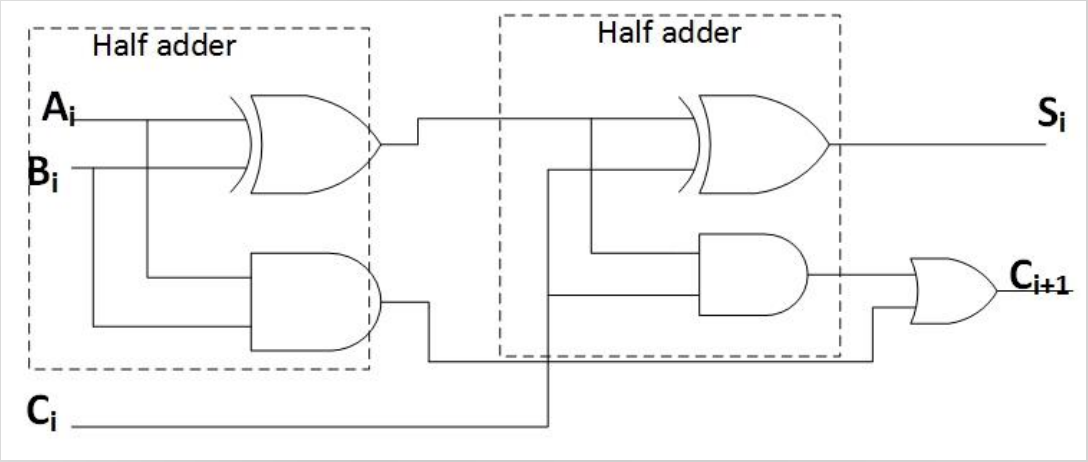


Πλήρης αθροιστής με 2 ημι-αθροιστές

- Ένας πλήρης αθροιστής μπορεί να υλοποιηθεί και με 2 ημι-αθροιστές και 1 πύλη OR, αφού το C_{i+1} μπορεί να εκφραστεί ως:
- $C_{i+1} = A_i B_i + A_i B_i' C + A_i' B_i C_i$

  $= A_i B_i + (A_i B_i' + A_i' B_i) C_i$

  $= A_i B_i + (A_i \oplus B_i) C_i$
- Και το $S_i = A_i \oplus B_i \oplus C_i$

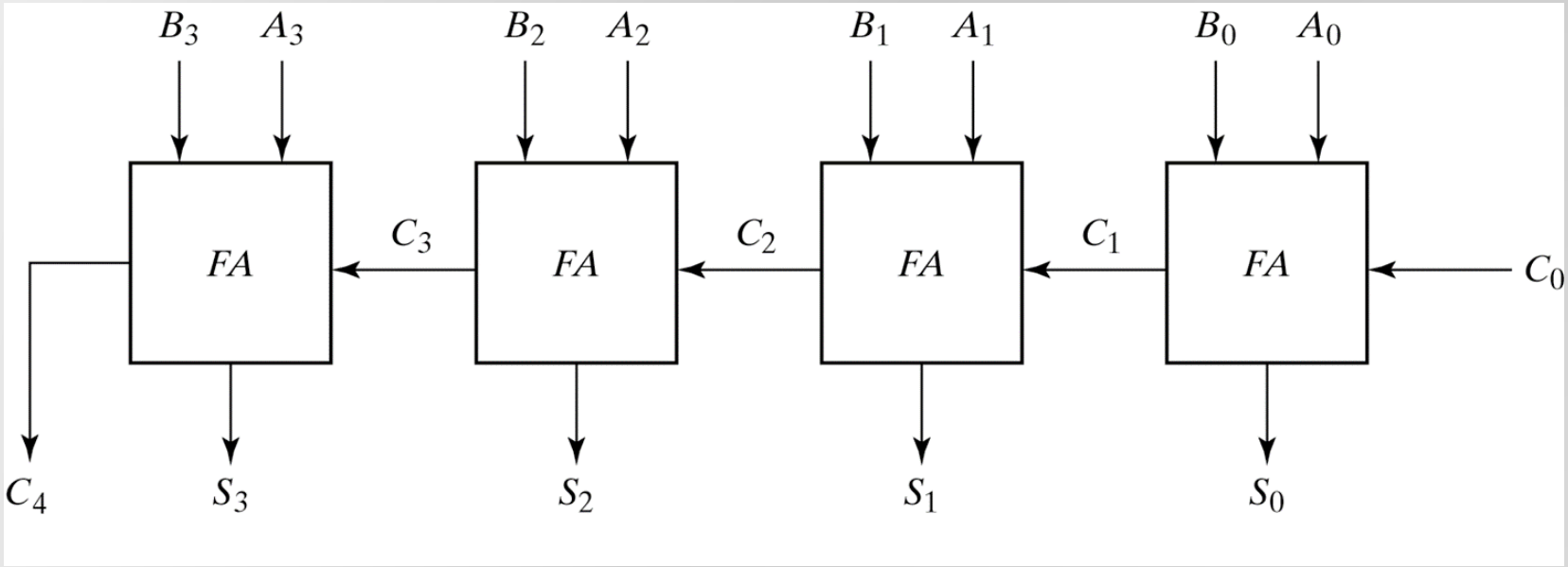


Δυαδικός αθροιστής

- Ψηφιακό κύκλωμα που παράγει το αριθμητικό άθροισμα δύο δυαδικών αριθμών.
- Μπορεί να κατασκευαστεί με πλήρεις αθροιστές σε παραθετική σύνδεση.

Δυαδικός αθροιστής 4bit (ή αθροιστής ριπής – ripple adder)

- Ο σχεδιασμός αυτού του κυκλώματος θα απαιτούσε 2^9 = 512 γραμμές σε πίνακα αληθείας.



Συνδυαστικοί αθροιστές n-bit

- Εκτελούν παράλληλη πρόσθεση πολλαπλών-bit.

1. Αθροιστής Ριπής (Ripple Carry Adder).

- Απλός σχεδιασμός.
- Χρονοβόρος. Γιατί; (θα δείτε σε λίγο!)

2. Αθροιστής Πρόβλεψης Κρατούμενου (Carry Lookahead Adder).

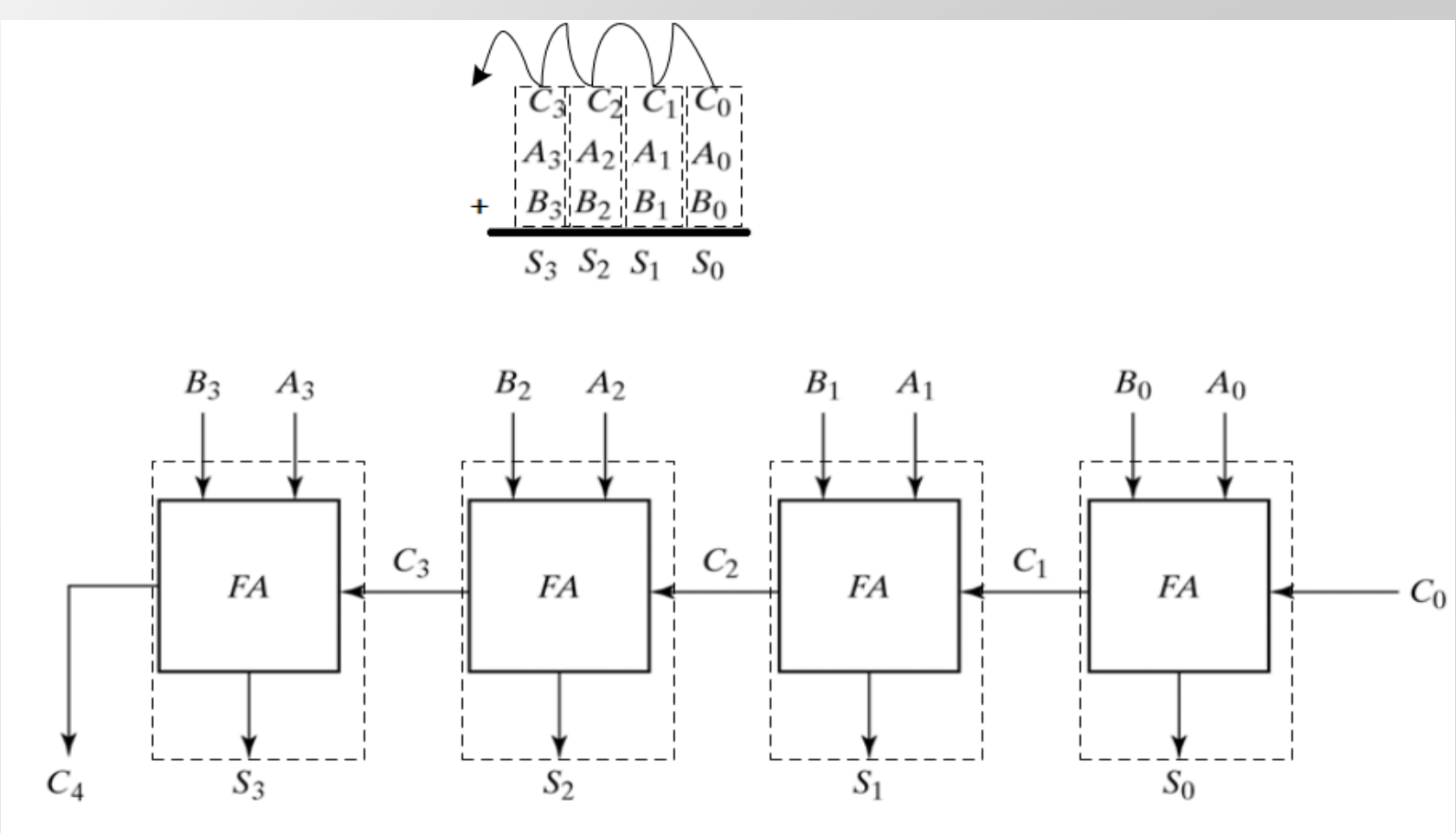
- Πιο πολύπλοκος σχεδιασμός.
- Μειώνει την καθυστέρηση του κυκλώματος.

Αθροιστής ριπής n-bit

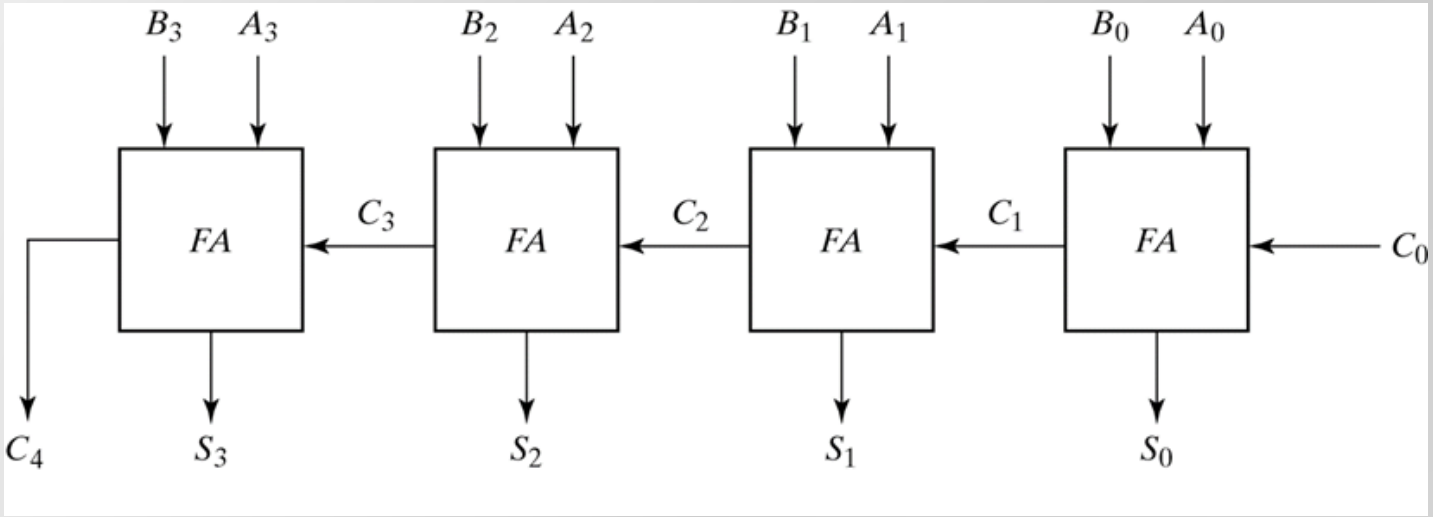
- Κατασκευάζεται με n πλήρες αθροιστές 1-bit, δομημένοι παράλληλα.

- Ο ένας πλήρης αθροιστής 1-bit διαδέχεται τον άλλο, έτσι ώστε το κρατούμενο εξόδου (carry out) από το ένα γίνεται το κρατούμενο εισόδου (carry in) του επόμενου.

Παράδειγμα: Αθροιστής ριπής 4bit



Καθυστέρηση αθροιστή ριπής



- Η καθυστέρηση του κυκλώματος ενός αθροιστή ριπής καθορίζεται από την καθυστέρηση του μονοπατιού του κρατούμενου από το LSB (C_0) στο MSB (C_n).
- Θεωρήστε την καθυστέρηση σε ένα 1-bit FA να είναι Δ. Τότε, η καθυστέρηση του αθροιστή ριπής n-bit είναι nΔ.

Διάδοση κρατουμένου

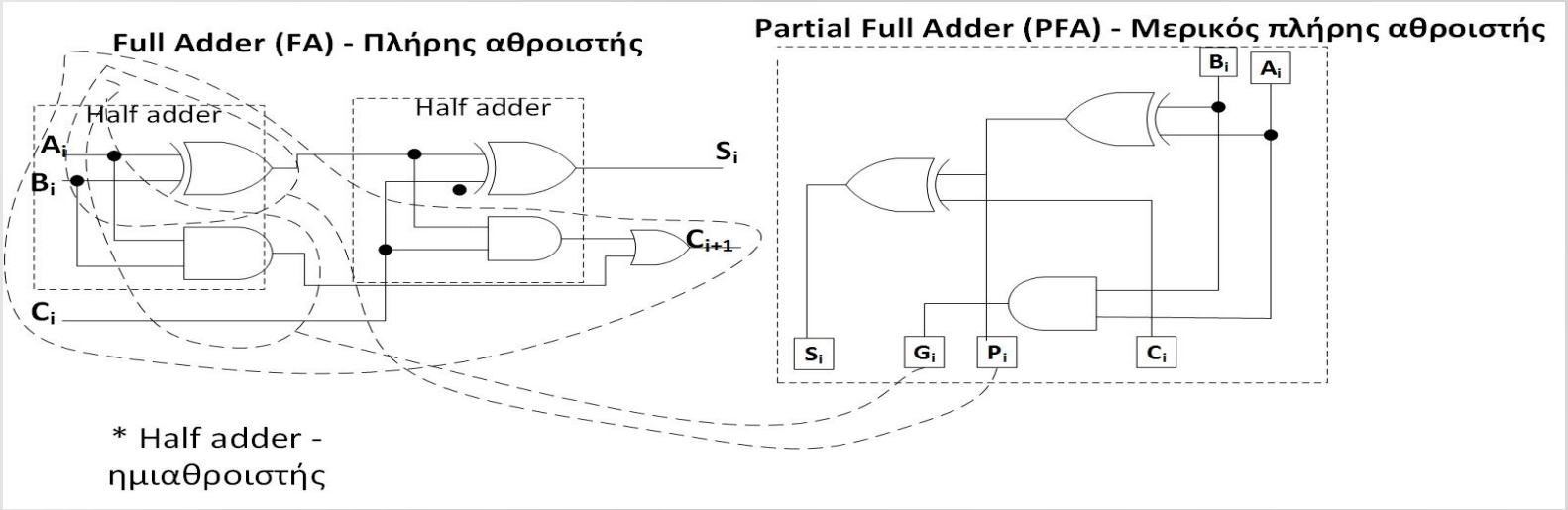
- Ο χρόνος διάδοσης κρατουμένου είναι περιοριστικός παράγοντας της ταχύτητας πρόσθεσης.
- Οι έξοδοι δε θα είναι σωστές αν δεν δοθεί αρκετός χρόνος στα σήματα να διαδοθούν.
- Μπορεί να αυξηθεί η πολυπλοκότητα και να μειωθεί ο χρόνος διάδοσης…

Αθροιστής Πρόβλεψης Κρατουμένου (carry look-ahead adder)

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

Σχεδιασμός CLA (1)

- Από ένα FA, διαχωρίζουμε μεταξύ της παραγωγής (generation) του κρατουμένου (όταν ένα νέο κρατούμενο παράγεται, C_{out} =1) και της μετάδοσης (propagation) του κρατούμενου (όταν υπάρχον C_{in} μεταδίδεται στο C_{out}).
- Παραγωγή : $G_i = A_i B_i$: if 1 , $C_{i+1} = 1$
- Μετάδοση : $P_i = A_i \oplus B_i$: εάν 1 τότε $C_{i+1} = C_i$

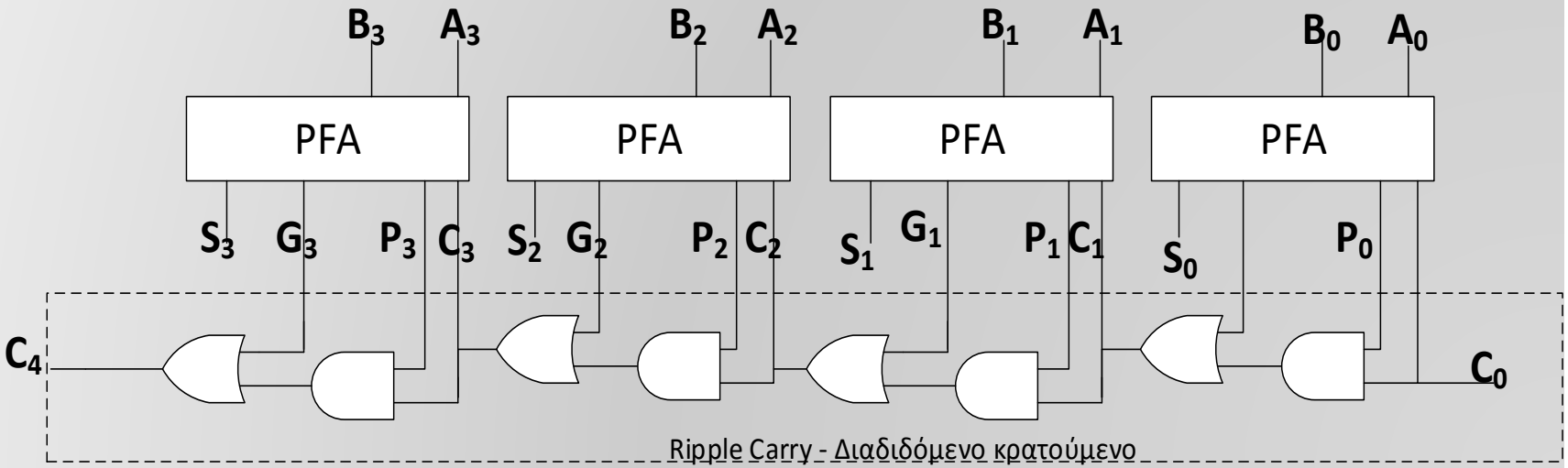


Σχεδιασμός CLA (2)

- Ένα bit από λογική G / P μόνο δεν βοηθά, αλλά…
- Διαδοχική λογική G / P μπορεί να παράγει το κρατούμενο εξόδου ενός μπλοκ.

Σχεδιασμός CLA (3)

- $C_{i+1} = G_i + P_i C_i$
- Ο σχεδιασμός του PFA διαχωρίζει την λειτουργικότητα (και άρα την υλοποίηση) του S από αυτή του G / P .



Σχεδιασμός CLA (4)

- Μπορεί ο σχεδιασμός της προηγούμενης διαφάνειας να λύσει το πρόβλημα της μεγάλης καθυστέρησης;

- Όχι, το κρατούμενο εξόδου συνεχίζει την «κυμάτωση».
- Ιδέα: χρήση δυο επίπεδων λογικής για την παραγωγή του κρατούμενου εξόδου από οποιοδήποτε μπλοκ C_i βάση του κρατούμενου εισόδου C_0 και των πρόσθετων bits A_i and B_i .

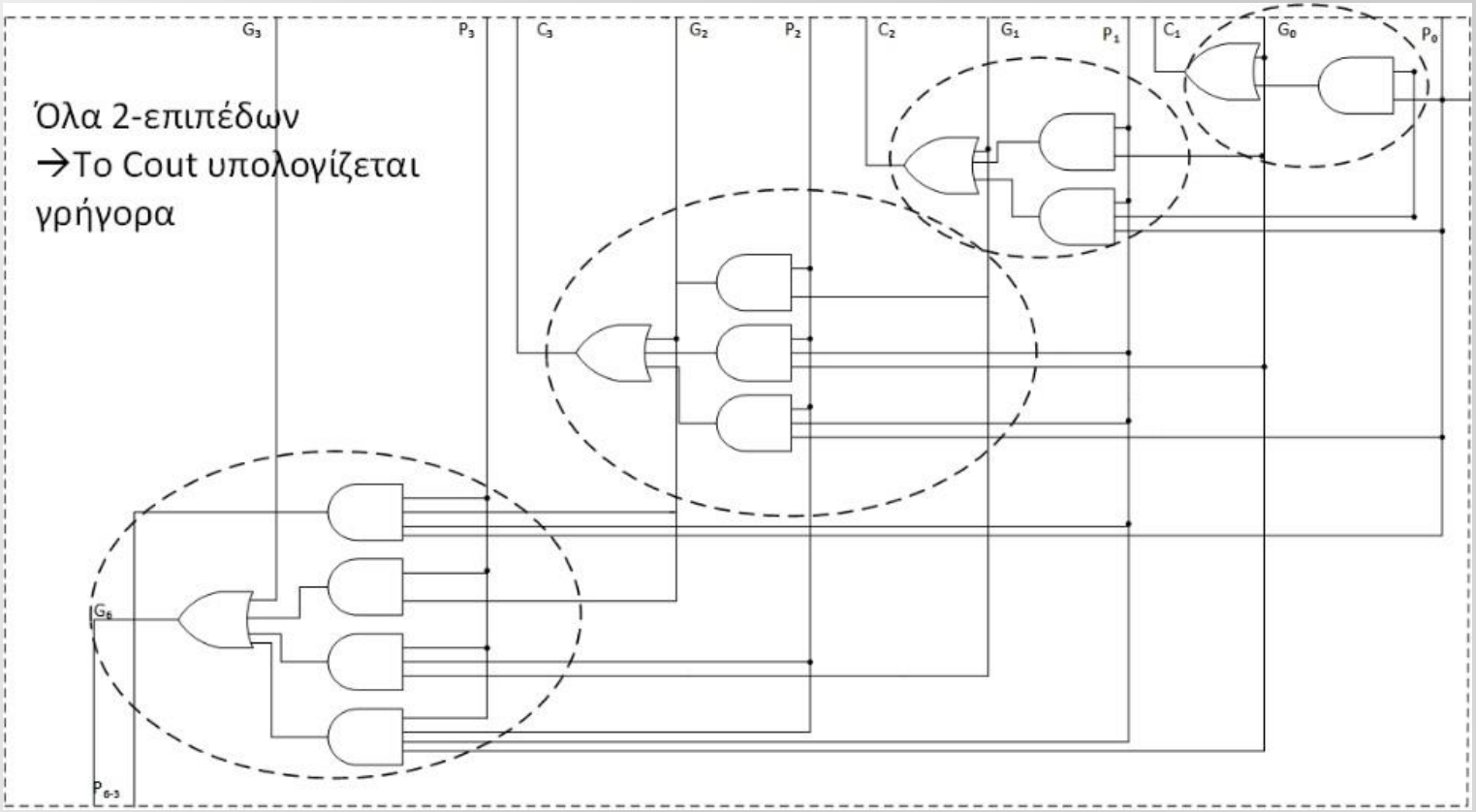
Μπλοκ CLA

Υλοποίηση:

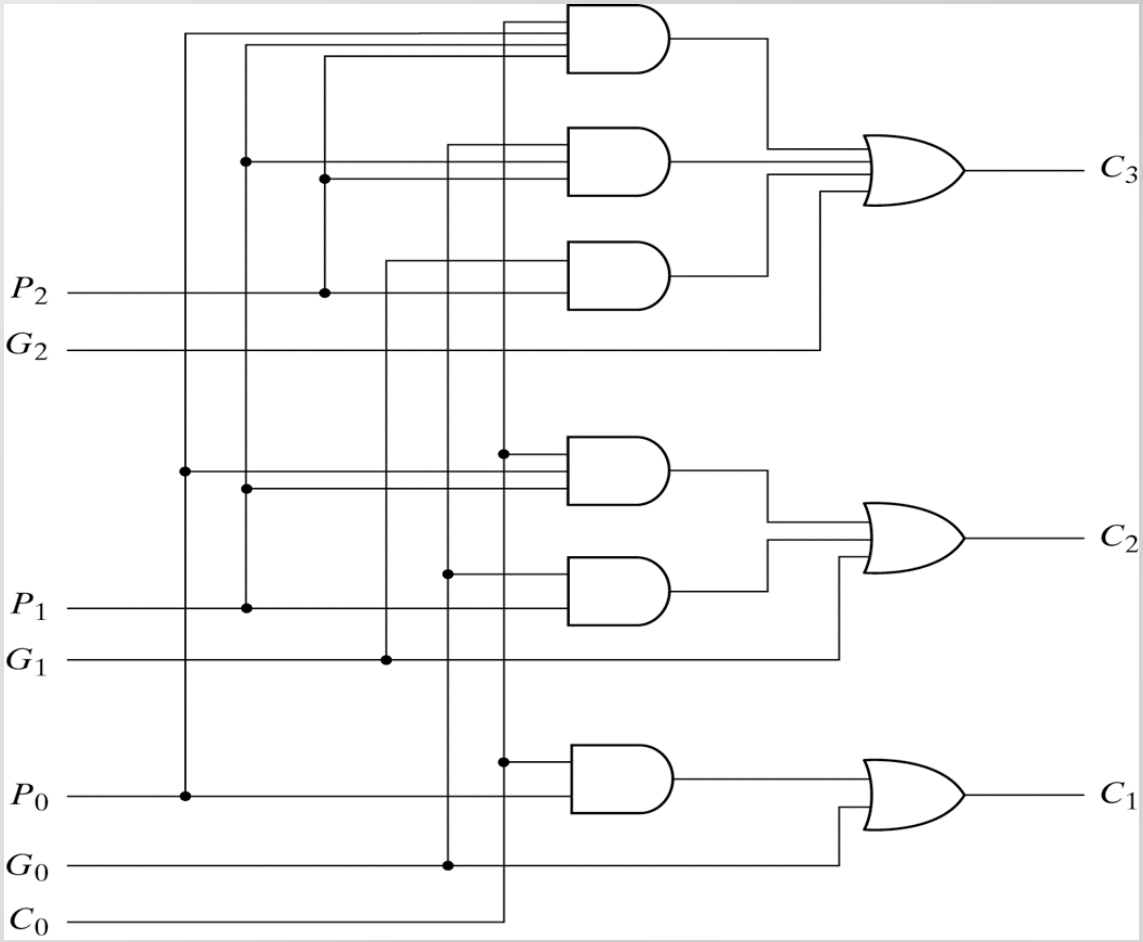
- $C_1 = G_0 + P_0C_0$
- $C_2 = G_1 + P_1C_1 = G_1 + P_1 (G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$
- $C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
- $C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

- G_{0-3} → Ομάδα Παραγωγής Κρατουμένου.
- P_{0-3} → Ομάδα Μετάδοσης Κρατουμένου.

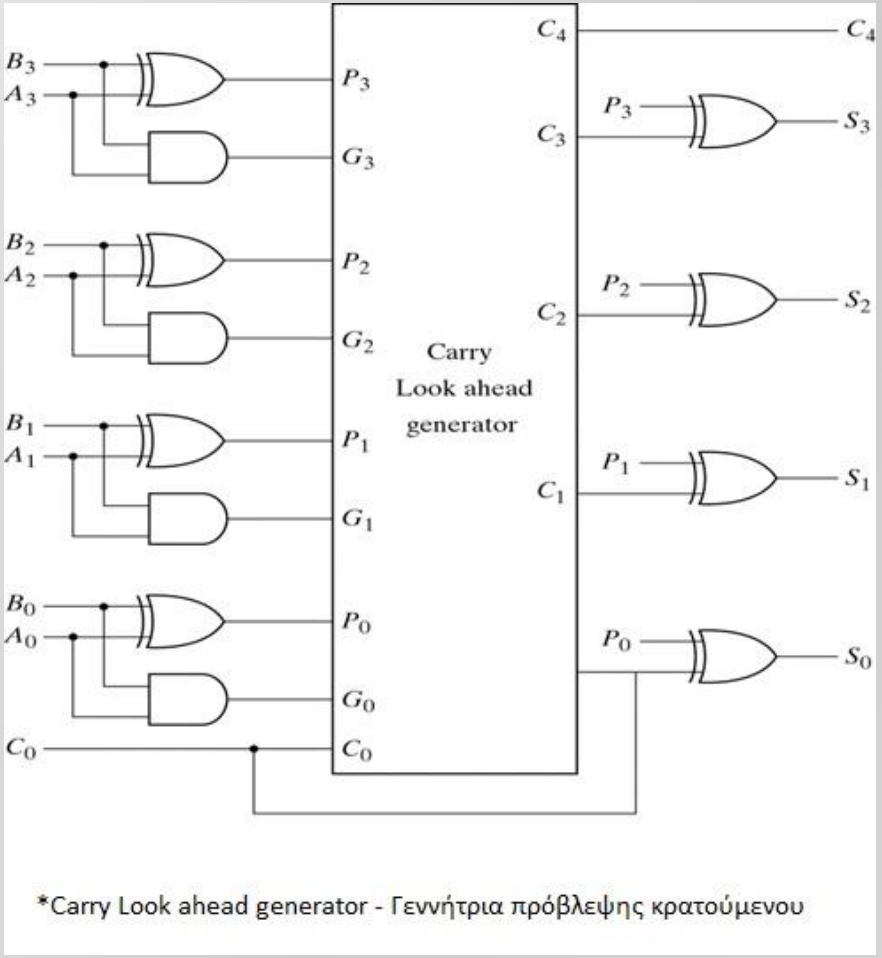
Λογική Παραγωγής - Μετάδοσης 4bit CLA



Λογικό διάγραμμα γεννήτριας πρόβλεψης κρατουμένου



Αθροιστής 4 bit με πρόβλεψη κρατουμένου



*Carry Look ahead generator - Γεννήτρια πρόβλεψης κρατούμενου

Τέλος Ενότητας