ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
ΕΛΛΑΔΑ 2008
Ανάπτυξη παντού. Ανάπτυξη για όλους.
Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης

Design Representations & Transformations

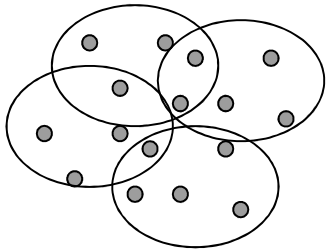
Βασικοί Ορισμοί Συνόλων

Set: συλλογή από στοιχεία, {}:σύνολο χωρίς διάταξη, (): με διάταξη

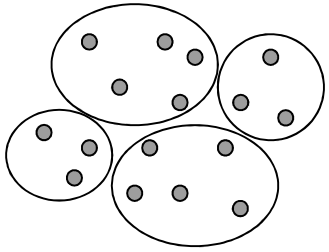
Set Cardinality: αριθμός στοιχείων συνόλου $|S|$.

Set Cover (κάλυψη): είναι ένα σύνολο υποσυνόλων του S των οποίων η ένωση δίνει το S.

Partition (διαμέριση) του S: κάλυψη από ξένα υποσύνολα (blocks).



Κάλυψη



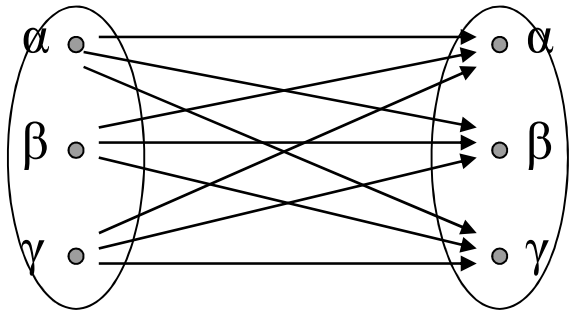
Διαμέριση

Βασικοί Ορισμοί Συνόλων

Καρτεσιανό γινόμενο δύο συνόλων XxY: το σύνολο όλων των διατεταγμένων ζευγών (x,y) με $x \in X$, $y \in Y$.

Relation R (σχέση) μεταξύ δύο συνόλων X, Y: είναι ένα υποσύνολο του καρτεσιανού γινομένου των συνόλων (xRy).

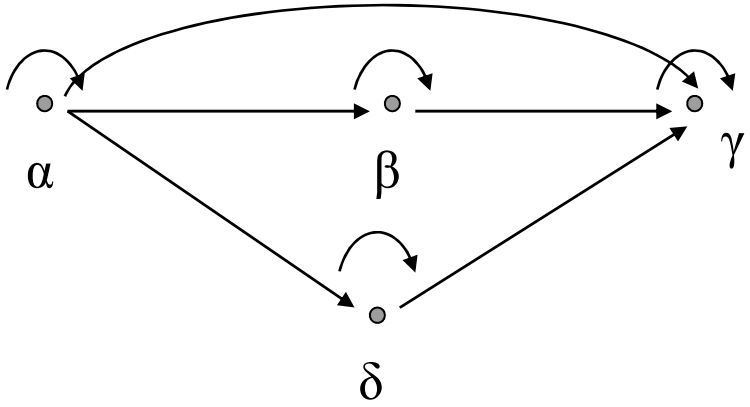
Σχέση ισοδυναμίας R: ανακλαστική [$(x, x) \in R$], συμμετρική [$(x, y) \in R \Rightarrow (y, x) \in R$], μεταβατική [$(x, y) \in R$ και $(y, z) \in R \Rightarrow (x, z) \in R$].



Βασικοί Ορισμοί Συνόλων

Partial Order (Μερική διάταξη): σχέση ανακλαστική, αντισυμμετρική [(x, y)$\in$R και (y, x)$\in$R$\Rightarrow$ y = x], και μεταβατική

Partial Order Set (Μερικά διατεταγμένο σύνολο): συνδυασμός ενός συνόλου και μίας μερικής διάταξης πάνω στο σύνολο.



Βασικοί Ορισμοί Συνόλων

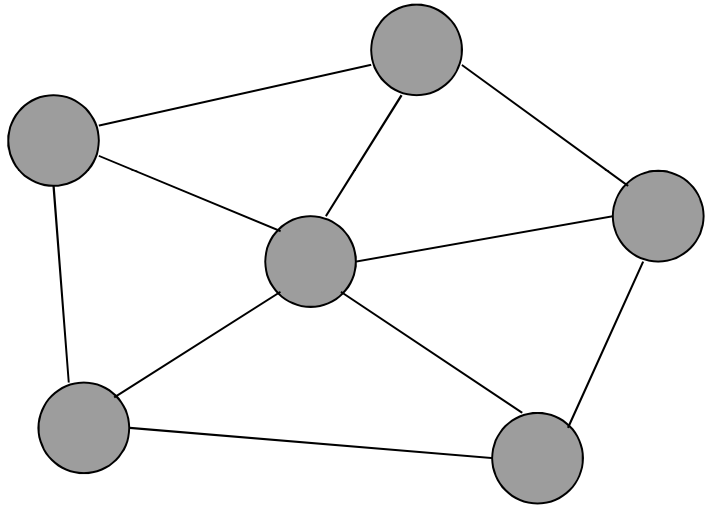
Συνεπής απαρίθμηση ενός μερικά διατεταγμένου συνόλου S είναι η ανάθεση ακεραίων i(S) έτσι ώστε $s_a R s_b$ συνεπάγεται $i(s_a)<i(s_b)$

Domain, co-Domain μίας συνάρτησης f: τα σύνολα X, Y αντίστοιχα με $f:X \rightarrow Y$.

Image υποσυνόλου A του Domain X υπό την f: $f(A)=\{f(x): x \in A\}$

Βασικοί Ορισμοί Γράφων

Γράφος G(V, E): είναι ένα ζεύγος V, E όπου V είναι ένα σύνολο και E μία δυαδική σχέση στο V. Τα στοιχεία του V ονομάζονται κορυφές, ενώ τα στοιχεία του E ακμές.

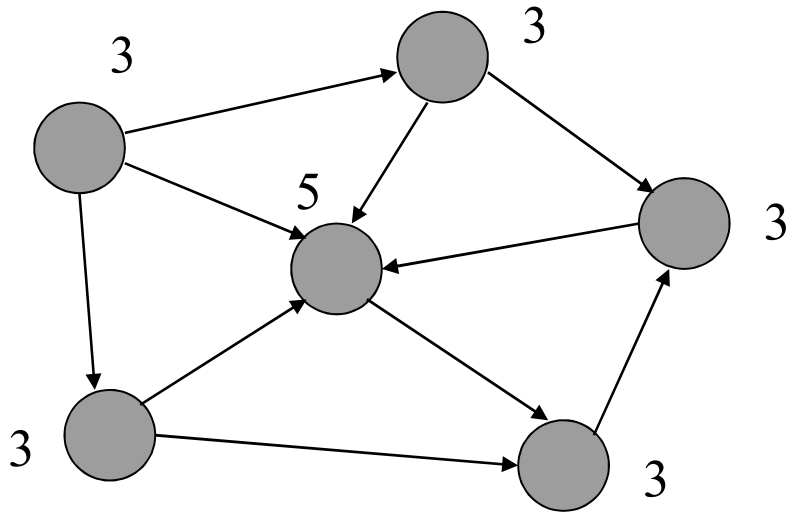


Βασικοί Ορισμοί Γράφων

Κατευθυνόμενος γράφος: οι ακμές είναι διατεταγμένα ζεύγη κορυφών.

Μια ακμή **προσπίπτει** (**incident**) σε μία κορυφή όταν η κορυφή βρίσκεται σε μία από τις δύο άκρες της.

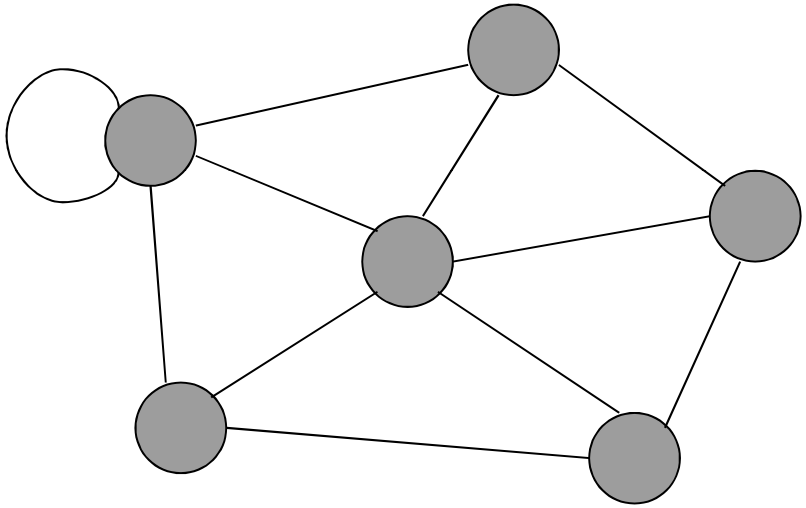
Βαθμός κορυφής (degree): αριθμός ακμών που προσπίπτουν στην κορυφή.



Μη Κατευθυνόμενοι Γράφοι

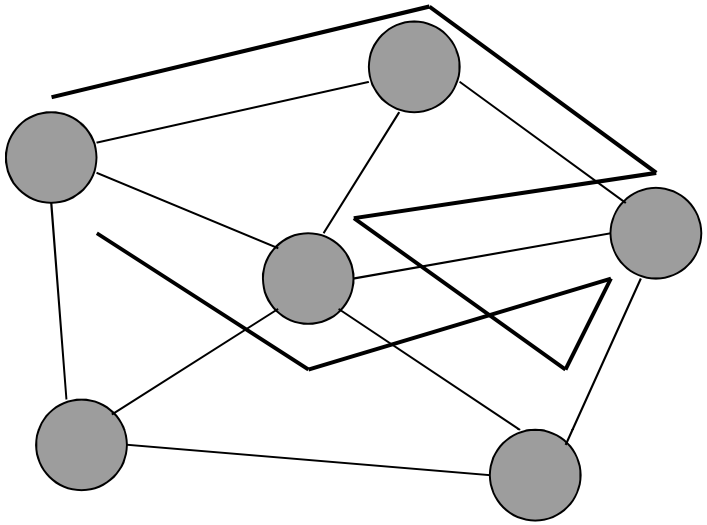
Γειτονικές κορυφές (adjacent): υπάρχει ακμή προσπίπτουσα και στις δύο.

Loop: μία ακμή με τις δύο άκρες της στην ίδια κορυφή.

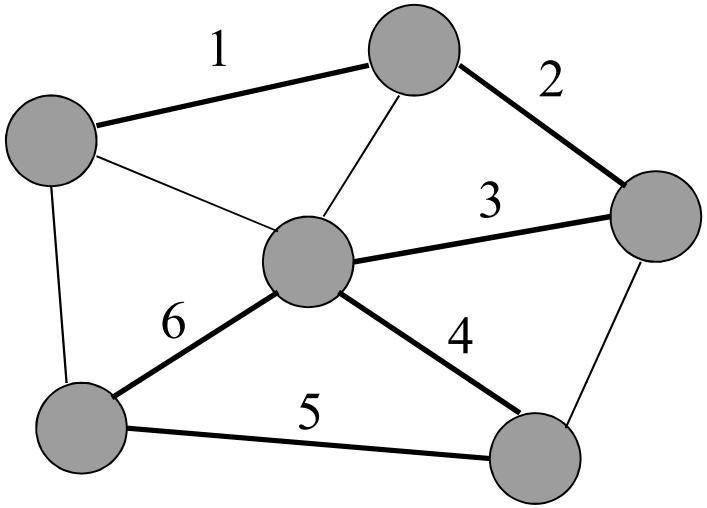


Απλός γράφος: δεν έχει loops και δεν υπάρχει ζεύγος κορυφών με περισσότερες της μίας ακμές μεταξύ τους.

Μη κατευθυνόμενοι Γράφοι

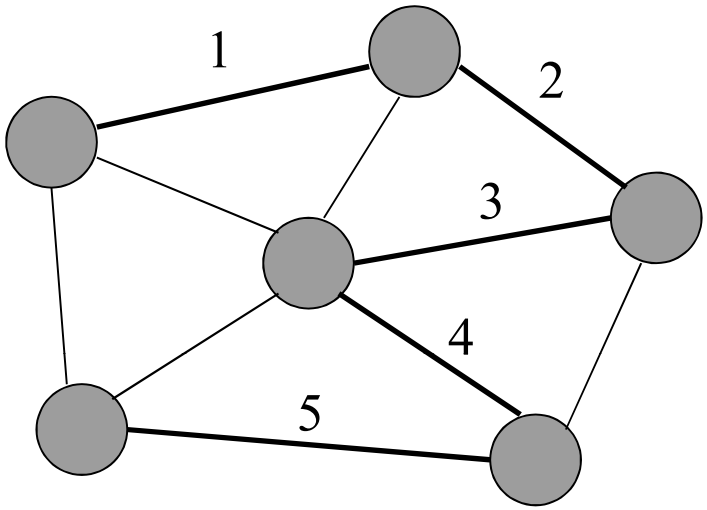


Περίπατος (walk): μία εναλλασσόμενη ακολουθία ακμών και κορυφών.

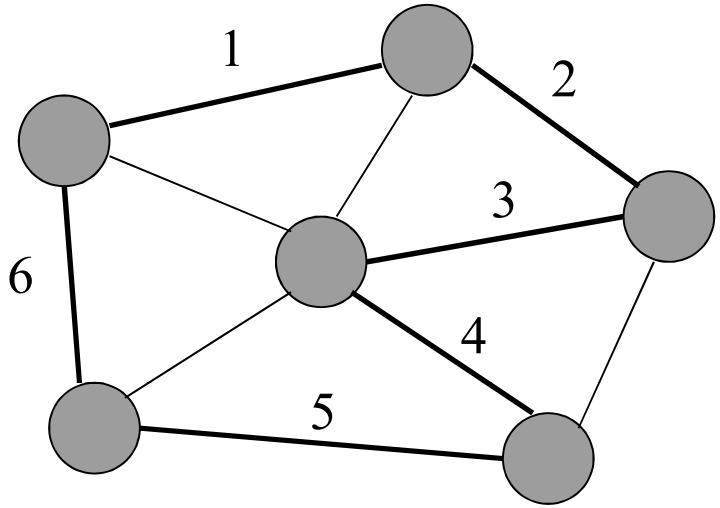


Πέρασμα (trail): είναι ένας περίπατος από ξεχωριστές ακμές.

Μη κατευθυνόμενοι Γράφοι



Μονοπάτι (path): είναι ένα πέρασμα από ξεχωριστές κορυφές.

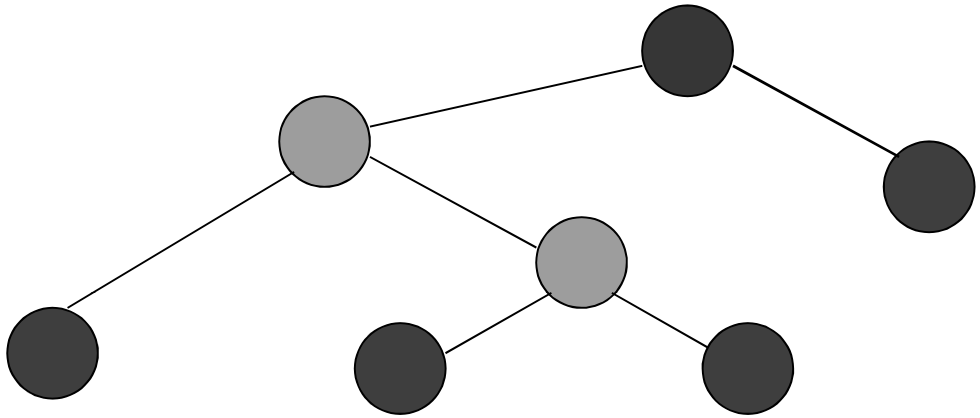


Κύκλος (cycle): ένας περίπατος κλειστός από ξεχωριστές κορυφές.

Μη κατευθυνόμενοι Γράφοι

Διασυνδεδεμένος γράφος: όλα τα ζεύγη κορυφών ενώνονται με ένα μονοπάτι.

Άκυκλος γράφος ή δάσος: ένας γράφος χωρίς κύκλους.



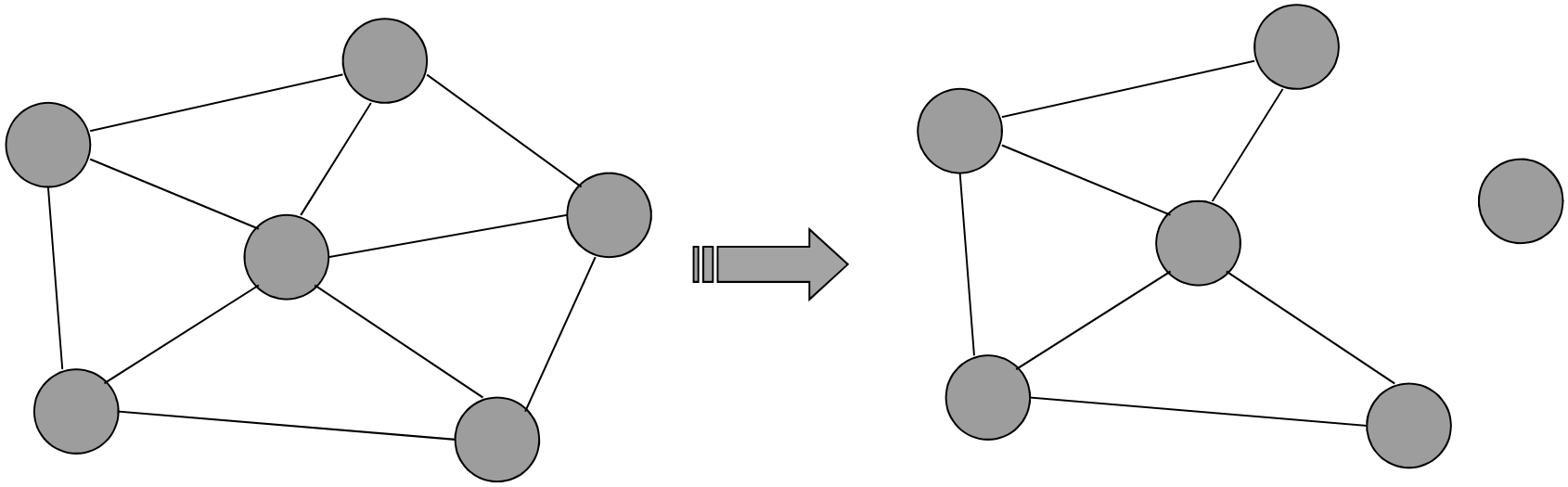
Δέντρο: διασυνδεδεμένος άκυκλος γράφος

Δέντρο με ρίζα: δέντρο με μία χωριστή κορυφή που ονομάζεται ρίζα (root).

Nodes / leaves: οι κορυφές ενός δέντρου (εκτός της ρίζας) / όταν έχουν μία μόνο γειτονική κορυφή

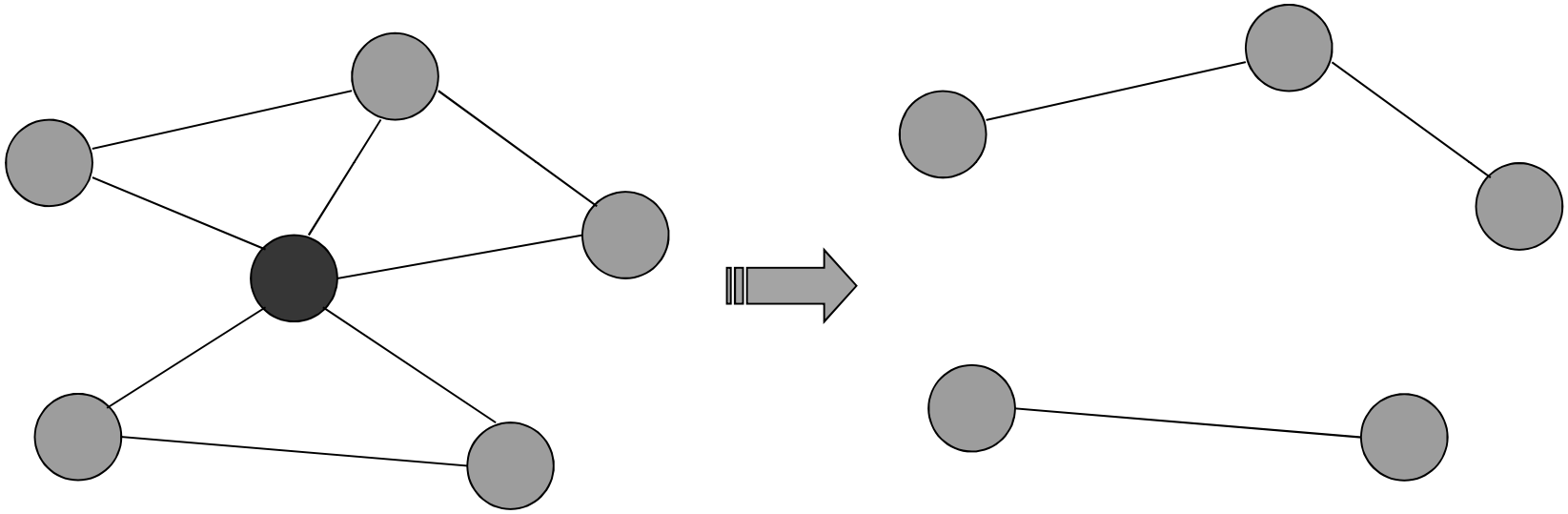
Μη κατευθυνόμενοι Γράφοι

Σύνολο αποκοπής (cutset): ένα ελάχιστο σύνολο ακμών που με την απομάκρυνσή του από τον γράφο τον κάνει μη διασυνδεδεμένο.

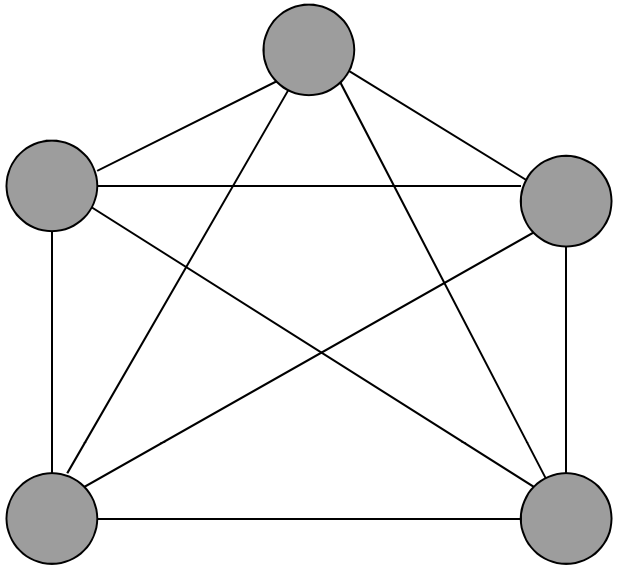


Μη κατευθυνόμενοι Γράφοι

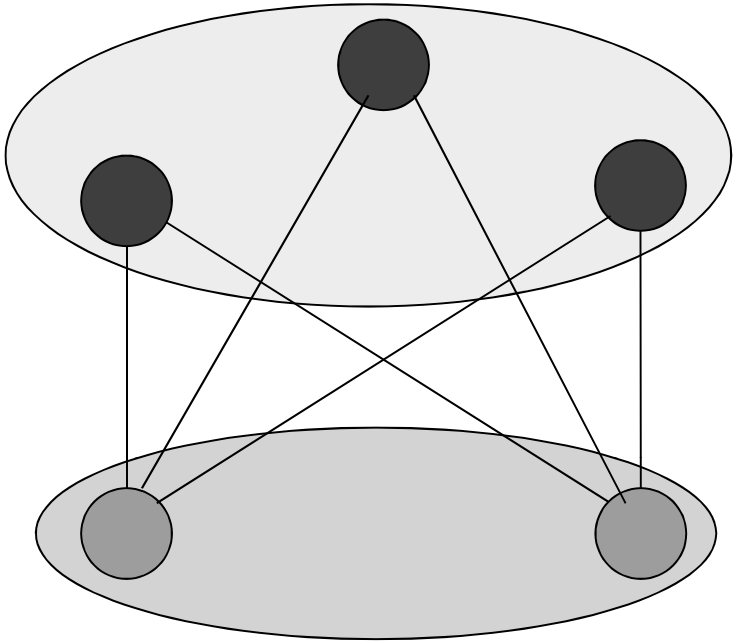
Σύνολο διαχωρισμού κορυφών (vertex separation set): ένα ελάχιστο σύνολο κορυφών που με την απομάκρυνσή του από τον γράφο τον κάνει μη διασυνδεδεμένο.



Μη κατευθυνόμενοι Γράφοι



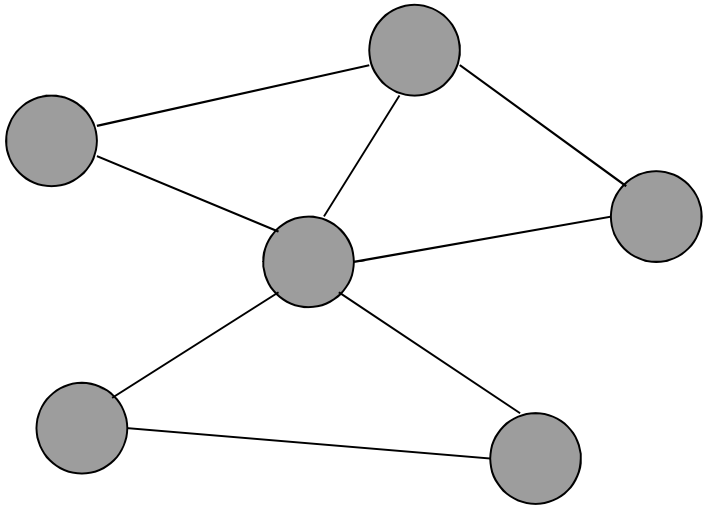
Πλήρης γράφος: κάθε ζεύγος κορυφών συνδέεται με μία ακμή.

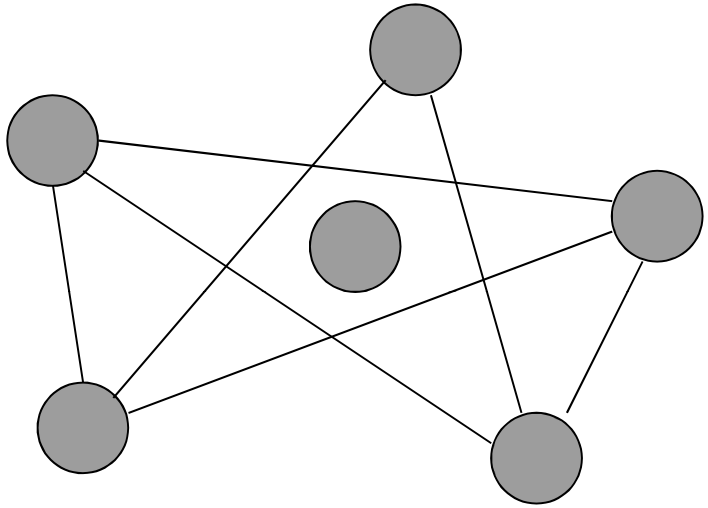


Διμερής γράφος (bipartite): το σύνολο κορυφών διαχωρίζεται σε δύο υποσύνολα έτσι ώστε κάθε ακμή έχει άκρα και στα δύο υποσύνολα.

Μη κατευθυνόμενοι Γράφοι

Συμπλήρωμα γράφου: το σύνολο ακμών είναι συμπληρωματικό.

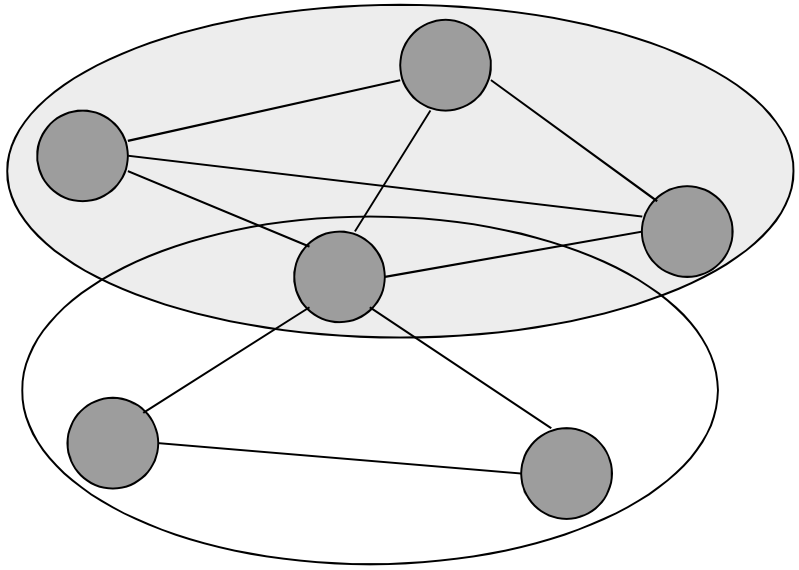




Υπογράφος του G(V, E): ένας γράφος με σύνολα κορυφών και ακμών που περιέχονται στα V, E.

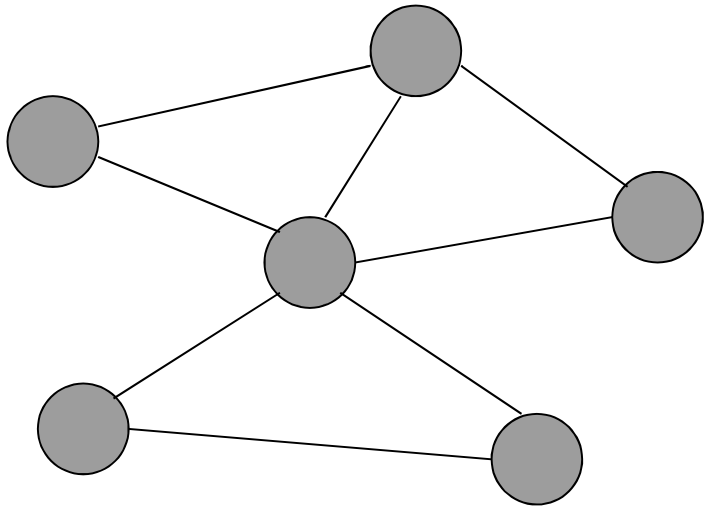
Μη κατευθυνόμενοι Γράφοι

Κλίκα γράφου: ένας πλήρης υπογράφος. Μέγιστη όταν δεν περιέχεται σε άλλη κλίκα.



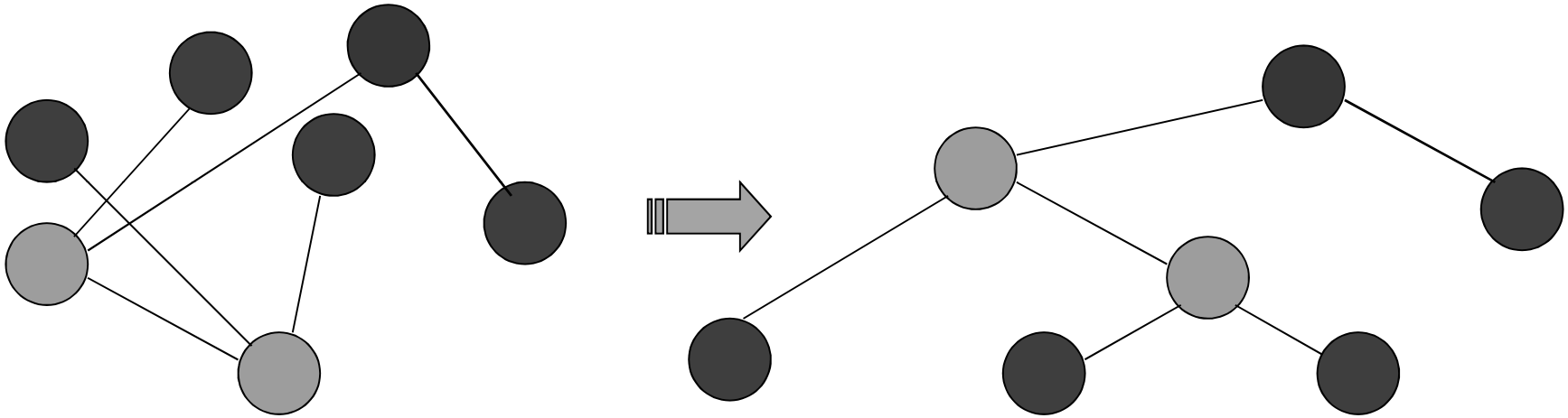
Μη κατευθυνόμενοι Γράφοι

Επίπεδος γράφος (planar): το διάγραμμά του πάνω σε επίπεδο δεν έχει δύο ακμές που να διασταυρώνονται.



Μη κατευθυνόμενοι Γράφοι

Ισομορφικοί γράφοι: υπάρχει αντιστοιχία ένα προς ένα μεταξύ των συνόλων κορυφών που διατηρεί την γειτονικότητα.

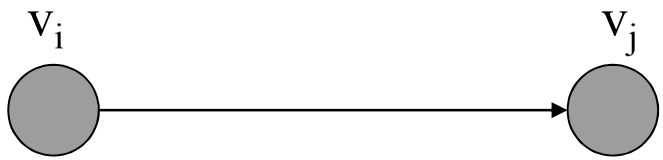


Κατευθυνόμενοι Γράφοι

Κεφαλή (head) / Ουρά (tail): σε μία κατευθυνόμενη ακμή (v_i, v_j) η κορυφή v_j είναι η κεφαλή και η v_i είναι η ουρά.

Βαθμός εισόδου κορυφής (indegree): αριθμός ακμών στις οποίες η κορυφή είναι κεφαλή.

Βαθμός εξόδου κορυφής (outdegree): αριθμός ακμών στις οποίες η κορυφή είναι ουρά.



Ουρά (tail) Κεφαλή (head)

Προκάτοχος (predecessor): Διάδοχος (successor)

Κατευθυνόμενοι Γράφοι

Περίπατος (walk): εναλλασσόμενη ακολουθία κορυφών και ακμών με την ίδια κατεύθυνση.

Πέρασμα (trail): περίπατος από ξεχωριστές ακμές με την ίδια κατεύθυνση.

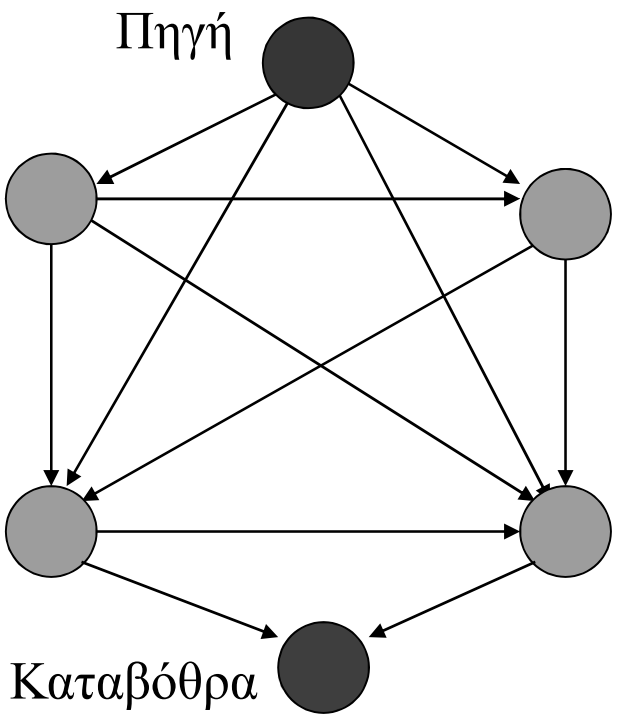
Μονοπάτι (path): είναι ένα πέρασμα από ξεχωριστές κορυφές.

Κύκλος (cycle): ένας περίπατος κλειστός από ξεχωριστές κορυφές

Οι **κατευθυνόμενοι άκυκλοι γράφοι** (Dags) αναπαριστούν μερικώς διατεταγμένα σύνολα.

Κατευθυνόμενοι Γράφοι

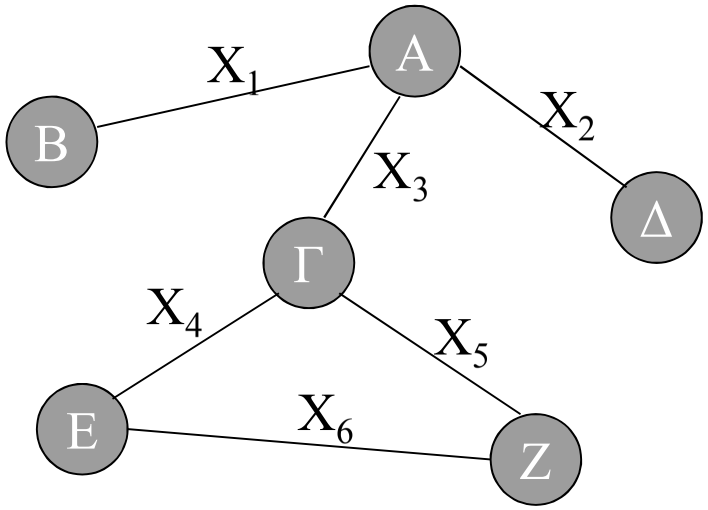
Πολωμένος άκυκλος γράφος: έχει δύο ξεχωριστές κορυφές, την πηγή και την καταβόθρα. Κάθε κορυφή είναι προσεγγίσιμη από την πηγή και προσεγγίζει την καταβόθρα.



Κατευθυνόμενοι Γράφοι

Incidence Matrix: μήτρα αναπαράστασης γράφου.

- Κάθε γραμμή αντιστοιχεί σε κορυφή και κάθε στήλη αντιστοιχεί σε ακμή.
- Η θέση (i, j)=1 εάν η ακμή j προσπίπτει στην κορυφή v_i.
- Για κατευθυνόμενους γράφους παίρνει τιμές 1 για κεφαλή και –1 για ουρά.



	X_1	X_2	X_3	X_4	X_5	X_6
A	1	1	1	0	0	0
B	1	0	0	0	0	0
Γ	0	0	1	1	1	0
Δ	0	1	0	0	0	0
E	0	0	0	1	0	1
Z	0	0	0	0	1	1

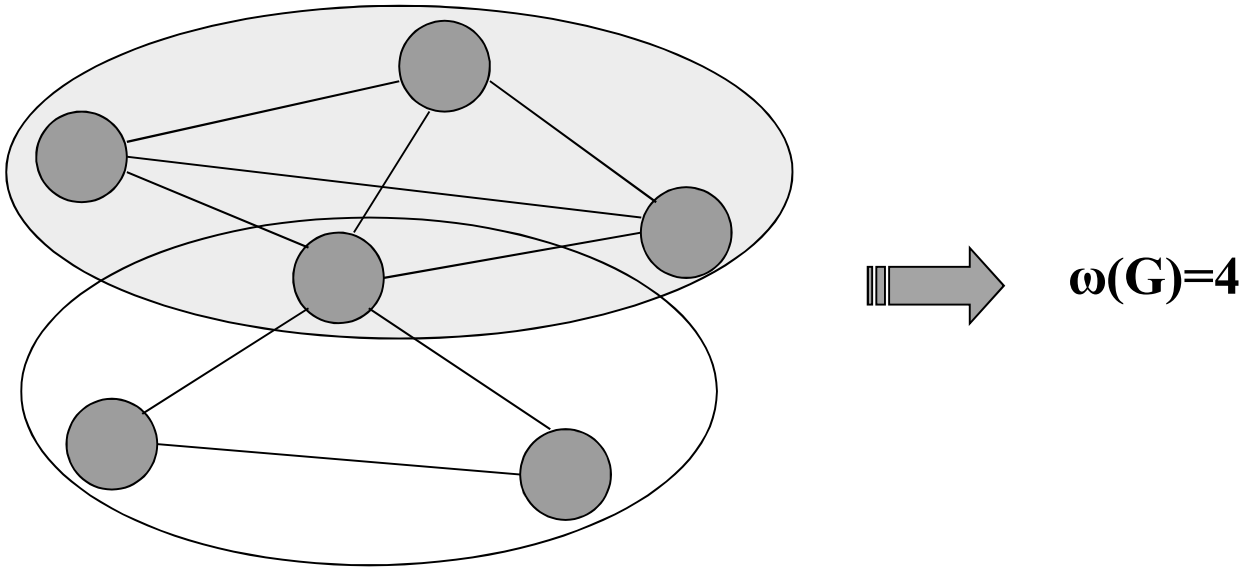
Κατευθυνόμενοι Γράφοι

Μήτρα γειτονικότητας (adjacency): γραμμές και στήλες αντιστοιχούν σε κορυφές. Η θέση (i, j)=1 εάν η κορυφή v_i γειτονεύει με την v_j.

Γράφοι με βάρη: κάθε ακμή ή/και κορυφή έχει βάρος.

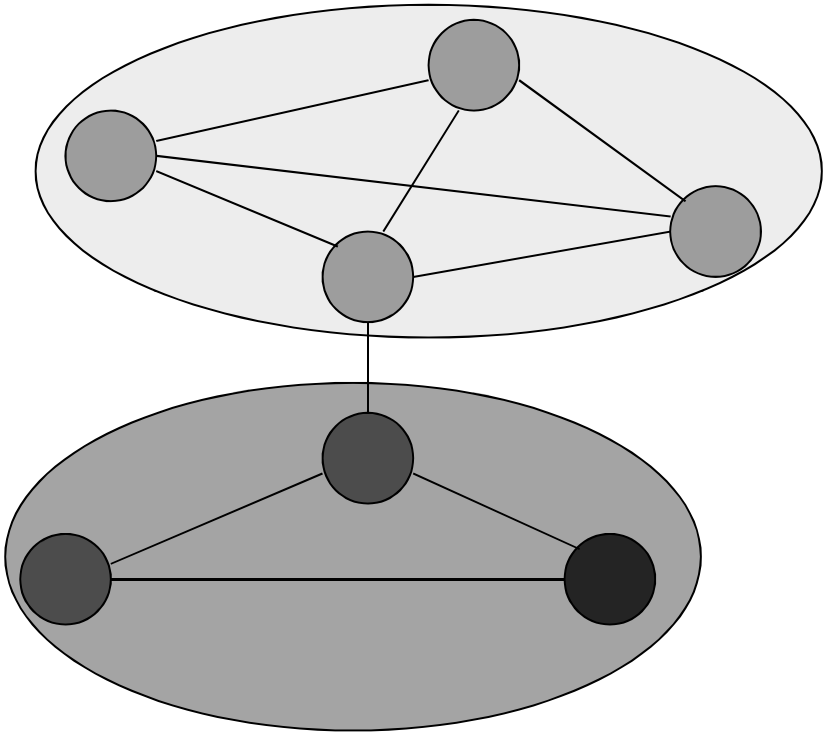
Μέγιστη κλίκα: η μεγαλύτερη κλίκα ενός γράφου.

Νούμερο κλίκας (ω(G)): το μέγεθος (αριθμός κορυφών) της μέγιστης κλίκας.



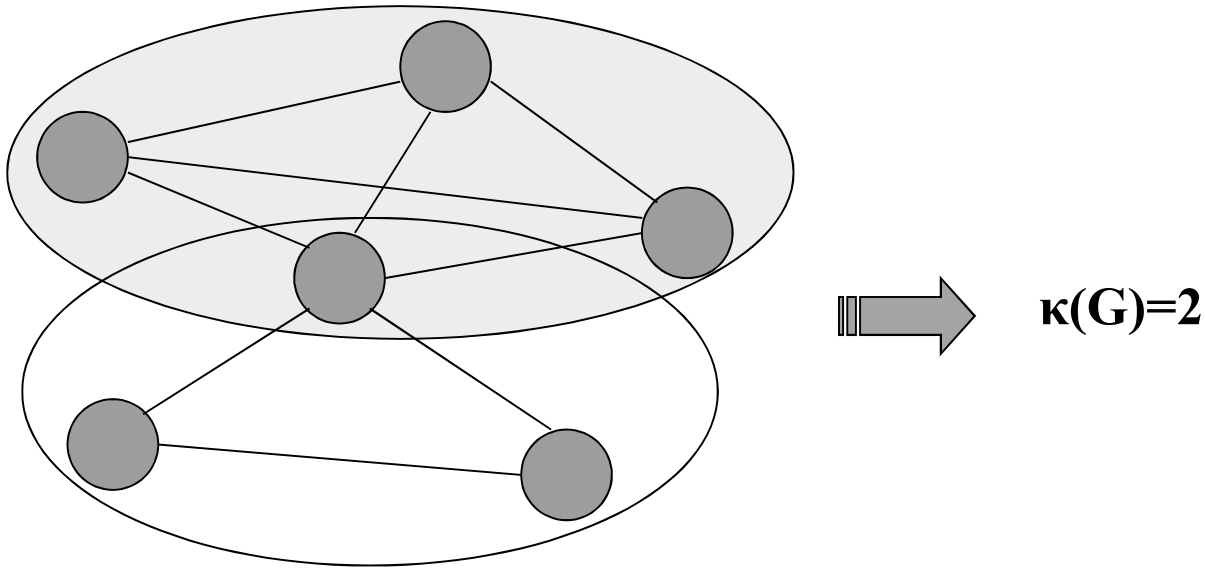
Κατευθυνόμενοι Γράφοι

Ένας γράφος **διαμερίζεται σε κλίκες** εάν το σύνολο κορυφών του διαμερίζεται σε διαφορετικά σύνολα κάθε ένα από τα οποία υπονοεί κλίκα.



Κατευθυνόμενοι Γράφοι

Ένας γράφος **καλύπτεται από κλίκες** εάν το σύνολο κορυφών του διαιρείται σε σύνολα (πιθανώς επικαλυπτόμενα) κάθε ένα από τα οποία υπονοεί κλίκα.



Αριθμός κάλυψης κλίκας: ο ελάχιστος αριθμός από κλίκες που καλύπτουν τον γράφο κ(G).

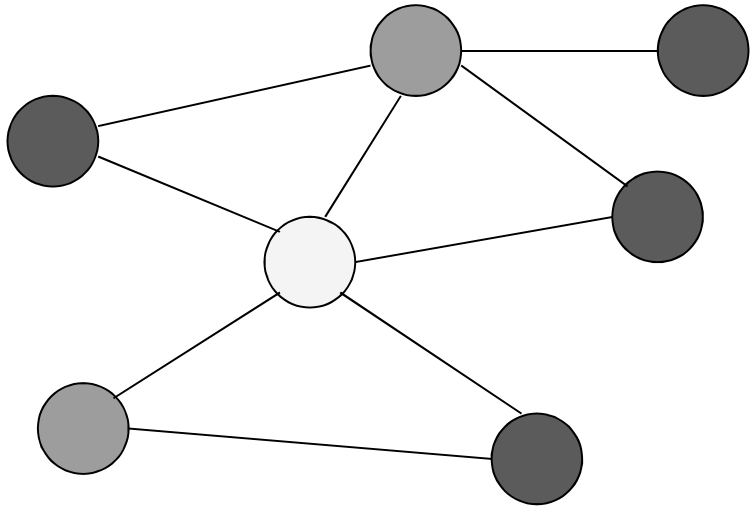
Κατευθυνόμενοι Γράφοι

Σταθερό σύνολο ή σύνολο ανεξαρτησίας: ένα υποσύνολο των κορυφών οπού δεν υπάρχουν δύο κορυφές γειτονικές.

Αριθμός σταθερότητας: α(G) είναι το μέγεθος του μεγαλύτερου σταθερού συνόλου του.

Χρωματισμός γράφου: η διαμέριση του γράφου σε σταθερά σύνολα.

Χρωματικός αριθμός χ(G) : ο μικρότερος αριθμός που μπορεί να είναι το μέγεθος μίας χρωματικής διαμέρισης (αριθμός χρωμάτων).



α(G)=4

χ(G)=3

Εισαγωγή

- ✓ Υπάρχει μεγάλη διαφορά μοντέλου περιγραφής και τελικού κυκλώματος.
- ✓ Χρειάζεται μία κανονική ενδιάμεση αναπαράσταση HDL – Hardware για να γίνεται αντιστοίχιση της περιγραφής σε διαφορετικές αρχιτεκτονικές.

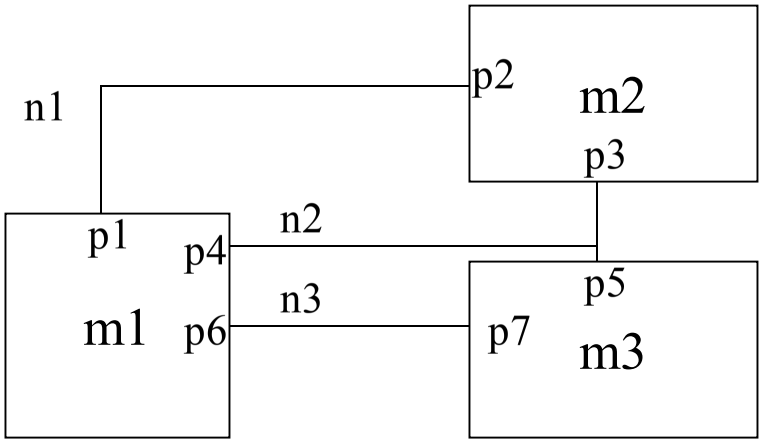
Η κανονική αναπαράσταση:

- ➢ διατηρεί την περιγραφή
- ➢ επιτρέπει την σταδιακή προσθήκη αποτελεσμάτων σύνθεσης (refinement).
- ➢ είναι ανεξάρτητη από τις HDLs.
- ➢ είναι αρκετά ισχυρή για να παρέχει διάφορα στυλ αρχιτεκτονικής.

Αφηρημένα μοντέλα

Δομές (Structures)

Σύνολο modules και nets και μία σχέση μεταξύ τους. Η σχέση αναπαρίσταται με μήτρα (incidence matrix).



m_1: n_1, n_2, n_3

m_2: n_1, n_2

m_3: n_2, n_3

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

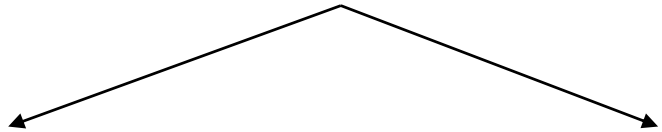
Συχνά οι μήτρες είναι αραιές οπότε χρησιμοποιούμε netlists.

Ιεραρχικότητα: ένα module αποτελείται από submodules – άλλη μήτρα.

Αφηρημένα μοντέλα

Λογικά Δίκτυα

Γενικευμένο λογικό δίκτυο: δομή με κάθε φύλλο =μία συνδυαστική ή ακολουθιακή (σύγχρονη) λογική συνάρτηση.



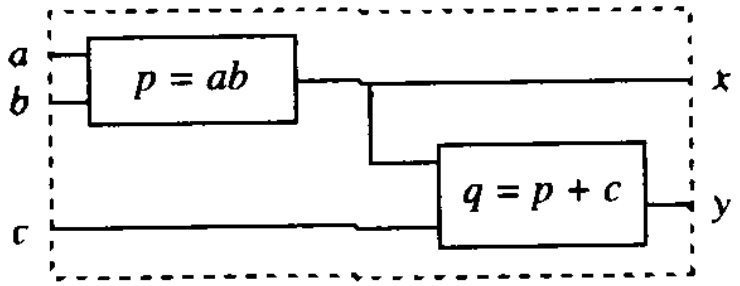
Συνδυαστικά Λογικά Δίκτυα Σύγχρονα Λογικά Δίκτυα

Αναπαράσταση με κατευθυνόμενους γράφους

Αφηρημένα μοντέλα

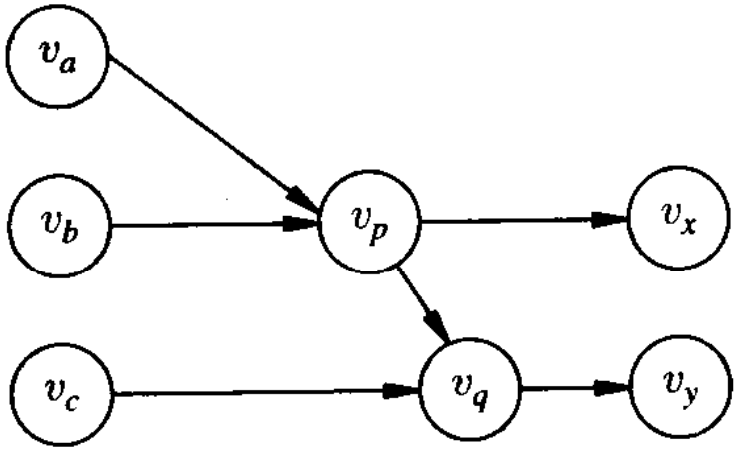
Συνδυαστικό Λογικό Δίκτυο

- Κάθε φύλλο σχετίζεται με μία πολλαπλής εισόδου/μίας εξόδου συνδυαστική λογική συνάρτηση (τοπική συνάρτηση – local function).
- Τα pins διαιρούνται σε εισόδους/εξόδους, κύριες εισόδους/εξόδους
- Κάθε γραμμή έχει μία αρχή και μία κατεύθυνση σε τερματικά σημεία.
- Η σχέση γραμμών και modules είναι μερικής διάταξης (δεν υπάρχουν κύκλοι σε αντιστοιχία με τα συνδυαστικά κυκλώματα).



Αφηρημένα μοντέλα

Αναπαράσταση με **λογικό γράφο δικτύου**: κατευθυνόμενος γράφος με το σύνολο κορυφών του σε 1-προς-1 αντιστοιχία με κύριες εισόδους/εξόδους και τοπικές συναρτήσεις.



Ένα λογικό δίκτυο είναι υβριδική περιγραφή συμπεριφοράς (incidence structure) – δομής (local functions).

Στην περίπτωση που οι τοπικές συναρτήσεις αντιπροσωπεύουν πύλες, πρόκειται για αναπαράσταση κατασκευής και ονομάζεται **bound or mapped network**.

Αφηρημένα μοντέλα

- ✓ Τα λογικά δίκτυα αναπαριστούν λογικές συναρτήσεις με δομημένο τρόπο.
- ✓ Εάν συνδυάσουμε τις τοπικές συναρτήσεις σύμφωνα με την δομή του γράφου, μπορούμε να υπολογίσουμε τις λογικές συναρτήσεις των εξόδων.
- ✓ Οι λογικές συναρτήσεις των εξόδων συχνά δεν είναι δυνατό να υπολογιστούν λόγω μεγέθους.
- ✓ Το λογικό δίκτυο δεν έχει τέτοιο πρόβλημα.

Αφηρημένα μοντέλα

Σύγχρονα λογικά δίκτυα: γενίκευση των συνδυαστικών λογικών δικτύων, για αναπαράσταση σύγχρονων ακολουθιακών κυκλωμάτων.

⇩

Τα φύλλα αναπαριστούν **τοπικές συναρτήσεις** και **στοιχεία καθυστέρησης.**

Η μερική διάταξη **δεν ισχύει** στο σύνολο των ακμών του γράφου καθώς τώρα υπάρχουν κύκλοι**.**

⇩

Η μερική διάταξη **ισχύει** στο σύνολο των ακμών του γράφου που δεν ξεκινούν από στοιχεία καθυστέρησης (δεν υπάρχουν κύκλοι που να μην συμπεριλαμβάνουν στοιχεία καθυστέρησης)**.**

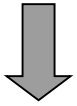
Διαγράμματα Καταστάσεων

Μηχανές πεπερασμένων καταστάσεων (ακολουθιακά κυκλώματα):

- Σύνολο διανυσμάτων εισόδου X, εξόδου Y
- Σύνολο καταστάσεων S και συνάρτηση μετάβασης καταστάσεων δ:X x S→S
- Συνάρτηση εξόδου λ: X x S→Y (Mealy), λ: S→Y (Moore)
- Αρχική κατάσταση

Τρόποι αναπαράστασης:

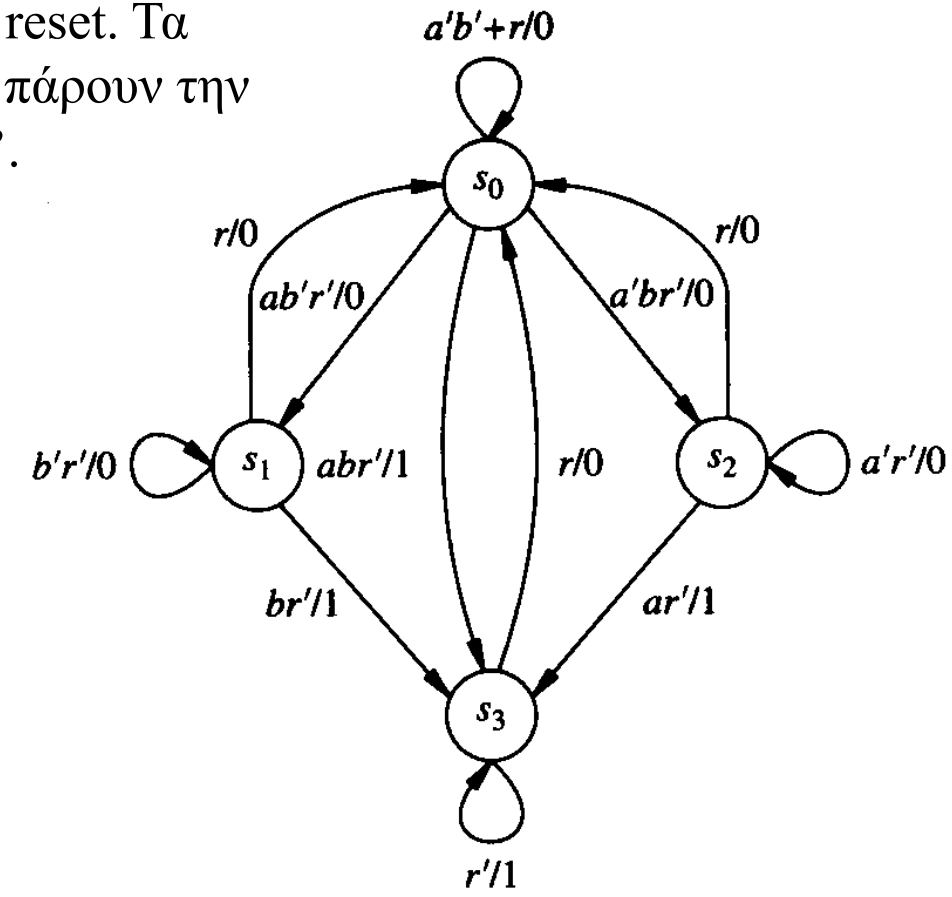
Πίνακας Καταστάσεων – Διάγραμμα Μετάβασης Καταστάσεων



κατευθυνόμενος γράφος με κορυφές τις καταστάσεις και ακμές τις μεταβάσεις από κατάσταση σε κατάσταση

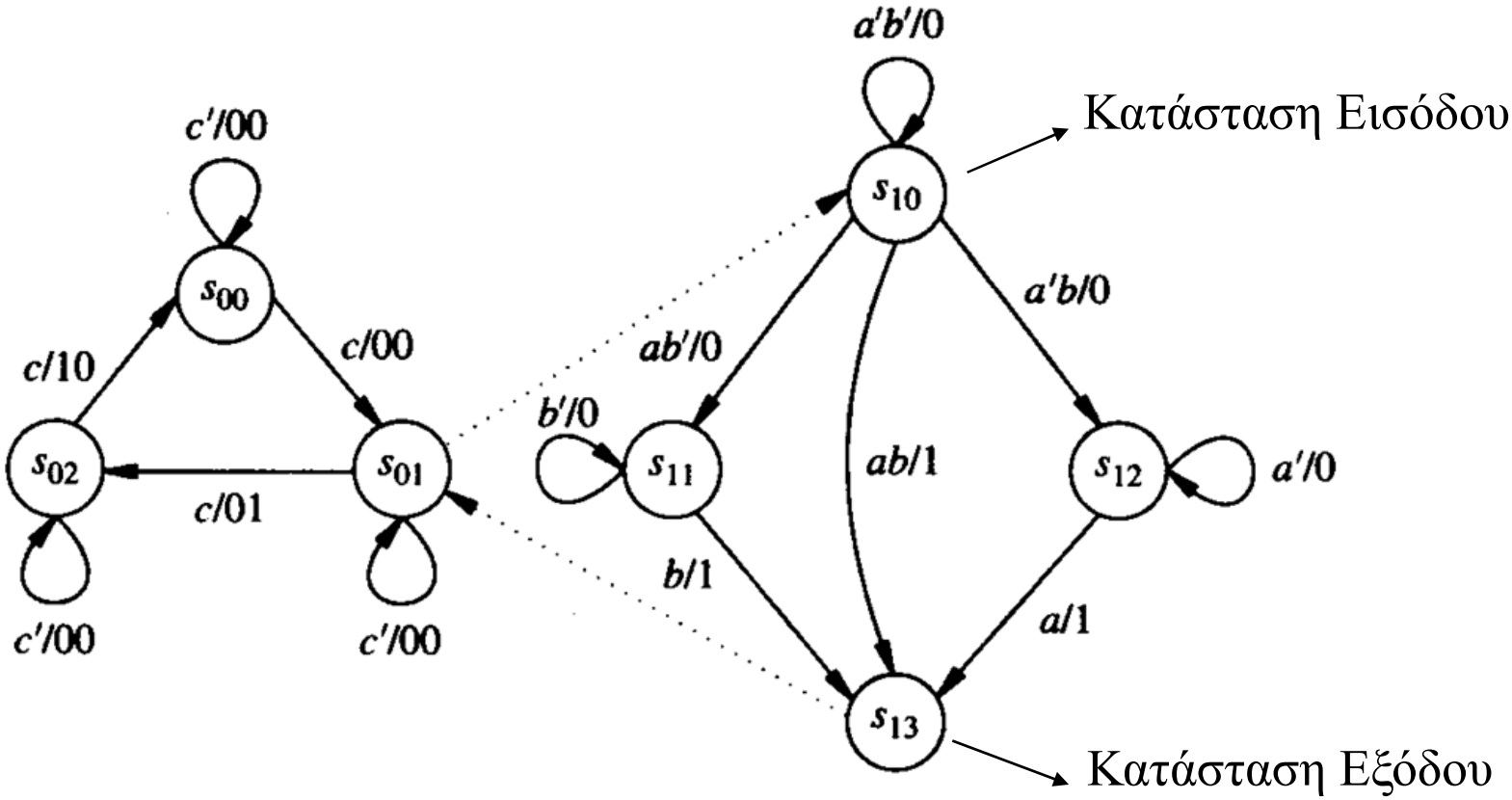
Διαγράμματα Καταστάσεων

Συγχρονιστής δύο σημάτων (a,b) με είσοδο reset. Τα σήματα πρέπει να πάρουν την τιμή ‘1’.



Διαγράμματα Καταστάσεων

Ιεραρχικό Διάγραμμα καταστάσεων: υπάρχουν κορυφές που αντιστοιχούν σε υποδιαγράμματα.



Γράφοι ροής δεδομένων & ακολουθιών

Τα αφηρημένα μοντέλα συμπεριφοράς στο επίπεδο αρχιτεκτονικής περιγράφονται με **λειτουργίες** (operations) και **εξαρτήσεις** (dependencies).

Οι εξαρτήσεις οφείλονται στους ακόλουθους παράγοντες:

- **Διαθεσιμότητα** δεδομένων (Αποτελέσματα λειτουργίας=Δεδομένα επόμενης).
- **Σειριακότητα** εκτέλεσης λειτουργιών από τις προδιαγραφές πχ. εκτέλεση λειτουργίας και κατόπιν ενημέρωση σημαίας.
- **Διαμοίραση** κοινών resources (εκτέλεση σε διαφορετικούς χρόνους)

Γράφοι ροής δεδομένων

Ένας ***γράφος ροής δεδομένων*** αναπαριστά λειτουργίες και εξαρτήσεις:

- ✓ Είναι κατευθυνόμενος γράφος.
- ✓ Οι κορυφές αντιστοιχούν μία προς μία με τις λειτουργίες.
- ✓ Οι λειτουργίες χρειάζονται δεδομένα και δίνουν αποτελέσματα.
- ✓ Οι κατευθυνόμενες ακμές αναπαριστούν την μεταφορά πληροφοριών από λειτουργία σε λειτουργία
- ✓ Υπονοείται η ύπαρξη μεταβλητών που αποθηκεύουν τέτοιες πληροφορίες.
- ✓ Για τις μεταβλητές υπάρχει χρόνος ζωής.
- ✓ Γέννηση είναι η **πρώτη στιγμή** που η μεταβλητή είναι έξοδος μίας λειτουργίας.
- ✓ Θάνατος είναι η **τελευταία στιγμή** που η μεταβλητή είναι είσοδος μίας λειτουργίας

Γράφοι ροής δεδομένων

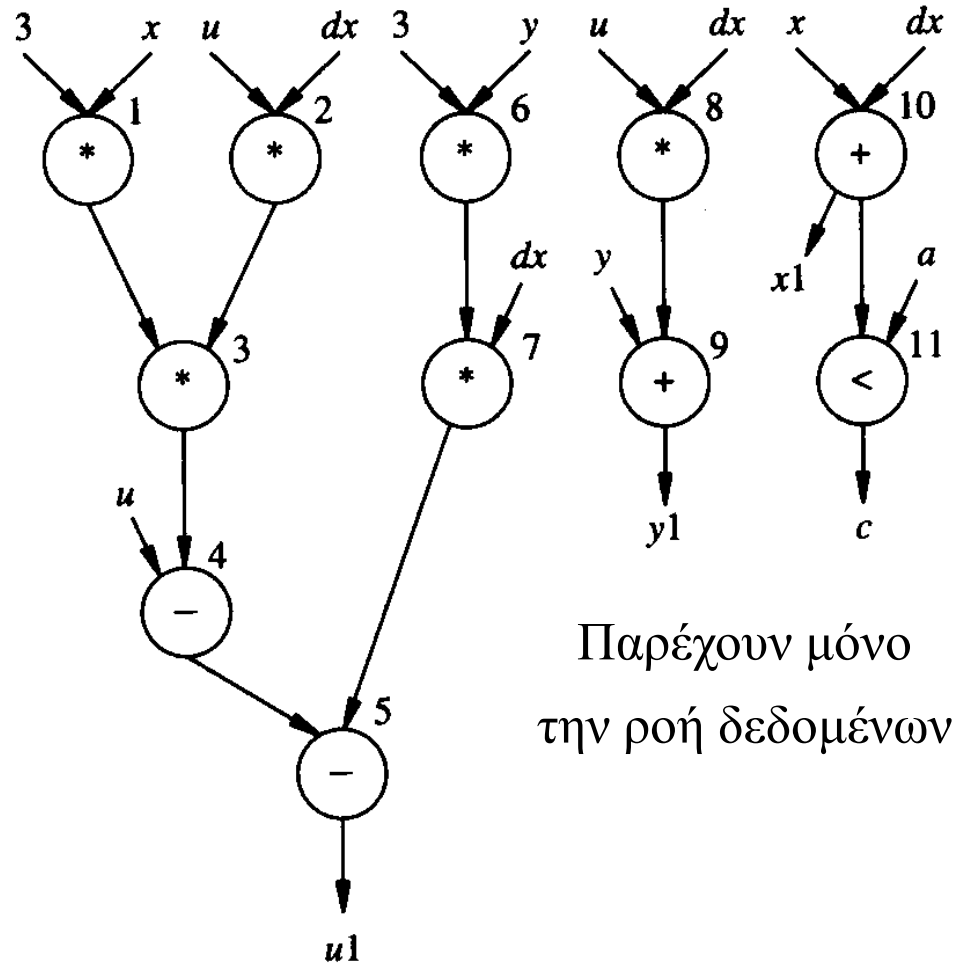
Data Flow Graph

xl = x+dx;

ul = u-(3*x*u*dx)-(3*y*dx);

yl = y+u*dx;

c = xl < a;



Υπάρχουν πολλοί διαφορετικοί γράφοι (αντιμεταθετική /προσεταιριστική ιδιότητα)

Παρέχουν μόνο την ροή δεδομένων

Γράφοι ροής ελέγχου-δεδομένων

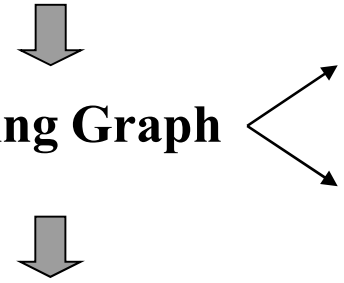
Control-Data Flow Graph

Ροή Ελέγχου: Και οι δομές που αναπαριστούν έλεγχο ροής δεδομένων μπορούν να αναπαρασταθούν γραφικά:

- Οι δομές ελέγχου (ροής) είναι η διακλάδωση και η επανάληψη.
- Επεκτείνουμε τους γράφους ροής δεδομένων με την εισαγωγή κορυφών διακλάδωσης και επανάληψης.
- **Κορυφή διακλάδωσης**: η ουρά ενός συνόλου εναλλακτικών μονοπατιών, και επιλέγει εκτιμώντας κάποια συνθήκη.
- **Κορυφή επανάληψης**: η ουρά δύο μονοπατιών, το ένα είναι το μονοπάτι επανάληψης και το άλλο είναι το μονοπάτι εξόδου.

Γράφοι ελέγχου ροής δεδομένων

Ένα είδος **Control-Data Flow Graph**



Sequencing Graph

Η ροή δεδομένων αναπαρίσταται με γράφους

Η ροή ελέγχου αναπαρίσταται με ιεραρχία.

Sequencing Graphs: Είναι ιεραρχίες κατευθυνόμενων γράφων με τα ακόλουθα χαρακτηριστικά:

- Έχουν δύο είδη κορυφών: λειτουργίες και συνδέσμους.
- Ένας σύνδεσμος διασυνδέει τον γράφο με έναν άλλο γράφο (ιεραρχικά χαμηλότερο).
- Οι λειτουργίες και εξαρτήσεις αναπαρίστανται με κορυφές και κατευθυνόμενες ακμές.

Sequencing Graphs χωρίς ιεραρχία

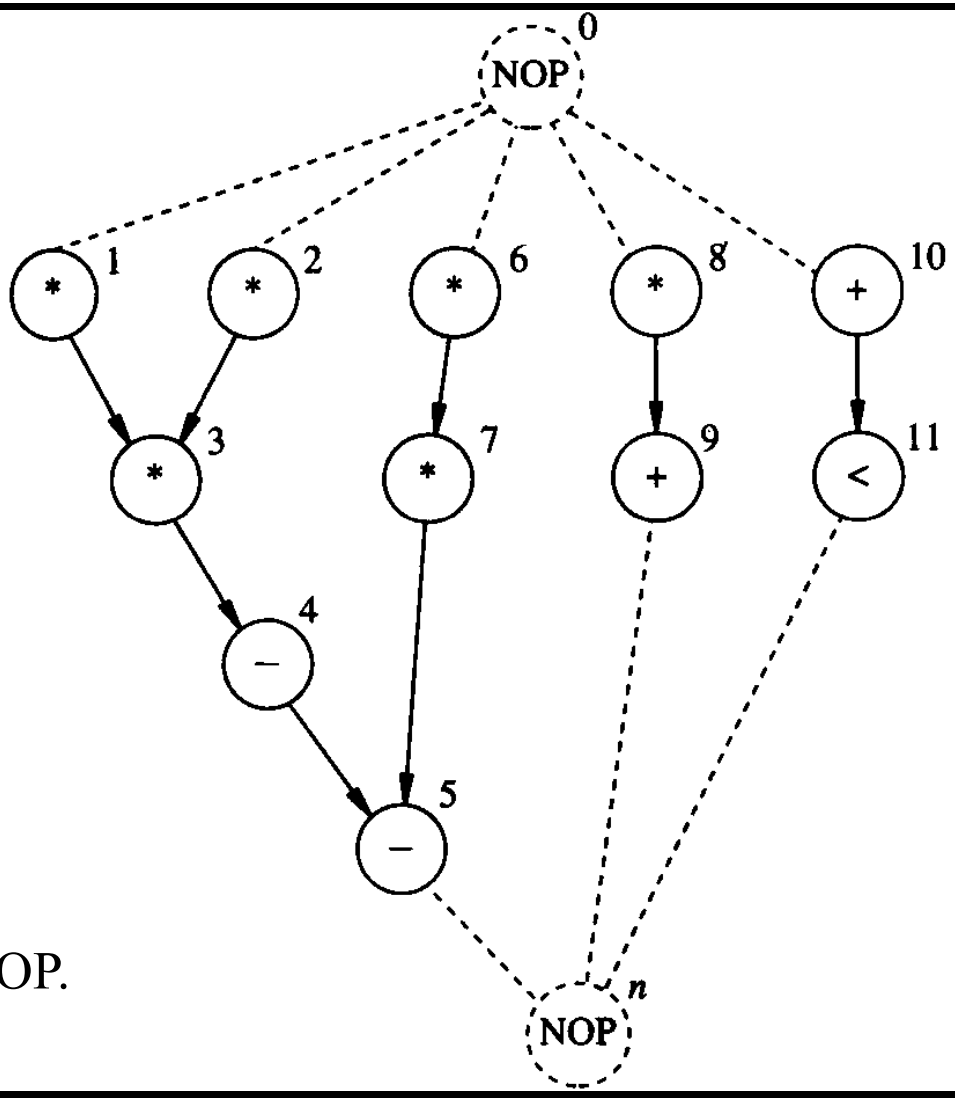
A. Sequencing Graphs χωρίς ιεραρχικές κορυφές:

- Είναι **άκυκλοι γράφοι** και **πολωμένοι** (δύο ειδικές κορυφές είναι η πηγή και η καταβόθρα / πρώτη και τελευταία λειτουργία του γράφου).
- Κάθε ακμή $v_i \rightarrow v_j$ υπονοεί ότι η πράξη v_j έπεται της πράξης v_i.
- Κάθε μονοπάτι στον γράφο αναπαριστά παράλληλες πράξεις.

Sequencing Graphs χωρίς ιεραρχία

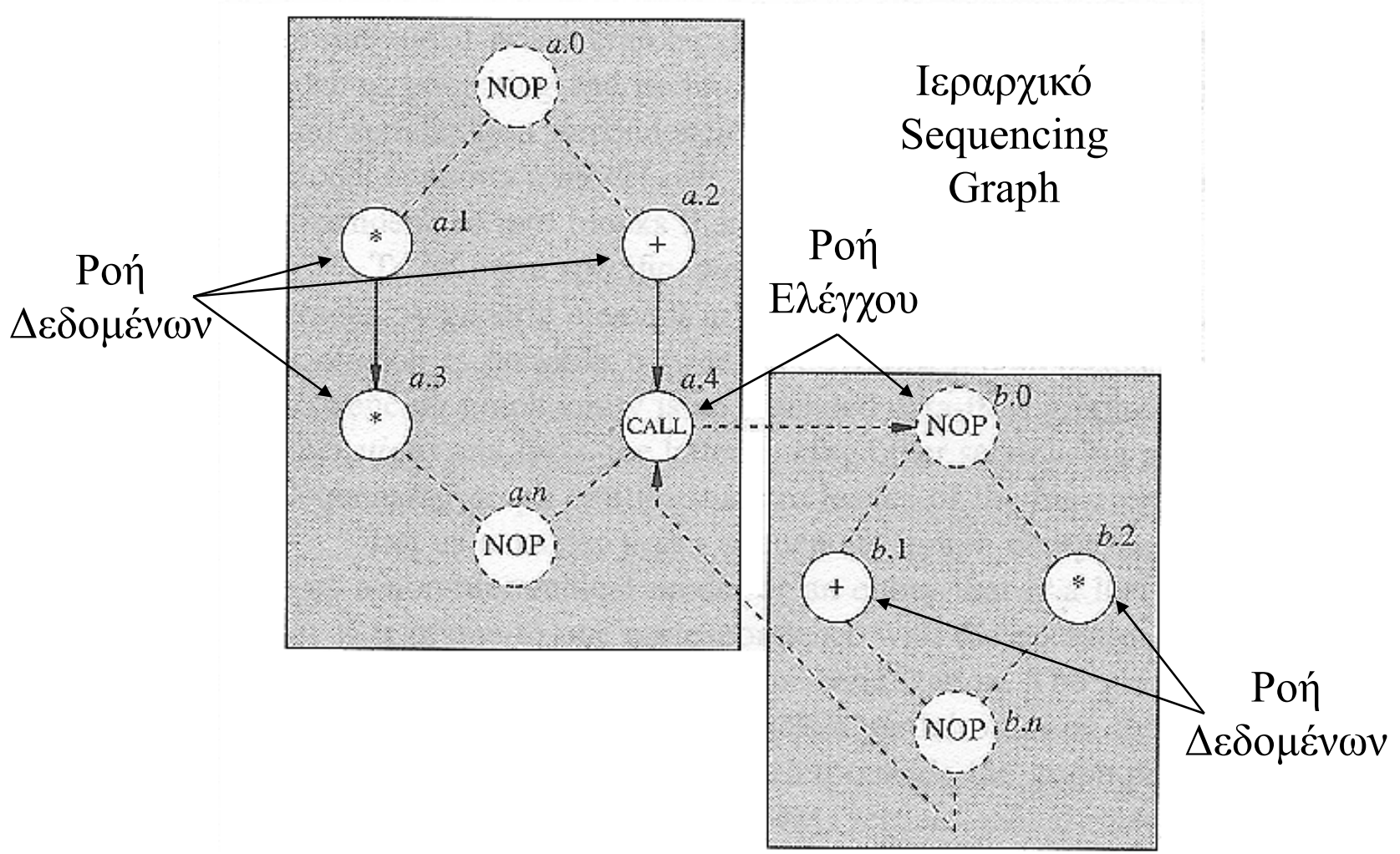
Sequencing Graph

1. Άκυκλος (μερική διάταξη)
2. Πολωμένος



Η πηγή και η καταβόθρα είναι NOP.

Sequencing Graphs με ιεραρχία

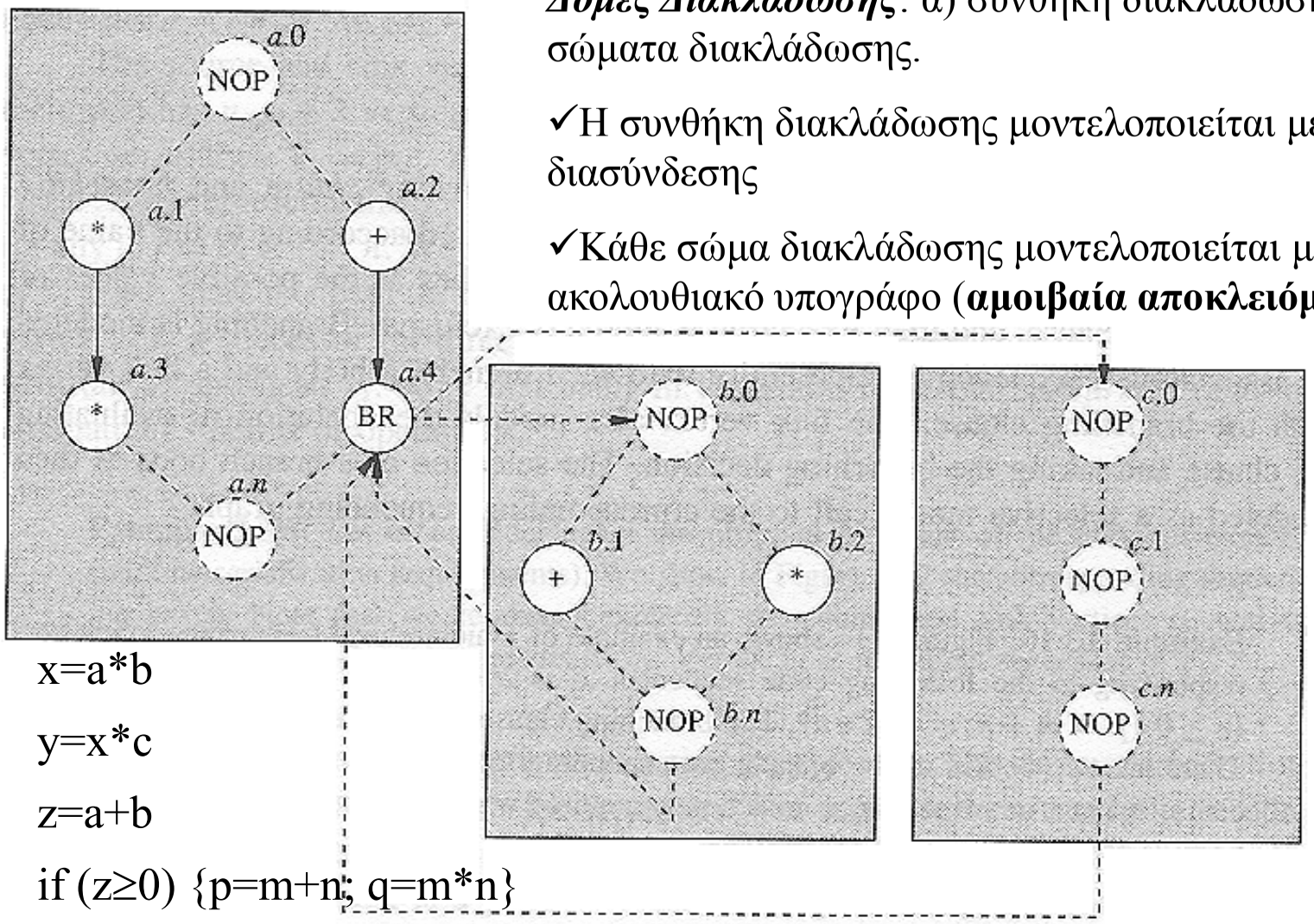


Sequencing Graphs

Δομές Διακλάδωσης: α) συνθήκη διακλάδωσης και β) σώματα διακλάδωσης.

✓Η συνθήκη διακλάδωσης μοντελοποιείται με μία κορυφή διασύνδεσης

✓Κάθε σώμα διακλάδωσης μοντελοποιείται με έναν ακολουθιακό υπογράφο (**αμοιβαία αποκλειόμενοι**).



x=a*b

y=x*c

z=a+b

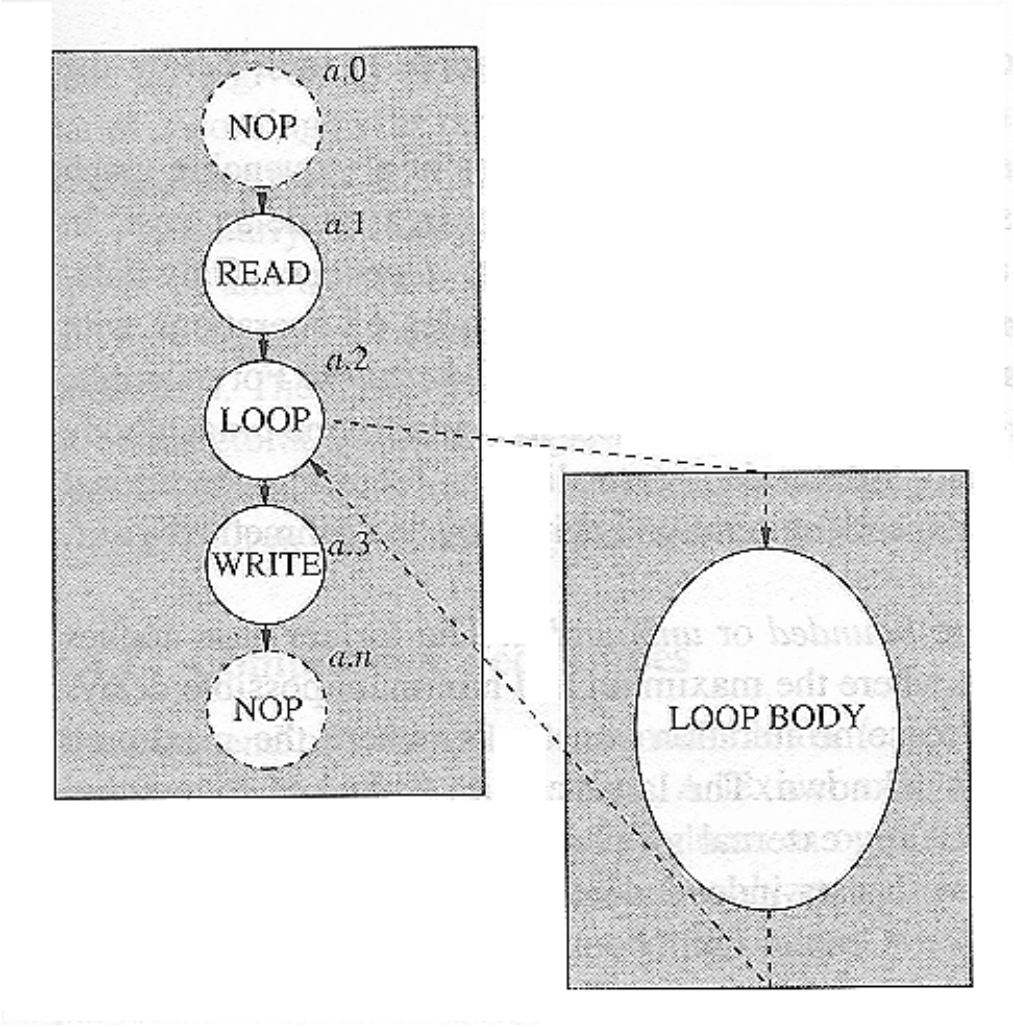
if (z≥0) {p=m+n; q=m*n}

Sequencing Graphs

Δομές Επανάληψης:

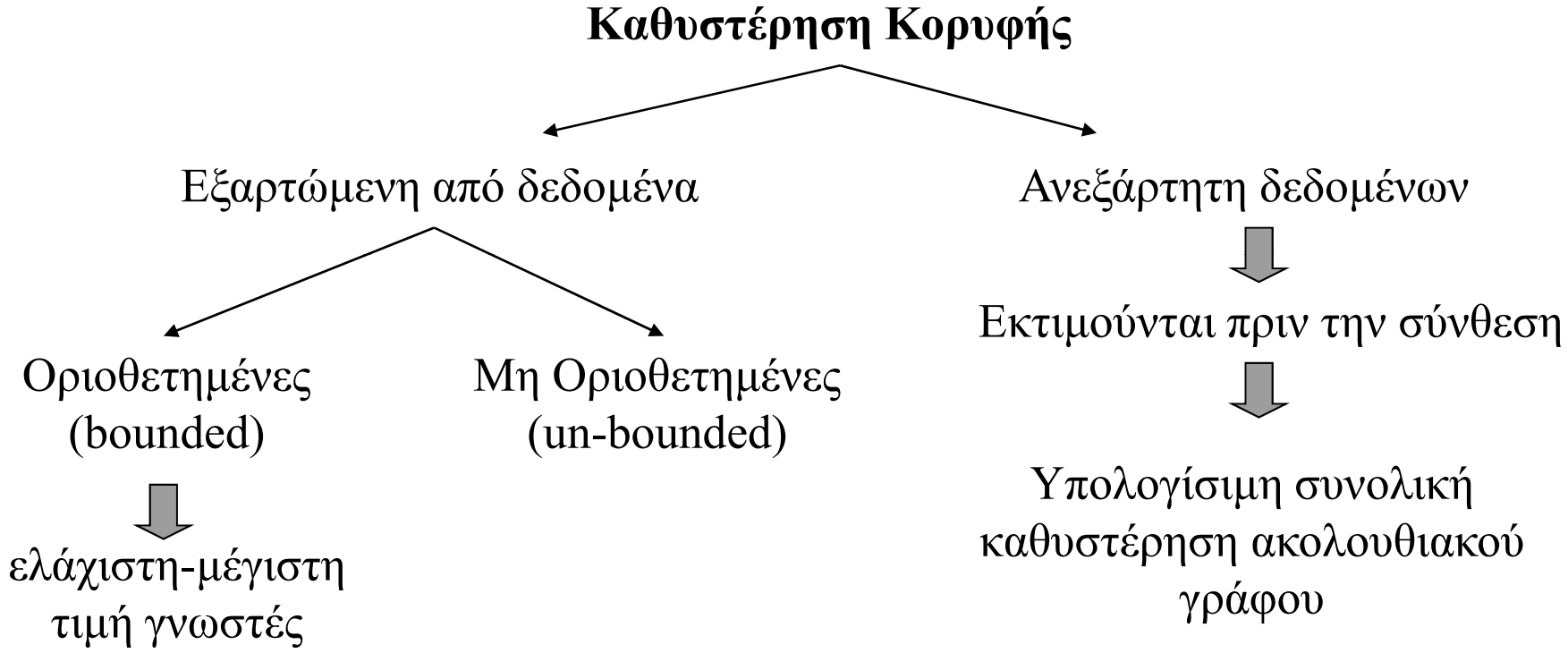
α) συνθήκη επανάληψης (κορυφή διασύνδεσης)

β) σώμα επανάληψης (ακολουθιακός υπογράφος)



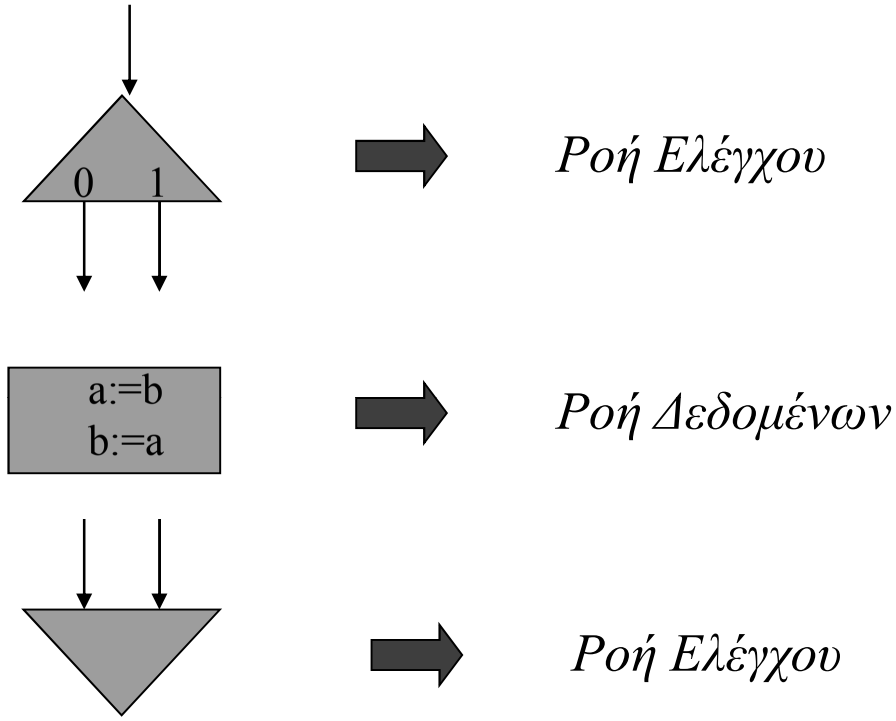
Sequencing Graphs

Σε ακμές και κορυφές τοποθετούνται ιδιότητες (εκτιμήσεις επιφάνειας και καθυστέρησης).

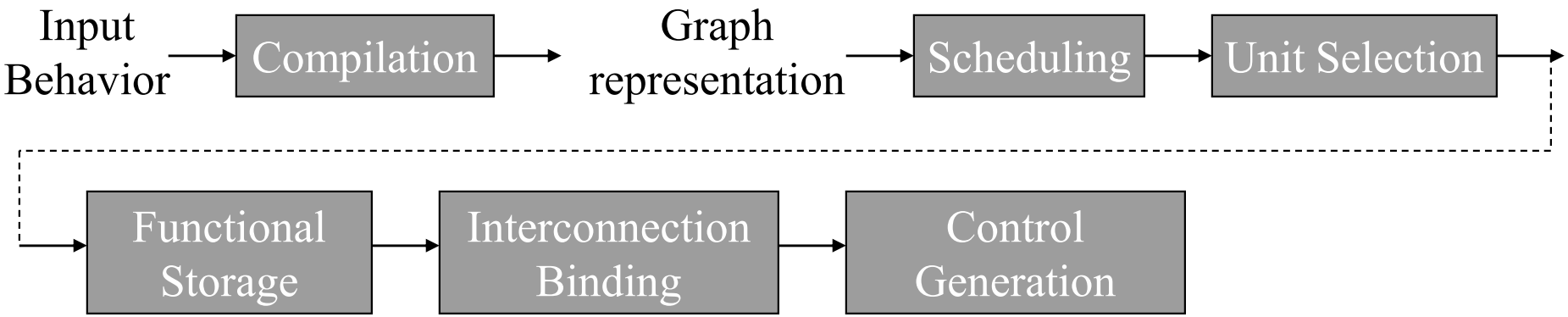


CDFG χωρίς ιεραρχία

Ροή Ελέγχου & Δεδομένων στον ίδιο γράφο



Design Flow



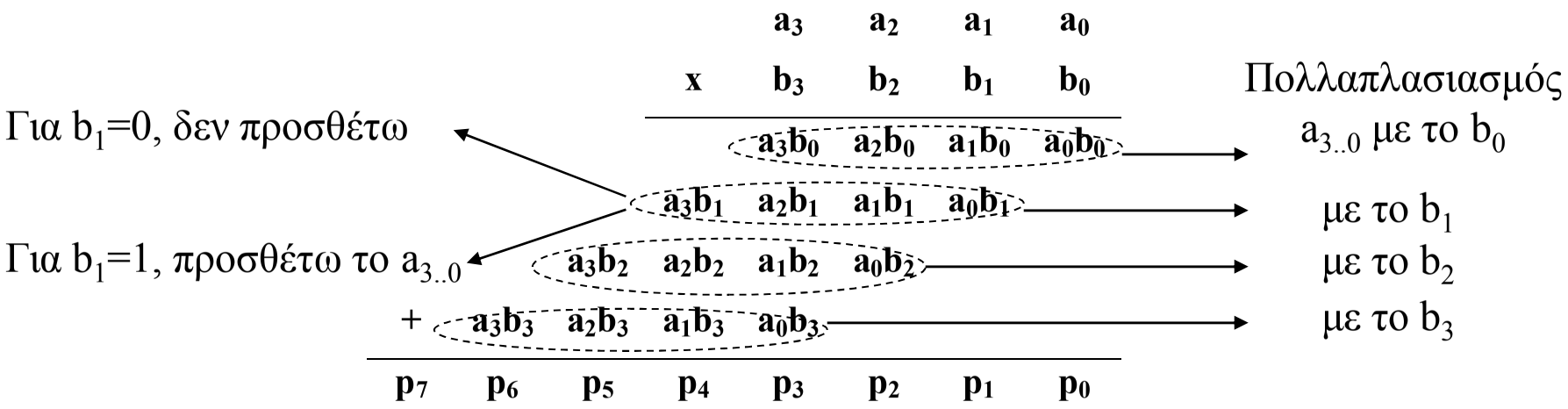
Παράδειγμα:
Πολλαπλασιαστής Ολίσθησης

Αλγόριθμος Πολλαπλασιασμού

Πολλαπλασιασμός 4x4

- Χρειαζόμαστε δύο καταχωρητές μήκους 4 bits για τα a, b, έναν καταχωρητή μήκους 8 bits για το αποτέλεσμα και έναν αθροιστή.
- Οι καταχωρητές πρέπει να έχουν την δυνατότητα ολίσθησης.
- Ολισθαίνουμε δεξιά κάθε φορά το προηγούμενο άθροισμα μερικών γινομένων, και το προσθέτουμε στο νέο μερικό γινόμενο.

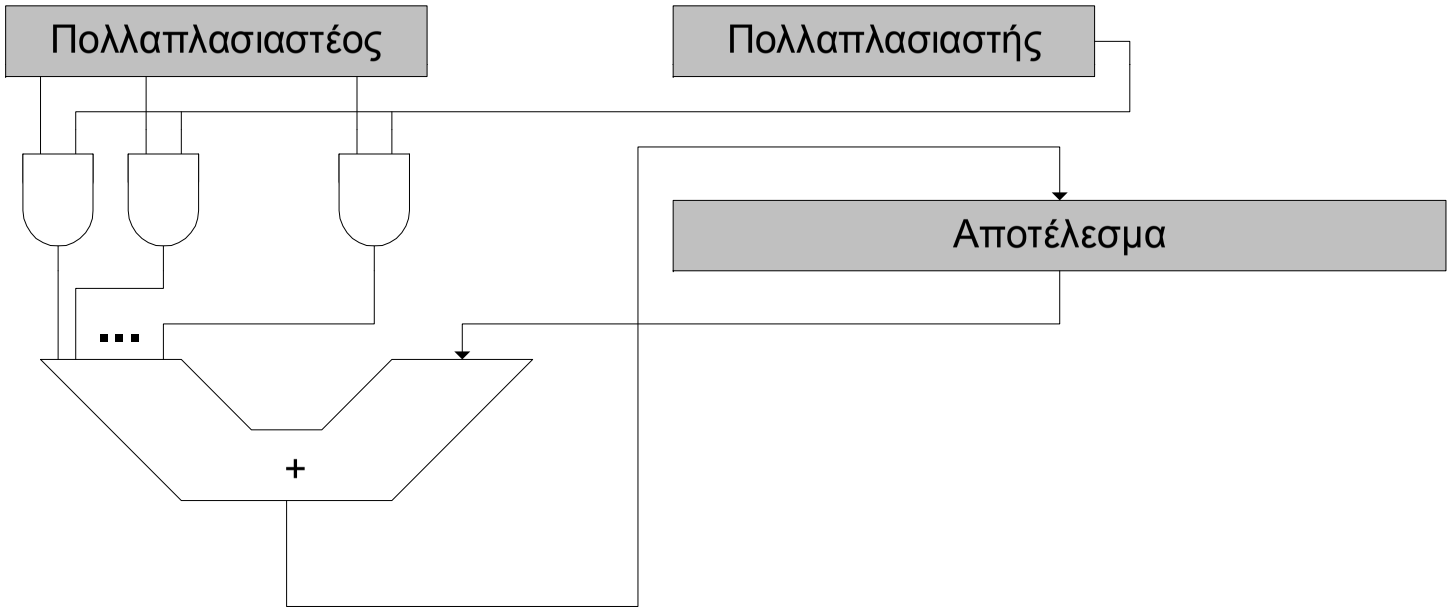
Αλγόριθμος Πολλαπλασιασμού



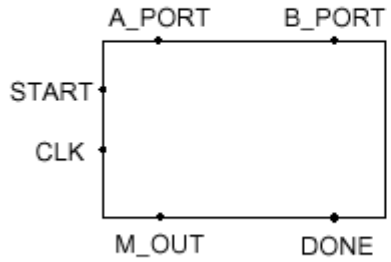
p7	p6	p5	p4	p3	p2	p1	p0	Κύκλος
30	20	10	00					1
	30	20	10	00				
31	30+21	20+11	10+01	00				2
c_1	31	30+21	20+11	10+01	00			
c_1+32	31+22	30+21+12	20+11+02	01+10	00			3
c_2	c_1+32	31+22	30+21+12	20+11+02	01+10	00		
c_2+33	c_1+32+23	31+22+13	30+21+12+03	20+11+02	01+10	00		4
c_3	c_2+33	c_1+32+23	31+22+13	30+21+12+03	20+11+02	01+10	00	
Πρόσθεση								
Ολίσθηση								

Αλγόριθμος Πολλαπλασιασμού

- Προϋποθέτει την δυνατότητα ολίσθησης του πολλαπλασιαστή και του αποτελέσματος στους αντίστοιχους καταχωρητές.
- Έχει μικρό απαιτούμενο κόστος σε υλικό.
- Απαιτεί n κύκλους ρολογιού για να δώσει το αποτέλεσμα



Design Flow




```
entity MULT is
 port ( A_PORT,
        B_PORT:  in bit_vector(3 downto 0);
        M_OUT:  out bit_vector(7 downto 0);
        CLK:  in CLOCK;
        START:  in BIT;
        DONE:  out BIT;
      );
end MULT;
```

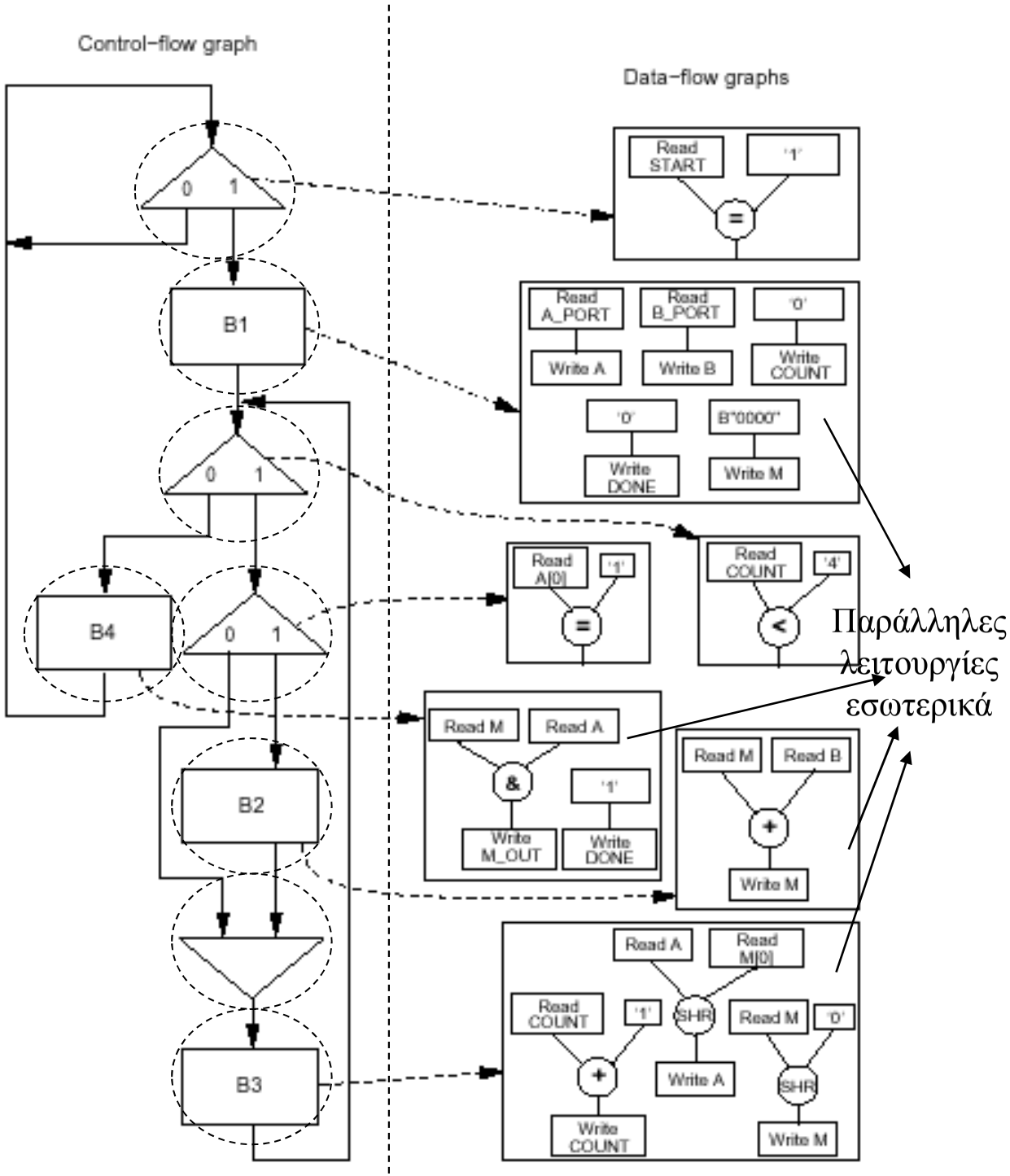
```
architecture SHIFT_MULT  of  MULT  is
begin
     process
          variable A, B, M  :  BIT_VECTOR;
          variable COUNT  :  INTEGER;
     begin
          wait until (START = 1);
          A := A_PORT;   COUNT := 0;
          B := B_PORT;   DONE <= '0';
          M := B"0000";
          while  (COUNT < 4)  loop
               if (A(0) = '1')  then
                         M  :=  M + B;
               end if;
               A:=SHR(A,M(0));
               M:=SHR(M,cin);
               COUNT := COUNT + 1;
          end loop;
          M_OUT  <=  M & A;
          DONE <= '1';
     end process;
end SHIFT_MULT;
```

Οι εντολές εκτελούνται ακολουθιακά εκτός εάν δεν έχουν εξαρτήσεις μεταξύ τους

1st step: Control Data Flow Graph

Control Flow

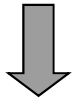
Data Flow



Control – Data Flow Graph

Ούτε η VHDL ούτε ο γράφος δείχνουν με ποιον τρόπο θα υλοποιηθεί σε hardware η περιγραφή:

- ✓ Μεταβλητές όπως η A, B, M δεν έχουν ανατεθεί σε αποθηκευτικά στοιχεία.
- ✓ Λειτουργίες όπως η πρόσθεση δεν έχουν ανατεθεί σε λειτουργικές μονάδες.
- ✓ Δεν έχουν καθορίσει ακολουθία καταστάσεων.
- ✓ Δεν έχουν καθορίσει σήματα ελέγχου για την ενεργοποίηση των λειτουργικών μονάδων του Datapath.



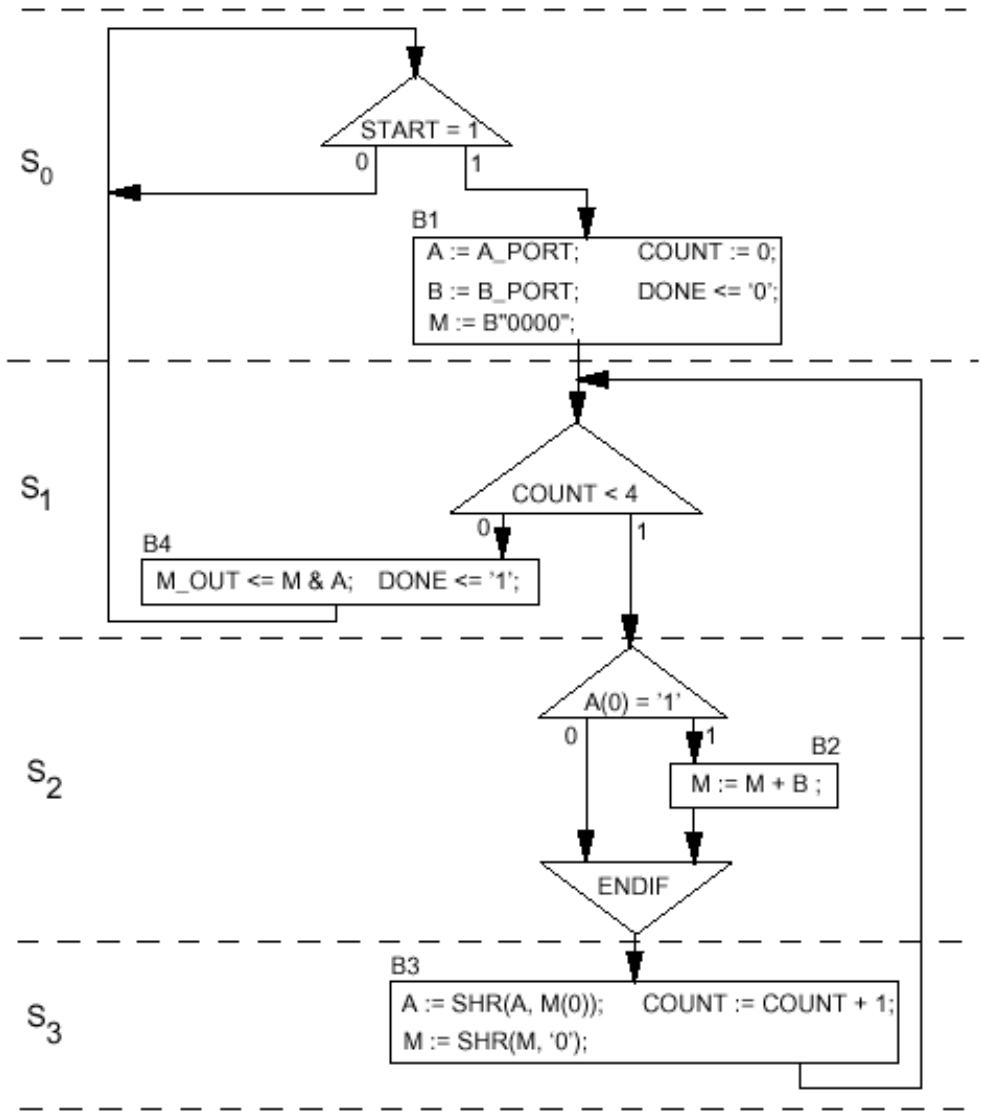
Θα το υλοποιήσουμε με το μοντέλο FSMD

2nd step = Scheduling

Ανάθεση λειτουργιών σε σύγχρονα βήματα

4 βήματα

Καταστάσεις του FSM

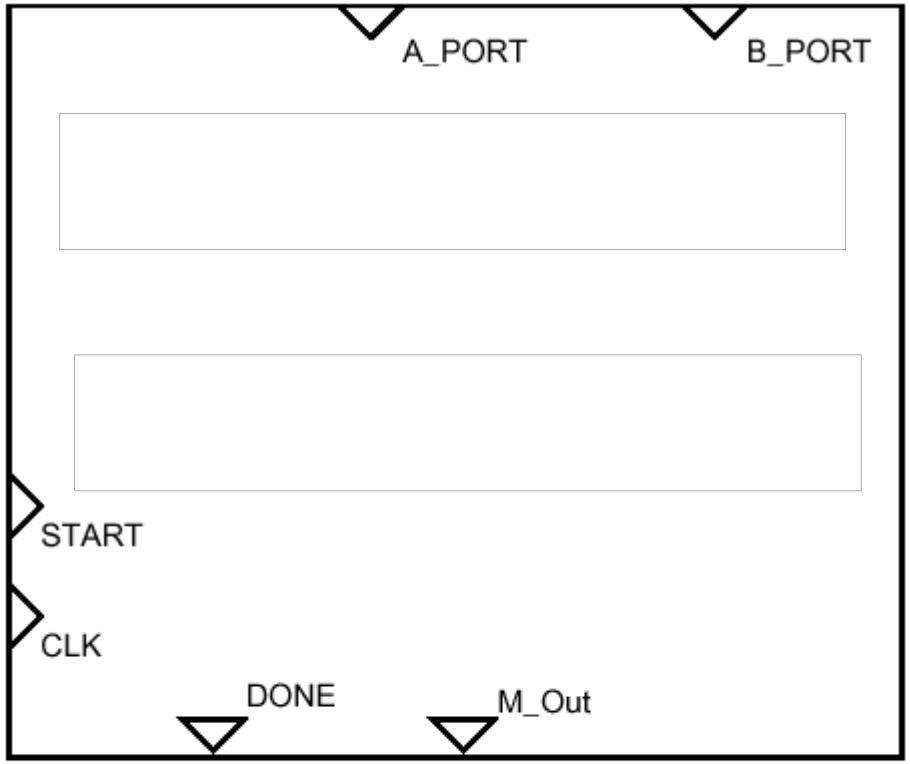


3rd step (a) = Unit Selection

1. Δέσμευση αποθηκευτικών μονάδων για μεταβλητές **A**, **B**, **M** και **Count**

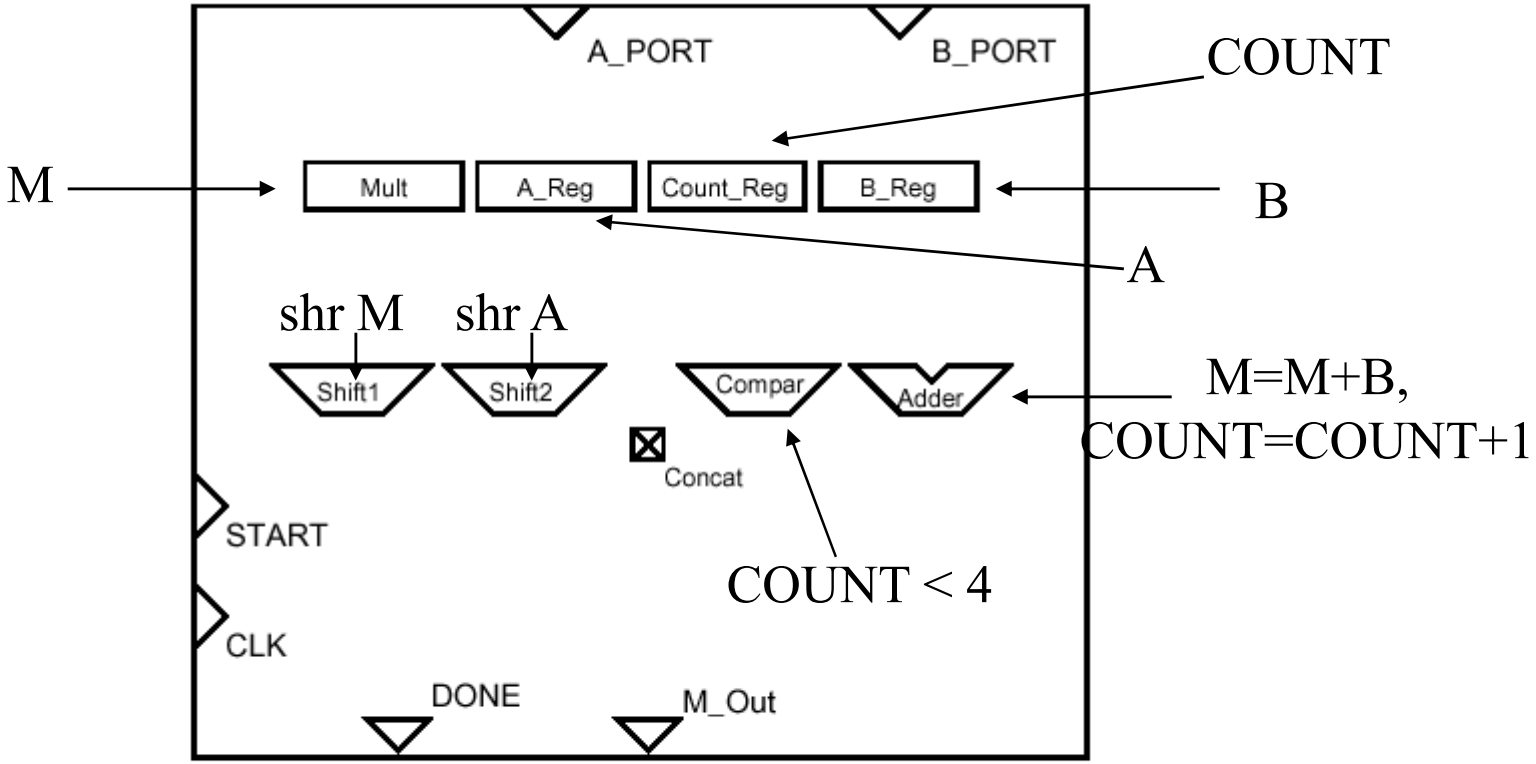
2. Επιλογή λειτουργικών μονάδων για λειτουργίες **πρόσθεσης**, **ολίσθησης** και **σύγκρισης**

Οι μονάδες θα πρέπει να δεσμευτούν (binding) δηλαδή να τους ανατεθεί κάποια λειτουργία



Initial Allocation

3rd step (b) = Unit Binding



Initial Allocation

4^{th} step = Connection

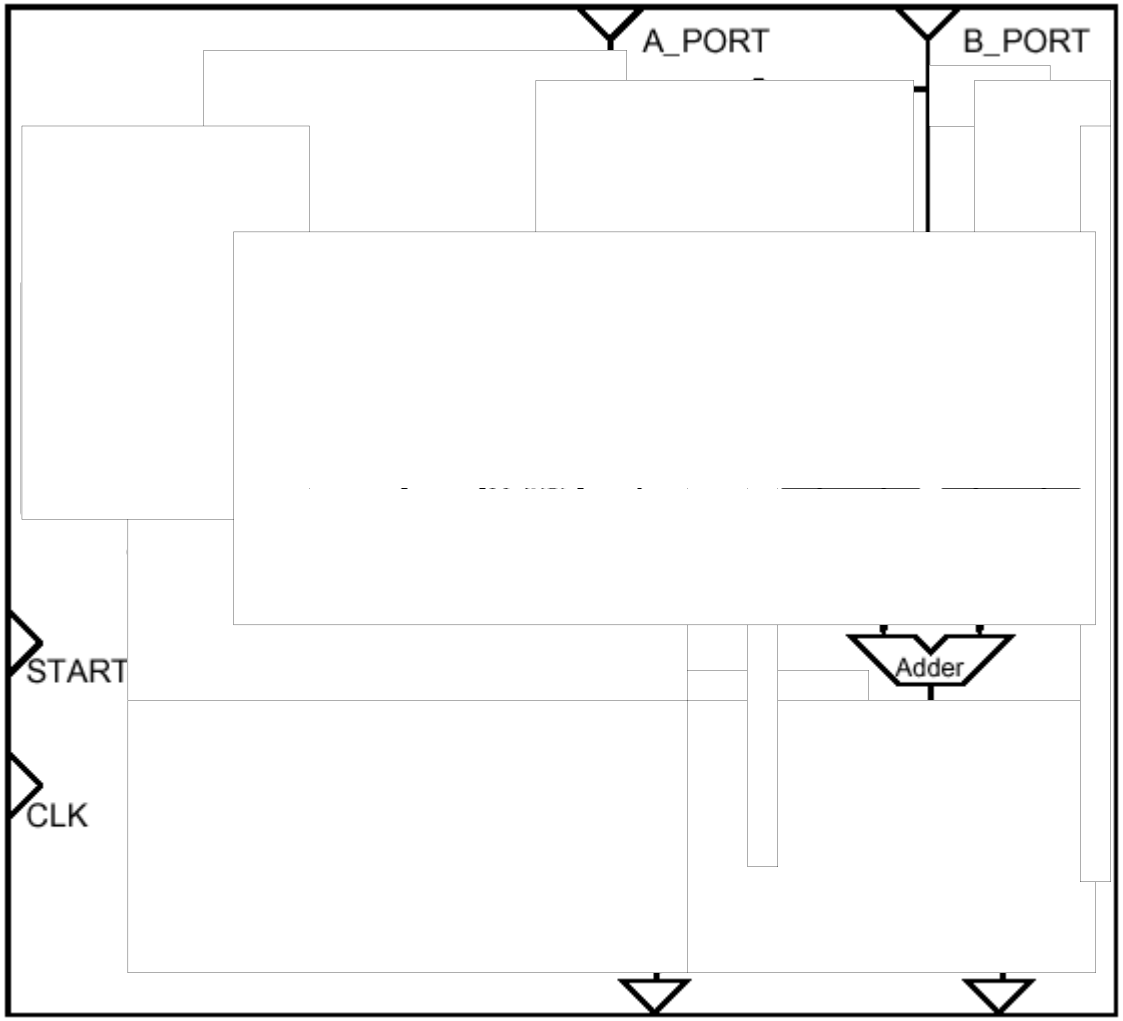
Κατά την διασύνδεση, όπου υπάρχουν πολλαπλές είσοδοι χρησιμοποιούνται πολυπλέκτες

Παράδειγμα: οι πράξεις

M=M+B,
COUNT=COUNT+1

Απαιτούν πολυπλέκτες στις εισόδους του αθροιστή

M, COUNT B, 1



5th step (a) = Control Unit FSMD

Στόχος της σύνθεσης της μονάδας ελέγχου είναι η δημιουργία των κατάλληλων **σημάτων ενεργοποίησης** των λειτουργιών στα διάφορα βήματα, και η υλοποίηση της **ακολουθίας** των βημάτων.

Αποστολή σημάτων λειτουργιών (**OP**) σε κάθε μονάδα.

Present State	Condition	Value	Actions	Next State
S0	START = 1	T	A := A_PORT; B := B_PORT; COUNT := 0; DONE := '0'; M := 0000";	S1
		F		S0
S1	COUNT < 4	T		S2
		F	M_OUT := M @ A; DONE := '1';	S0
S2	A(0) = 1	T	M := M + B;	S3
		F		S3
S3			A := SHR(A, M(0)); M := SHR(M. '0'); COUNT := COUNT + 1;	S1

FSMD state table

5th step (b) = Control Unit Component State-Table

Present State	Condition	Value	Actions	Next State
S1	Compar.LT	1		S2
		0	Concat(OP: concat, INPS: Mult, A_Reg); M_OUT(OP: load, INPS: Concat); Mux5(OP: c1, INPS: '0', '1'); DONE(OP: load, INPS: Mux5);	S0
S2	A_Reg(0)	1	Mux3(OP: c0, INPS: Mult, Count_Reg); Mux4(OP: c0, INPS: B_Reg, "0001"); Adder(OP: add, INPS: Mux3, Mux4); Mux1(OP: c1, INPS: Shift1, Adder); Mult(OP: load, INPS: Mux1);	S3
		0		S3

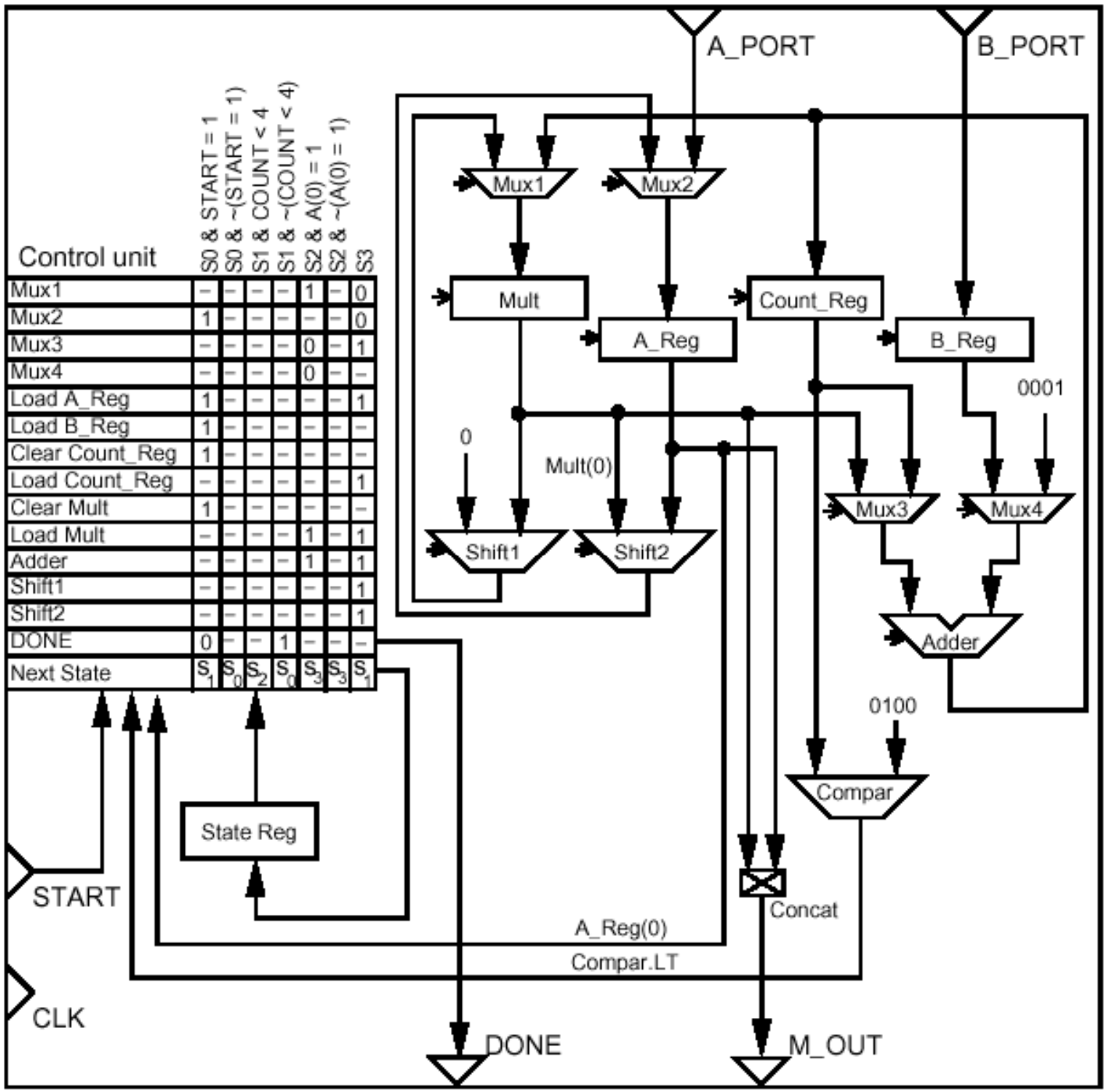
Πίνακας κατάστασης με ενεργοποιήσεις components / step

5th step (c)= Symbolic Control Table

Present State	Condition	Value	Actions	Next State
S1	Compar.LT	1		S2
		0	DONE := 1;	S0
S2	A_Reg(0)	1	Mux3.sel := 0; Mux4.sel := 0; Adder.add := 1; Mux1.sel := 1; Mult.load := 1;	S3
		0		S3

Πίνακας κατάστασης με ενεργοποιήσεις γραμμών ελέγχου

Αποτέλεσμα Σύνθεσης



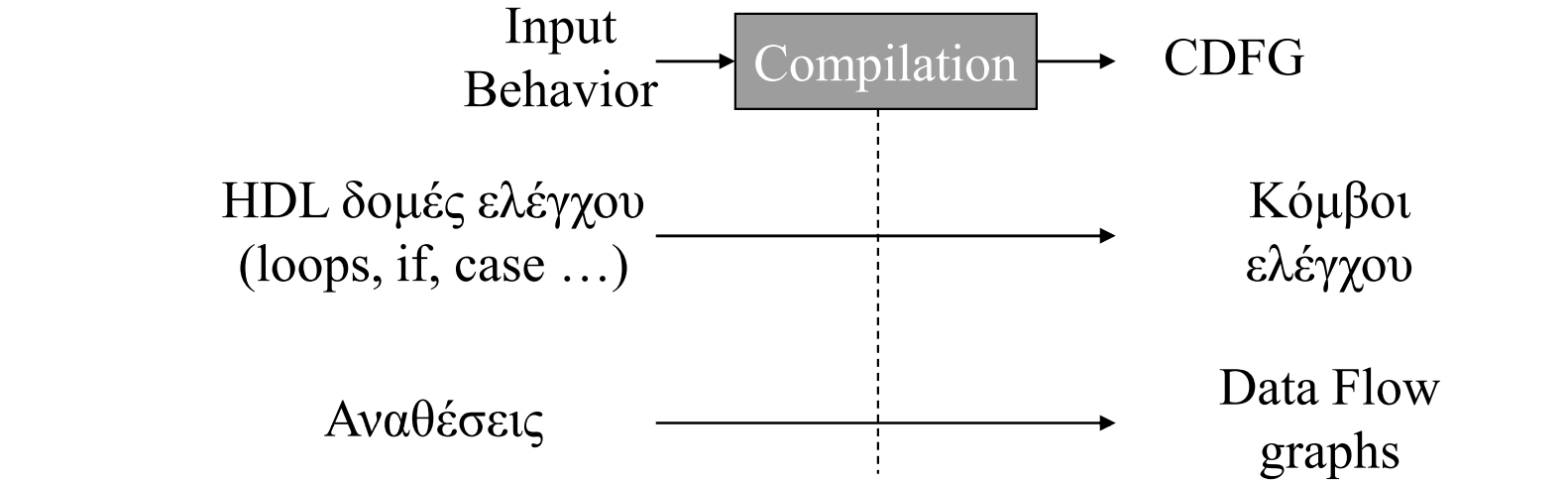
Σύνθεση

- ✓ Ξεκινώντας από την περιγραφή συμπεριφοράς προσθέσαμε πληροφορία κατά την σύνθεση:

  μονάδες λειτουργικές, αποθήκευσης, διασυνδέσεις, ακολουθιακότητα εκτέλεσης κλπ.

- ✓ Συσχετίσαμε αυτήν την πληροφορία με την περιγραφή.
- ✓ Η τελική αναπαράσταση παρέχει όλα τα ενδιάμεσα βήματα της σύνθεσης, και τις συσχετίσεις τους.

HDL - Compilation

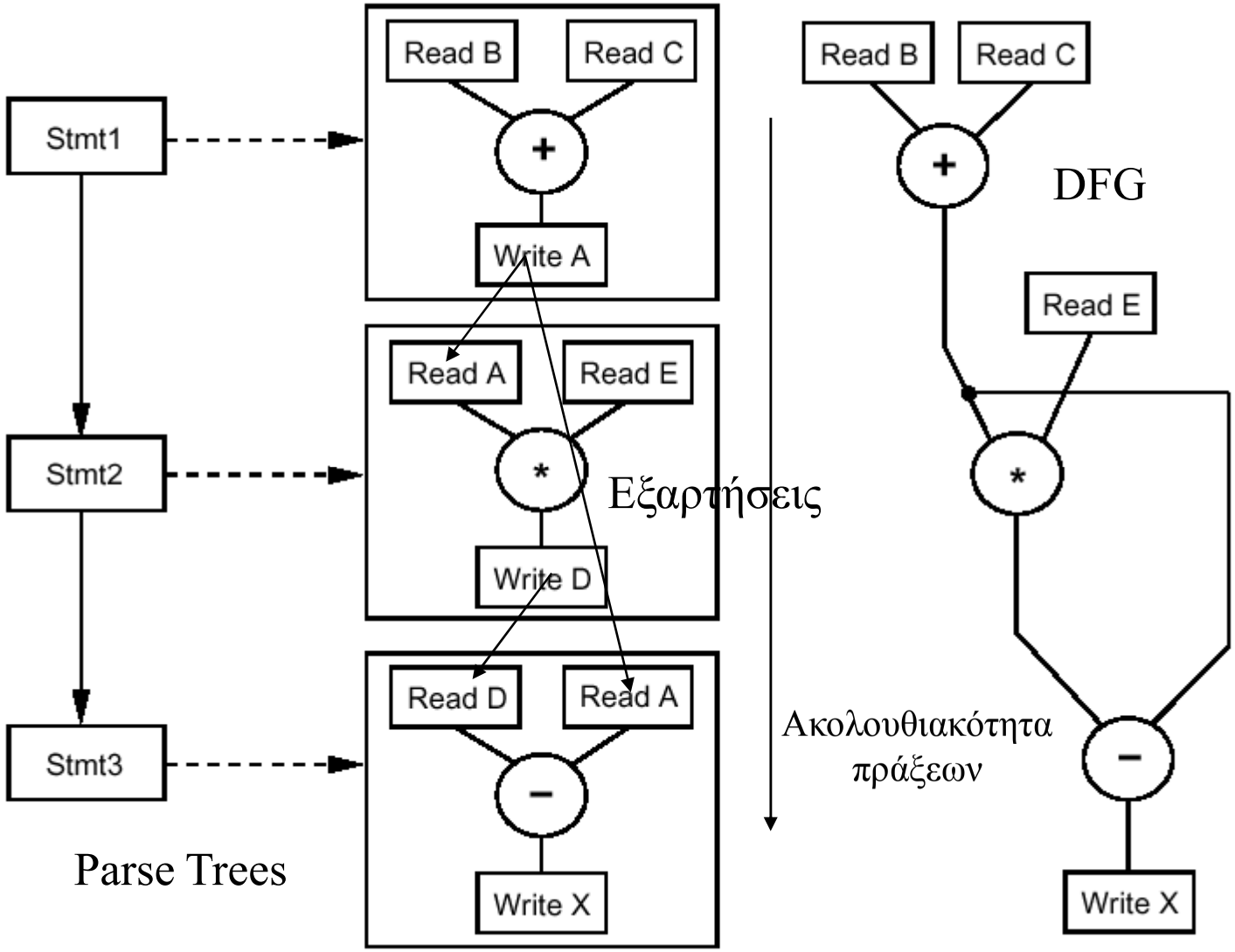


Παράδειγμα:

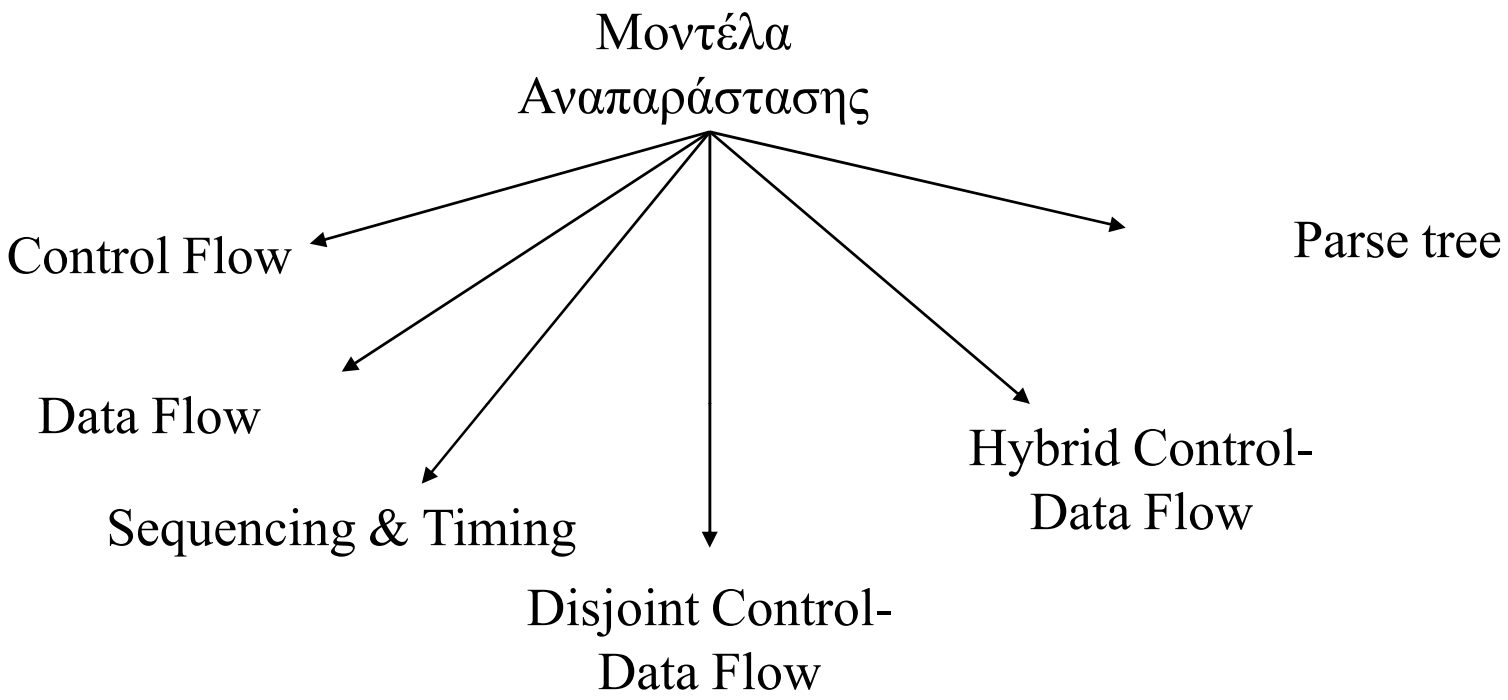
```
A := B + C;
D := A * E;
X := D – A;
```

Περιγραφή με
ακολουθιακότητα
λόγω εξαρτήσεων

HDL - Compilation



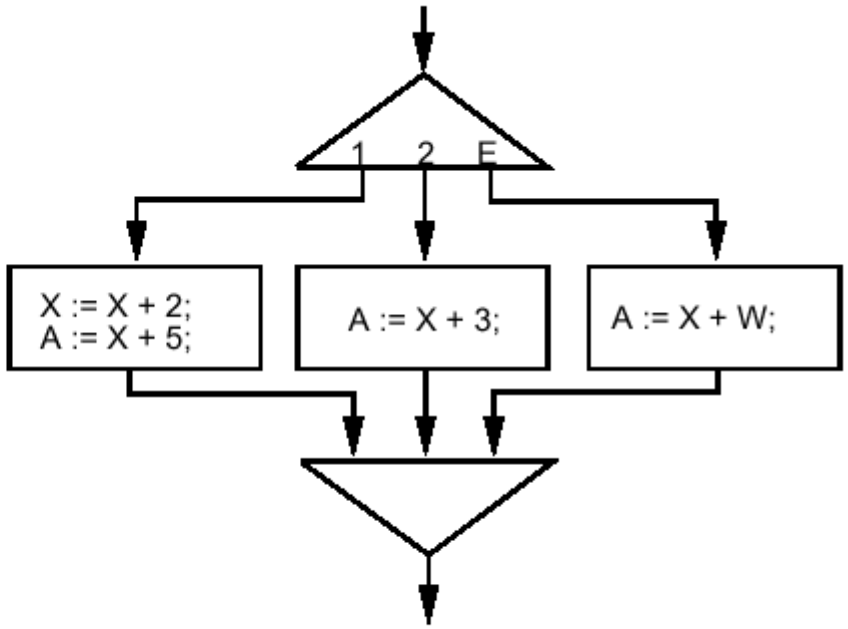
Αναπαράσταση HDL συμπεριφοράς



(1) Control-Flow representation

```
case C is
    when 1 => X := X + 2;
              A := X + 5;
    when 2 => A := X + 3;
    when others => A := X + W;
end case;
```

VHDL Description



Control flow representation

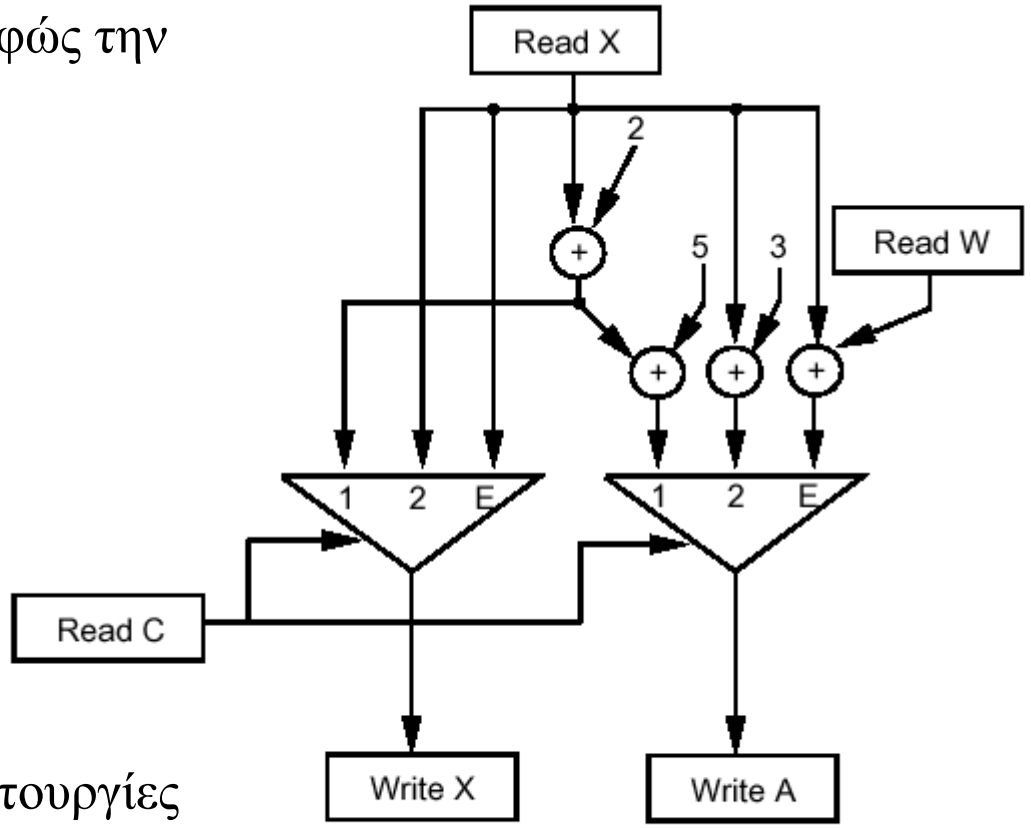
Η αναπαράσταση αυτή δείχνει σαφώς την ροή ελέγχου (πχ. αμοιβαίος αποκλεισμός όπως υλοποιείται το case) αλλά όχι την ροή δεδομένων.

(2) Data-Flow representation

Η αναπαράσταση αυτή δείχνει σαφώς την ροή δεδομένων.

```
case C is
    when 1 => X := X + 2;
              A := X + 5;
    when 2 => A := X + 3;
    when others => A := X + W;
end case;
```

VHDL Description



Dataflow representation

Κύκλοι = λειτουργίες

Παραλ\μα = αναθέσεις

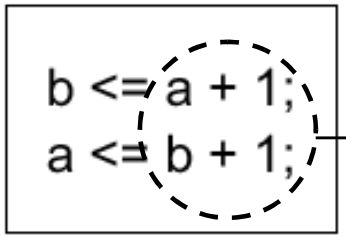
Τρίγωνα = επιλογή

Σύγκριση

- ✓ Και οι δύο αναπαραστάσεις δείχνουν τον παραλληλισμό λόγω αμοιβαίου αποκλεισμού μονοπατιών.
- ✓ Στην Data flow η εκτίμηση όλων των μονοπατιών γίνεται παράλληλα και βοηθά το scheduling.
- ✓ Η Data flow δεν βοηθά στην επαναδόμηση της αρχικής περιγραφής, αφού ενσωματώνει την πληροφορία ελέγχου στον γράφο.
- ✓ Οι nested περιγραφές ελέγχου, μπορούν να οδηγήσουν σε περίπλοκο γράφο, με πολλά επίπεδα επιλογέων, αφού γίνεται επιπεδοποίηση τους (flattening).

Sequencing

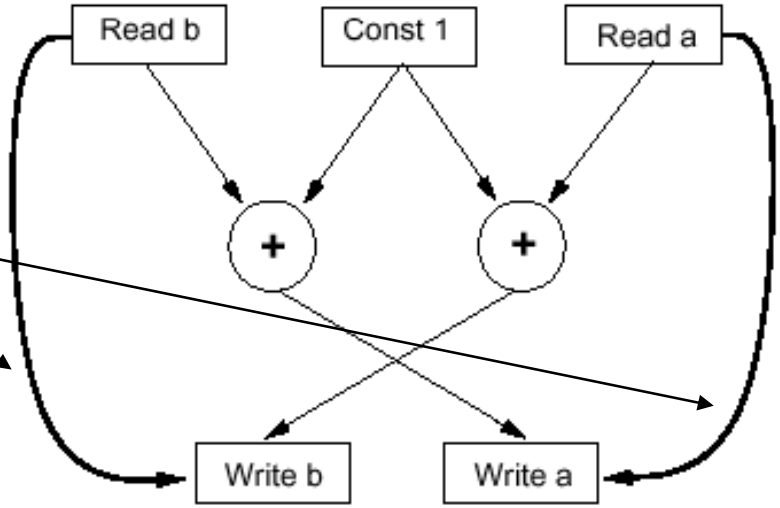
Πολλές φορές είναι σημαντικό να επιβάλλεται η ακολουθιακότητα. Πχ. εναλλαγή τιμών



Concurrent VHDL

Ταυτόχρονα οι πράξεις, αλλά οι εγγραφές μετά τις αναγνώσεις

Τα **έντονα βέλη (precedence arcs)** δείχνουν ποιες πράξεις προηγούνται από άλλες.



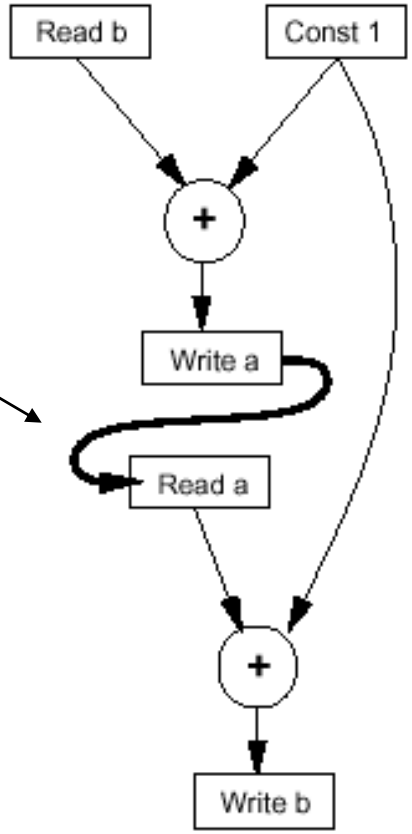
Sequencing

Ακολουθιακές πράξεις.

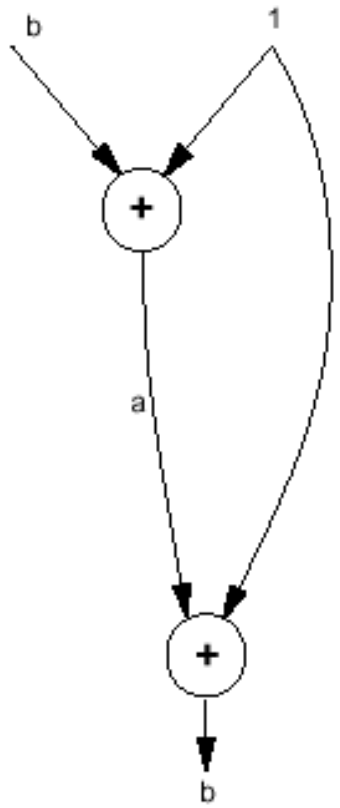
```
a := b + 1;
b := a + 1;
```

Seq. VHDL

Εδώ θέλουμε την εκτέλεση της πρώτης και μετά της δεύτερης (b:=b+2)

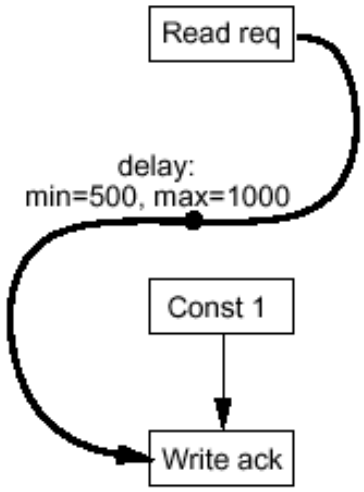


Σαφής αναπαράσταση ακολουθιακότητας

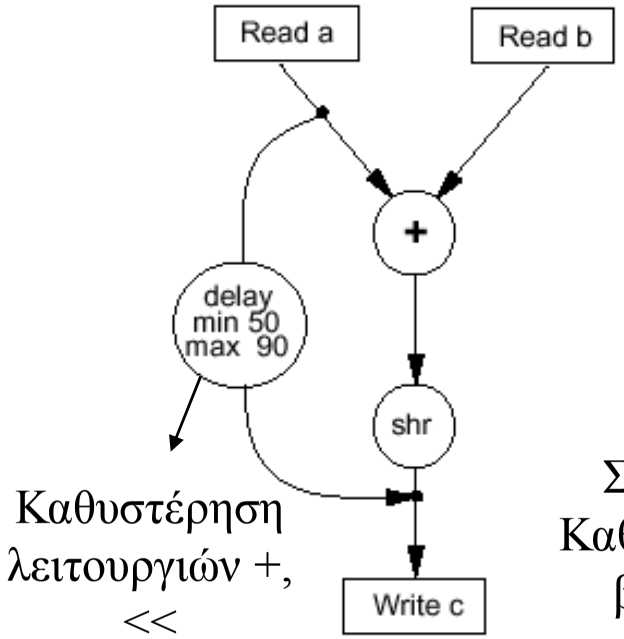


Υπονοούμενη αναπαράσταση ακολουθιακότητας

Timing

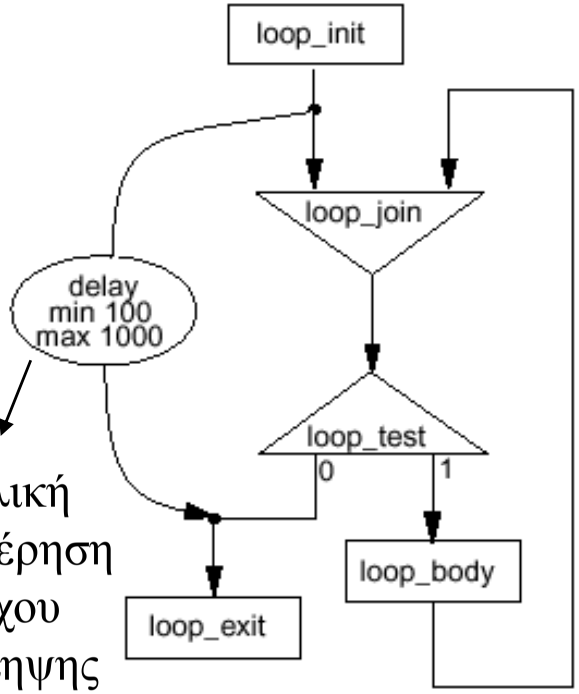


Dataflow annotation



Καθυστέρηση λειτουργιών +, <<

Timing in DFG



Συνολική Καθυστέρηση βρόγχου επανάληψης

Timing in CFG

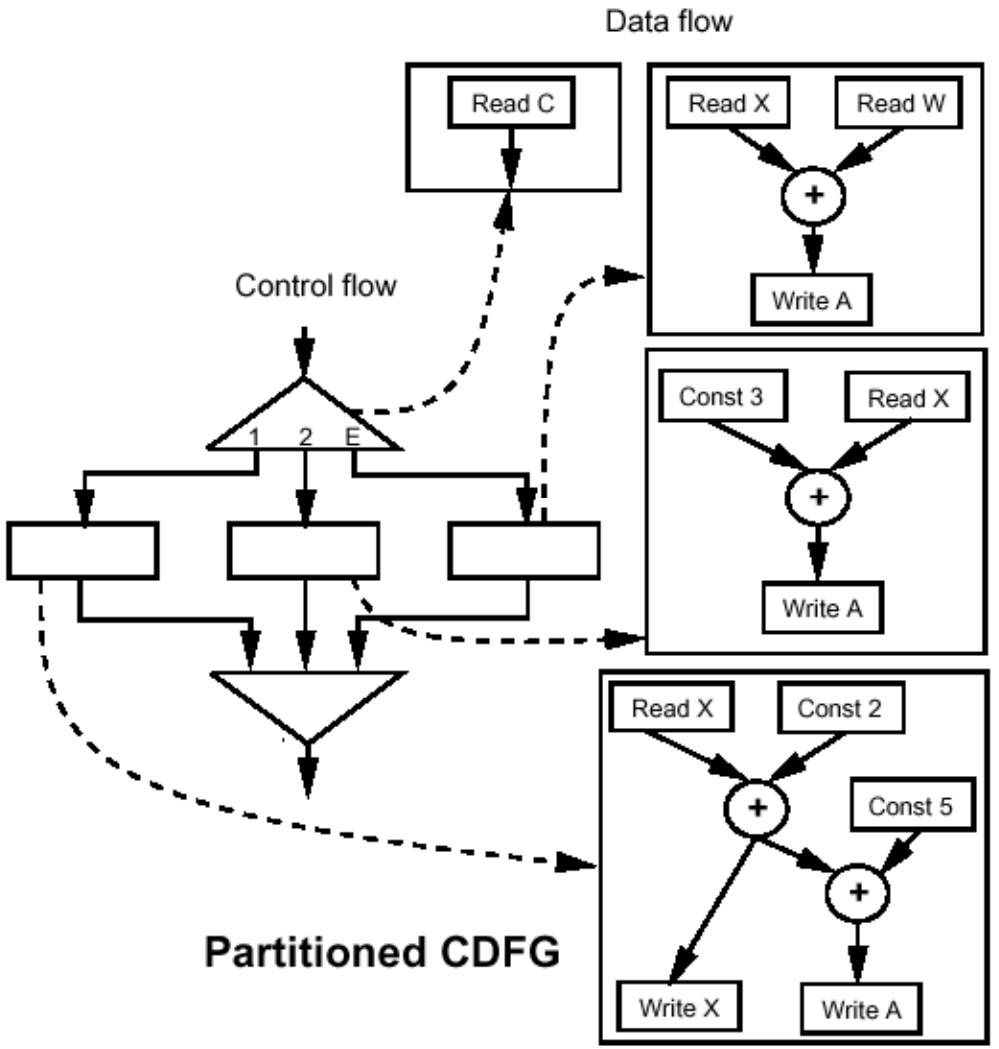
Η χρονική πληροφορία χρησιμοποιείται σαν περιορισμός για scheduling, unit selection και unit binding και φυσικά περιορισμός απόδοσης.

(3) Χωριστές Αναπαραστάσεις Control-Data Flow

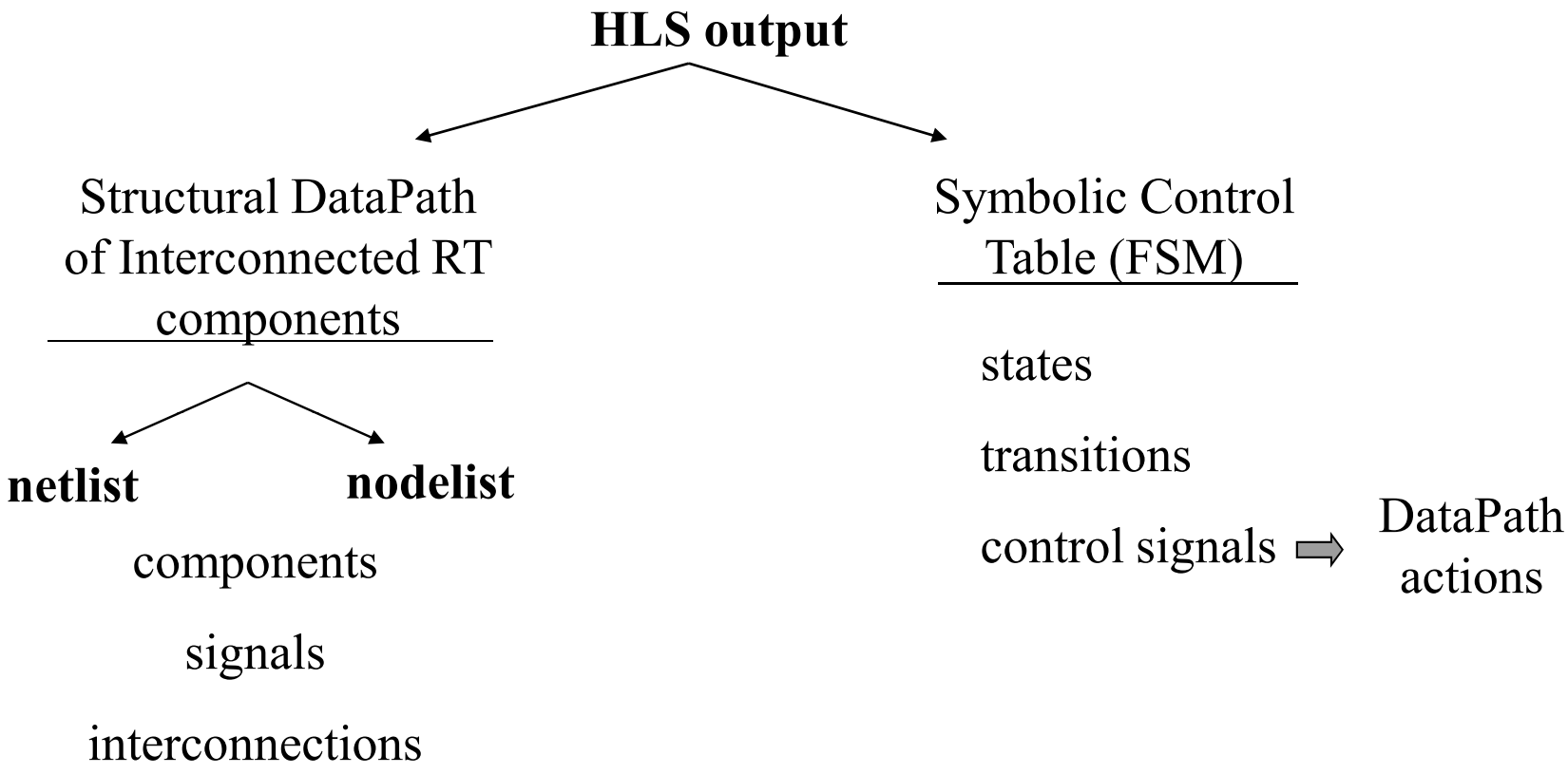
```
1. case C is
2.     when 1 =>  X := X + 2;
3.                A := X + 5;
4.     when 2 =>  A := X + 3;
5.     when others => A := X + W;
6. end case;
```

VHDL Description

Οι αναπαραστάσεις δεδομένων και ελέγχου γίνονται χωριστά



Αναπαράσταση Αποτελεσμάτων Σύνθεσης



Compilation – Behavioral Optimization

Ένας compiler σε software αποτελείται από δύο τμήματα:

- **front end**: μεταφράζει ένα πρόγραμμα σε ενδιάμεση μορφή (εξαρτάται από την γλώσσα προγραμματισμού).
- **back end**: μετατρέπει την ενδιάμεση μορφή σε κώδικα μηχανής (εξαρτάται από την χρησιμοποιούμενη μηχανή).

Ένας compiler για hardware αποτελείται επίσης από front end και back end:

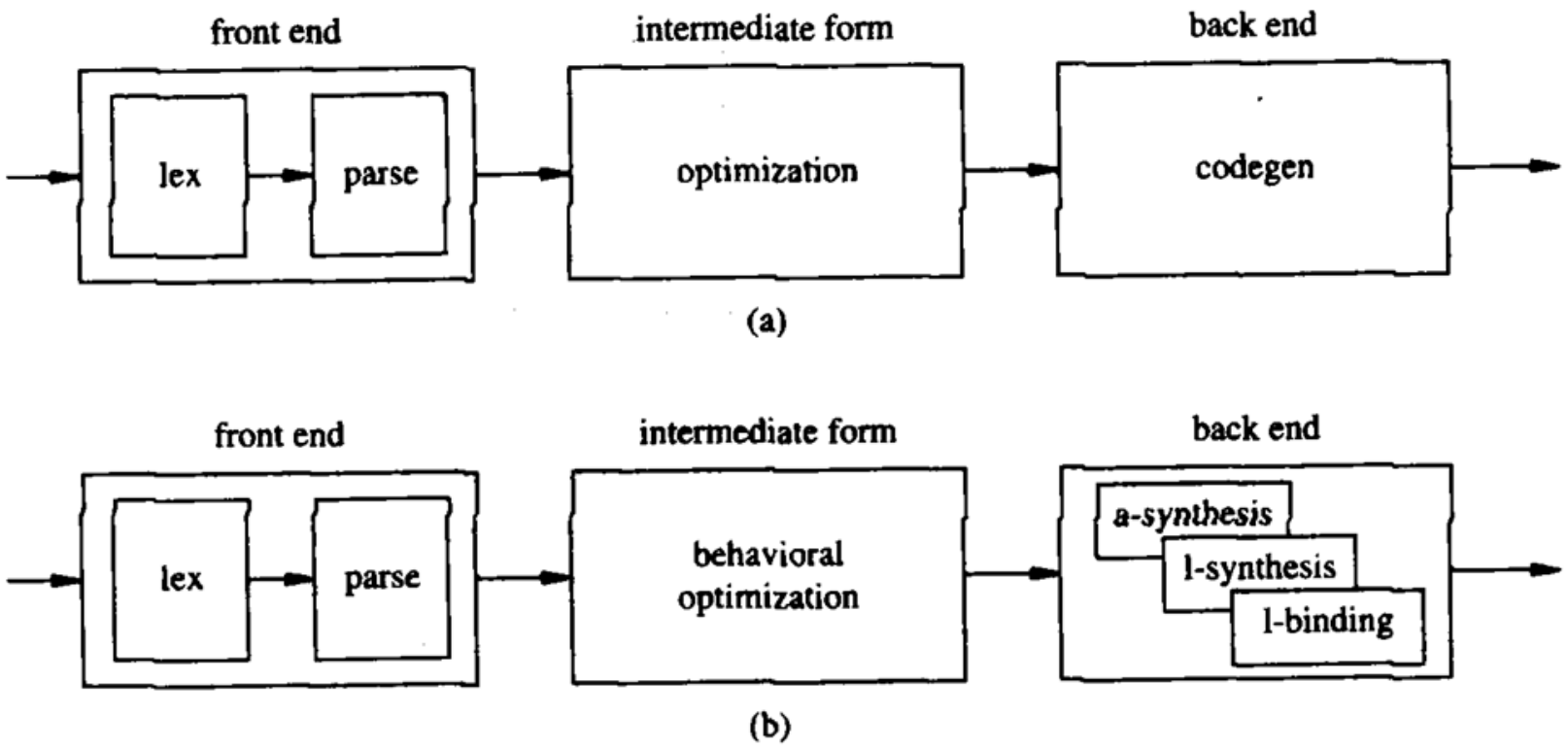
- ✓ Το back end είναι πιο περίπλοκο εδώ εξαιτίας των εσωτερικών απαιτήσεων σε χρονισμούς.
- ✓ Το front end είναι υπεύθυνο για:

  (α) λεκτική-συντακτική ανάλυση

  (β) παραγωγή της ενδιάμεσης μορφής

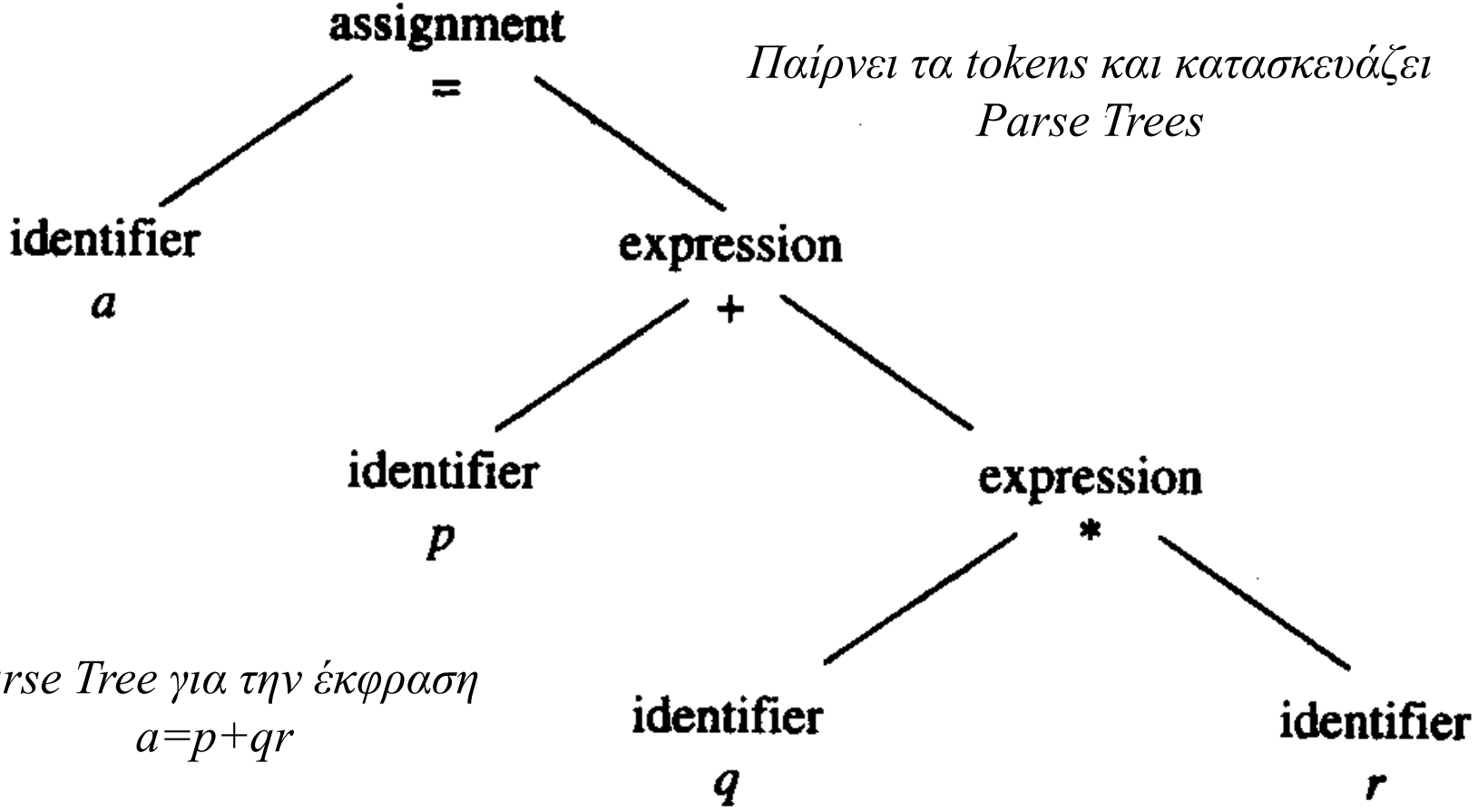
  (γ) παράγει parse trees με την ανάλυση των εκφράσεων

Compilation – Behavioral Optimization



Parser

Παίρνει τα tokens και κατασκευάζει Parse Trees



Parse Tree για την έκφραση a=p+qr

Compilation

Μεταγλώττιση structural περιγραφής:

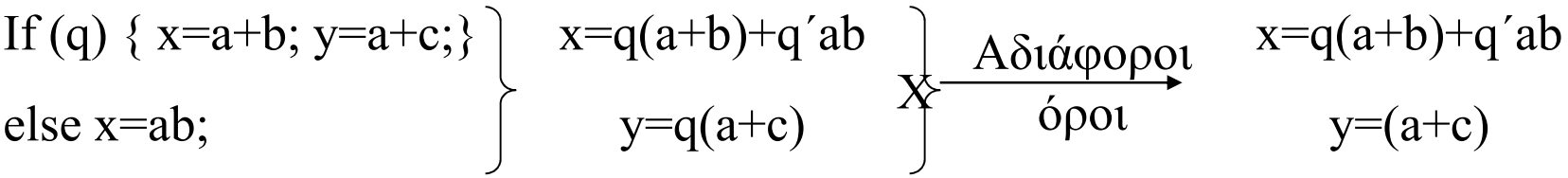
- Η περιγραφή δείχνει την σχέση μεταξύ pins (modules) και γραμμών διασύνδεσης.
- Το parse tree μπορεί εύκολα να μετατραπεί σε incidence matrix ή netlist.

Μεταγλώττιση περιγραφής συνδυαστικών λογικών κυκλωμάτων:

- ✓Στην απλούστερη περίπτωση έχουμε περιγραφή με λογικές συναρτήσεις.
- ✓Στις λογικές συναρτήσεις υπάρχει άμεση αντιστοιχία με την structural περιγραφή (διασυνδεόμενες πύλες).
- ✓Όταν χρησιμοποιούνται procedural assignments (μοντέλο συμπεριφοράς) τότε δεν υπάρχει άμεση αντιστοιχία (πχ δομές διακλάδωσης).

Compilation

Πχ. Δομές διακλάδωσης: μπορούν να αντικατασταθούν με λογικές εκφράσεις



- Συχνά οι δομές διακλάδωσης ελέγχουν την τιμή μία μεταβλητής τύπου απαρίθμησης.
- Κατά την σύνθεση η μεταβλητή πρέπει να παίρνει δυαδικές τιμές για να δίνει λογικό κύκλωμα.
- Η κωδικοποίηση μπορεί να γίνει κατά την περιγραφή ή από την σύνθεση με την κατάλληλη οδηγία.

Behavioral Optimization

Το πρώτο στάδιο βελτιστοποίησης γίνεται άμεσα στην περιγραφή της συμπεριφοράς. Φυσικά ακολουθούν και επόμενα στάδια.

- ✓ Δεν απαιτείται γνώση για την υλοποίηση του κυκλώματος.
- ✓ Επαναδιάταξη ενδιάμεσου κώδικα.
- ✓ Οι αλγόριθμοι βελτιστοποίησης συμπεριφοράς κατηγοριοποιούνται σε

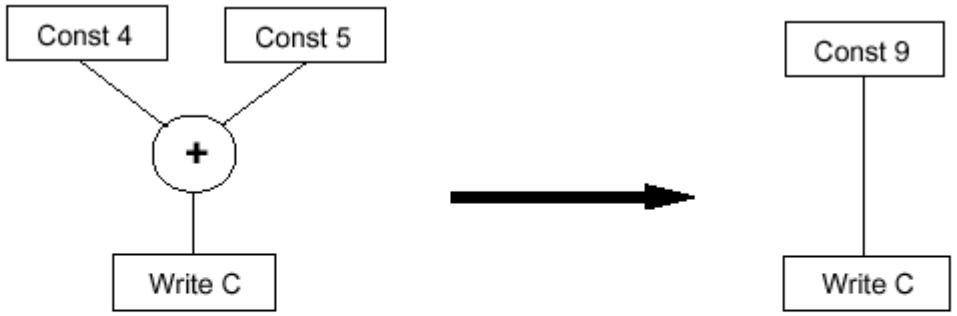
  (α) αλγορίθμους βελτιστοποίησης ροής δεδομένων και

  (β) αλγορίθμους βελτιστοποίησης ροής ελέγχου.
- ✓ Για την βελτιστοποίηση λαμβάνεται υπόψη το κόστος των resources (πχ ένας πολλαπλασιαστής είναι πολύ μεγαλύτερος από μία λογική μονάδα)

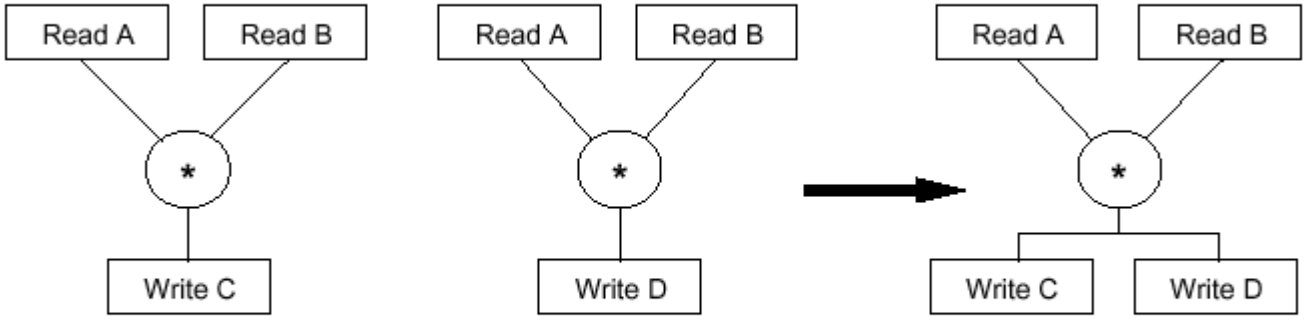
Transformations

- ✓ Η βελτιστοποίηση επιτυγχάνεται με εφαρμογή διαφόρων μετασχηματισμών.
- ✓ Οι μετασχηματισμοί εφαρμόζονται σε διάφορα στάδια της σύνθεσης:
  1. Κατά την μεταγλώττιση (compiler optimizations)
  2. Flow Graph transformations: μετατροπή τμημάτων της αναπαράστασης από control flow σε data flow με αλλαγή του βαθμού παραλληλισμού.
  3. Hardware specific transformations (πχ. αντικατάσταση του a*2 με ολίσθηση.

Compiler Transformations



Constant folding



Redundant operator elimination

Compiler Transformations

Εξάλειψη άχρηστου κώδικα

Άχρηστος κώδικας = λειτουργίες που δεν μπορούν να εκτελεστούν ποτέ ή τα αποτελέσματά τους δεν αναφέρονται πουθενά.

Μείωση κόστους τελεστή

Όταν ένας τελεστής μπορεί να αντικατασταθεί με έναν απλούστερο χωρίς να αλλάξει το αποτέλεσμα τότε έχουμε βελτιστοποίηση.

Πχ το y=3x μπορεί να γίνει t=x<<1 και y=t+x;

Μετακίνηση κώδικα

Ένα τμήμα κώδικα στο εσωτερικό ενός βρόγχου με τιμή ανεξάρτητη επαναλήψεων, μπορεί να τοποθετηθεί εκτός του βρόγχου.

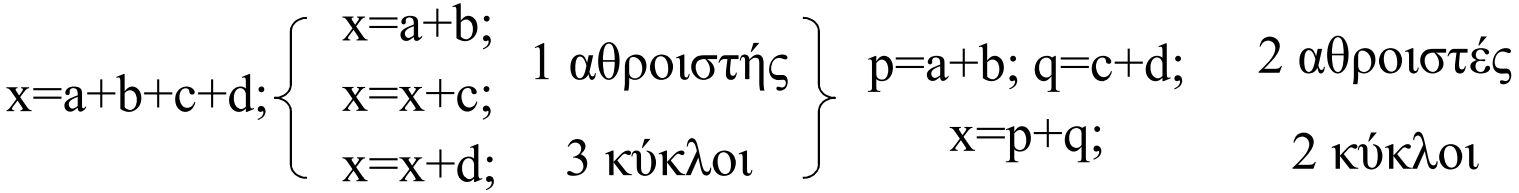
Εξάλειψη κοινών υποεκφράσεων.

Υπολογισμός κάθε υποέκφρασης μία φορά

Flow Graph Transformations

1. Μείωση ύψους δέντρου

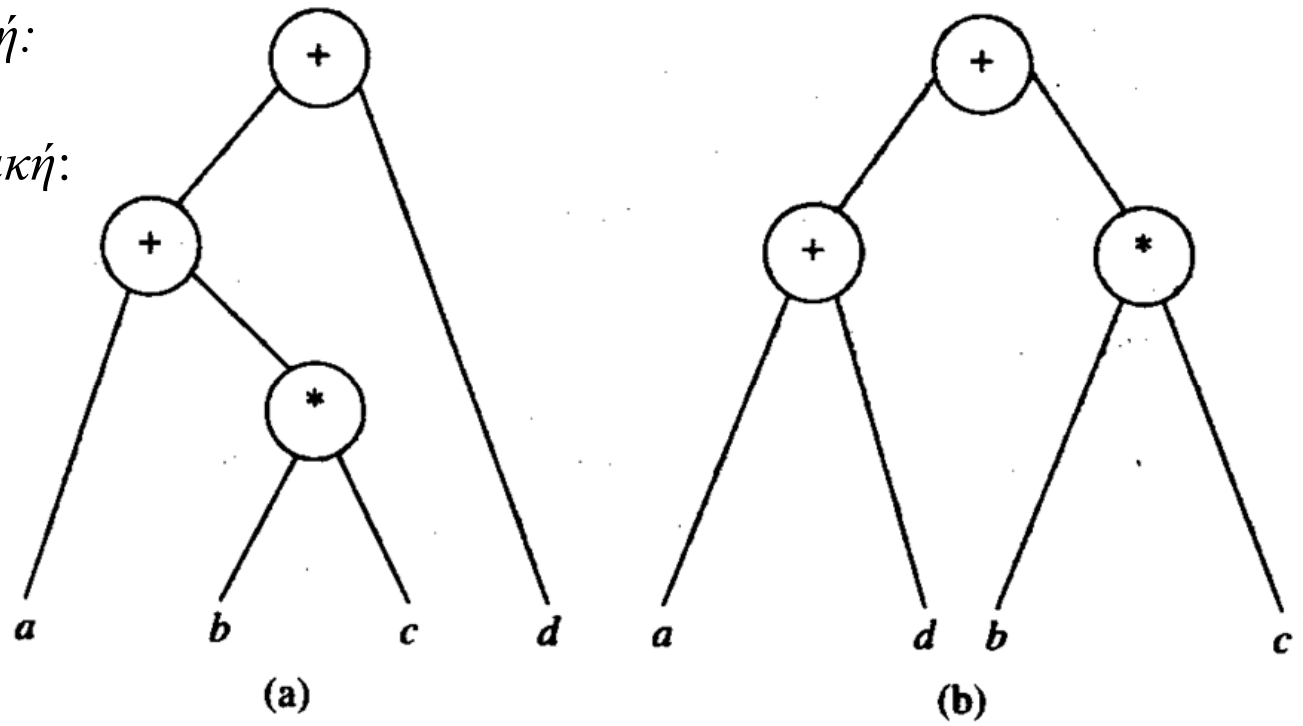
Διαίρεση έκφρασης σε υποεκφράσεις για να γίνεται εκμετάλλευση του παραλληλισμού.



- ✓ Ο απλούστερος αλγόριθμος μείωσης χρησιμοποιεί την αντιμεταθετική και προσεταιριστική ιδιότητα της πρόσθεσης και του πολλαπλασιασμού.
- ✓ Μπορεί να χρησιμοποιηθεί και η επιμεριστική ιδιότητα με το μειονέκτημα πρόσθεσης μίας επιπλέον λειτουργίας.

Flow Graph Transformations

x=a+bc+d;
Αντιμεταθετική:
x=a+d+bc
Προσεταιριστική:
x=(a+d)+bc

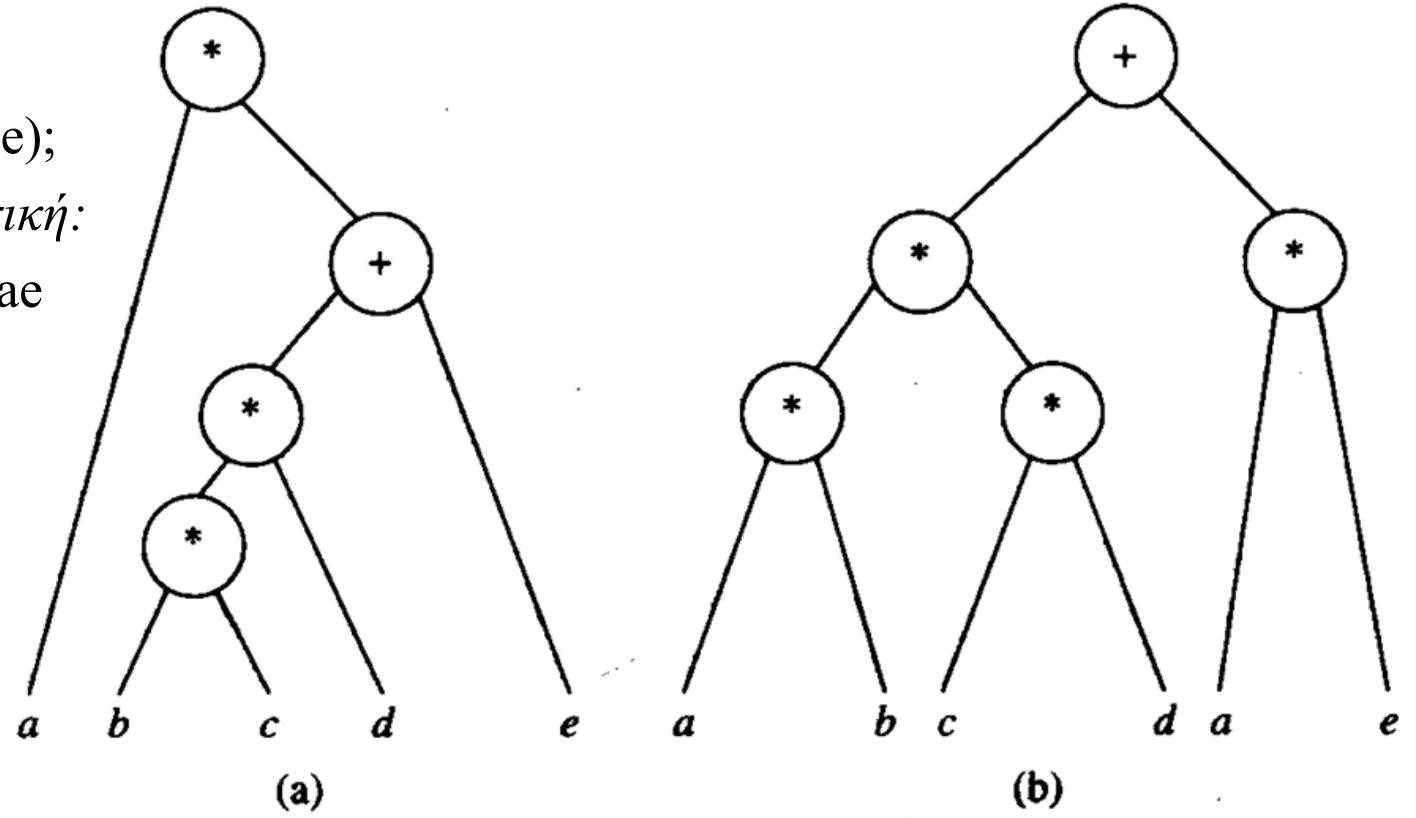


Flow Graph Transformations

x=a(bcd+e);

Επιμεριστική:

x= abcd+ae



Flow Graph Transformations

2. Μετατροπή ροής ελέγχου σε ροή δεδομένων

- ✓ Χρησιμοποιείται όταν ελέγχουμε (if) boolean εκφράσεις.
- ✓ Ο έλεγχος μεταφέρεται στο τέλος (οι διακλαδώσεις εκτελούνται παράλληλα και επιλέγεται το σωστό αποτέλεσμα)
- ✓ Στον γράφο ροής δεδομένων ο παραλληλισμός είναι σαφής.
- ✓ Απαιτείται επιπλέον υλικό

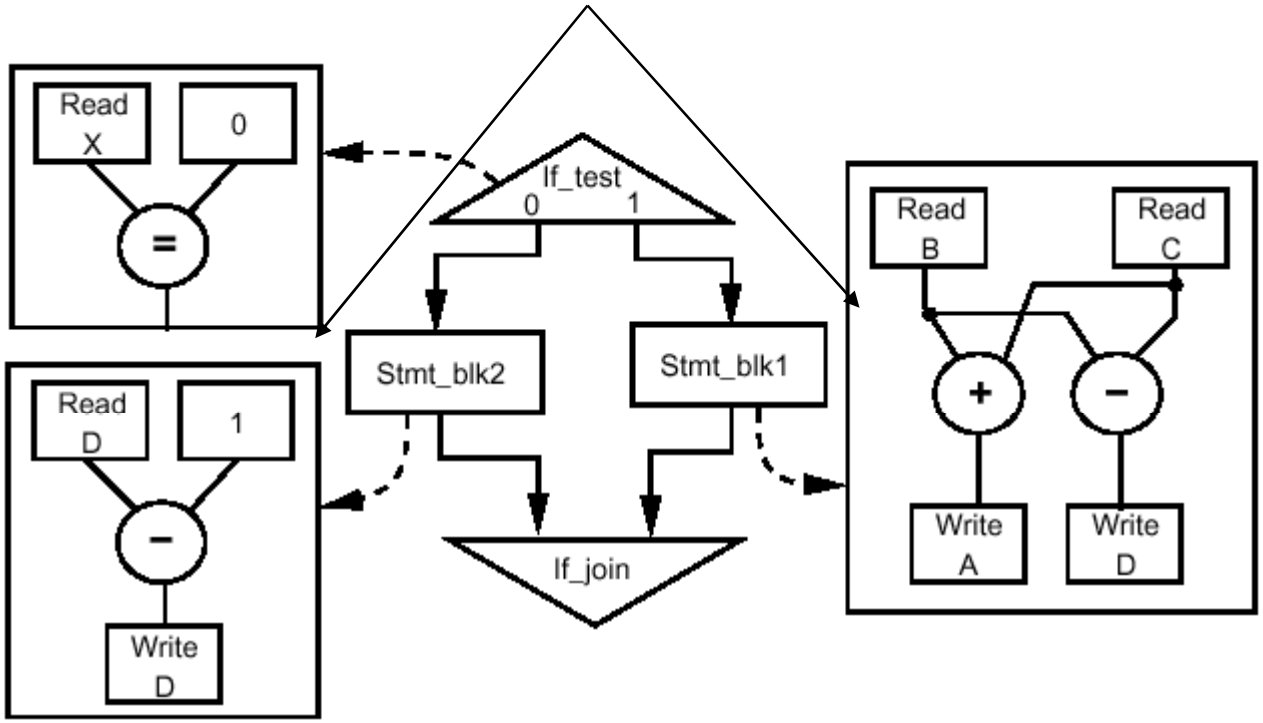
Παράδειγμα

```
if (X = 0) then
      A := B + C;
      D := B − C;
else
      D := D − 1;
end if;
```

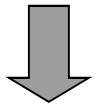
Textual representation

Flow Graph Transformations

Έλεγχος Πριν: Μπορεί να χρησιμοποιηθεί ένας '+' και ένας '-'

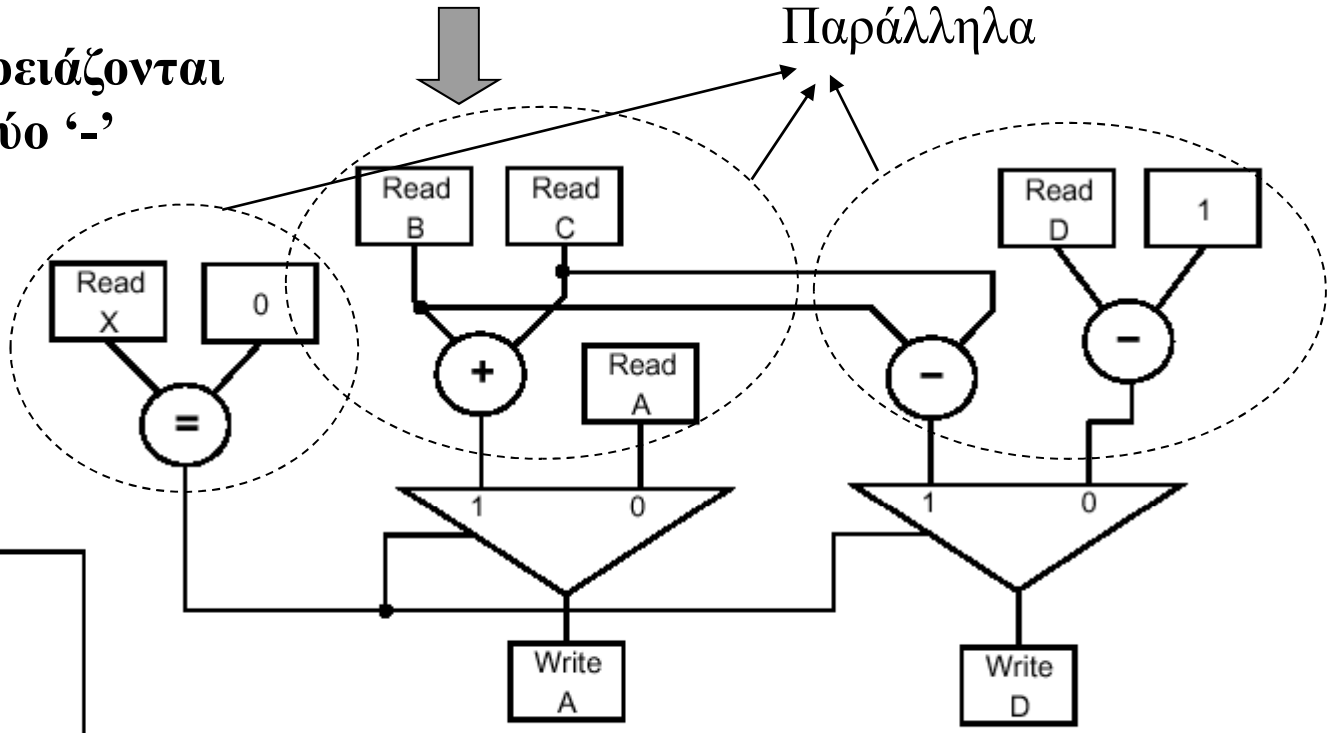


CF representation



Flow Graph Transformations

Έλεγχος Μετά: Χρειάζονται ένας '+' και δύο '-'



DF representation

```
if (X = 0) then
      A := B + C;
      D := B − C;
else
      D := D − 1;
end if;
```

Textual representation

Απαιτείται επιπλέον υλικό αλλά κερδίζει σε χρόνο αφού η σύγκριση γίνεται παράλληλα με τις πράξεις

Flow Graph Transformations

3. Model expansion

- Η τμηματοποίηση του κώδικα με χρήση υπορουτινών έχει πολλά πλεονεκτήματα.
- Πολλές φορές όμως η αφαίρεσή της (flattening) οδηγεί σε τοπικά καλύτερες λύσεις (αύξηση επίδρασης εμβέλειας).
- Γίνεται αυτόματα από τον compiler για την καλύτερη εκμετάλλευση κοινών υποεκφράσεων.
- Εάν η κλήση μίας υπορουτίνας γίνεται μία φορά τότε σίγουρα υπάρχει κέρδος. Εάν γίνεται περισσότερες τότε μπορεί να υπάρχει απώλεια εξαιτίας της μείωσης στην διαμοίραση του hardware.

x=a+b; y=a*b; z=foo(x,y); με foo(p,q){ t=q-p; return t; }

Με model expansion:

x=a+b; y=a*b; z=y-x;

Flow Graph Transformations

4. Loop expansion (unrolling)

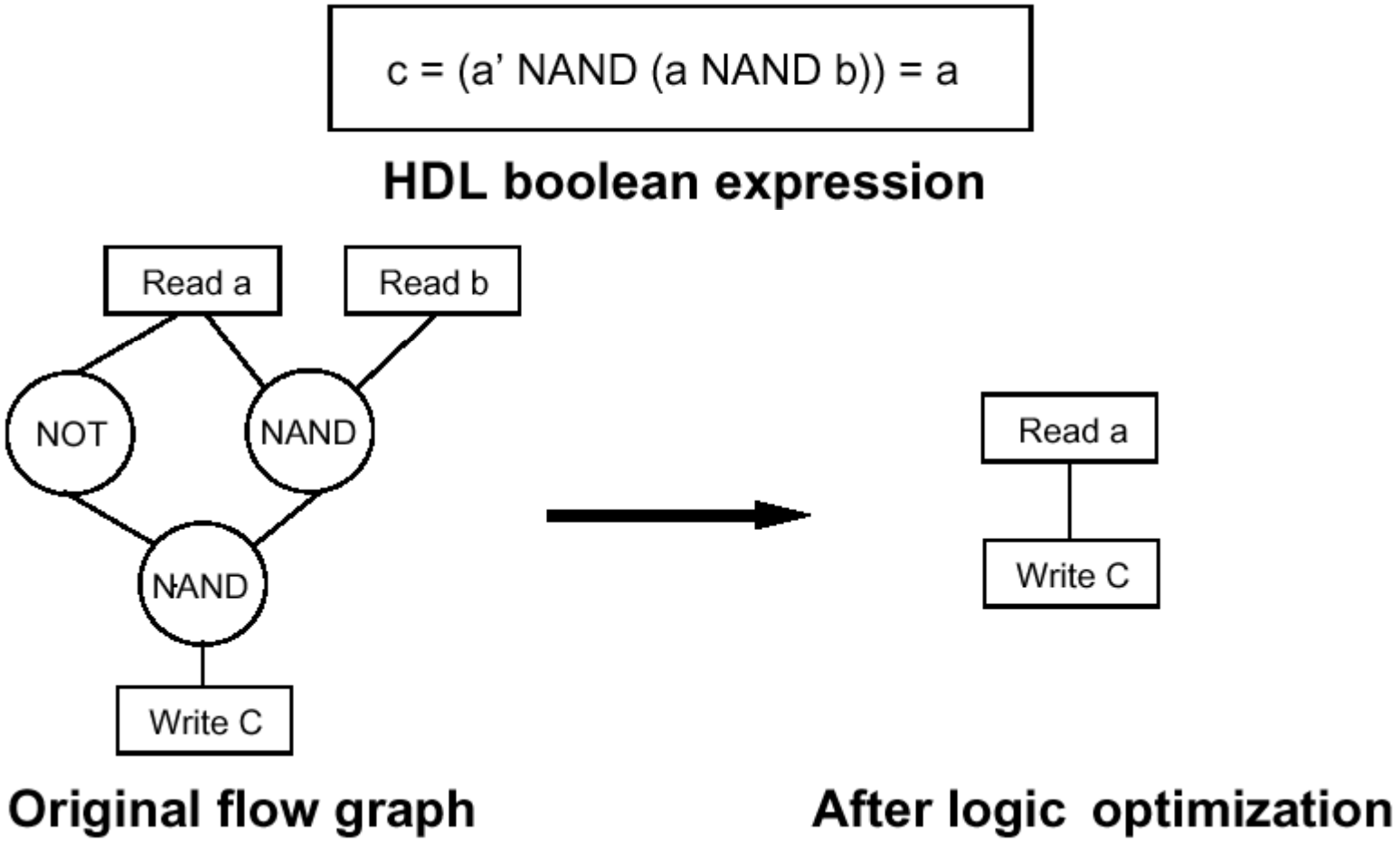
- Μία επαναληπτική δομή με σταθερό αριθμό επαναλήψεων μπορεί να αντικατασταθεί από τόσες επαναλήψεις του σώματος της επανάληψης.
- Για μεγάλο αριθμό επαναλήψεων έχουμε αρκετό κώδικα.

Παράδειγμα :

x=0; for (i=1; i≤3; i++) x=x+a[i]; μπορεί να γίνει x=a[1]+a[2]+a[3];

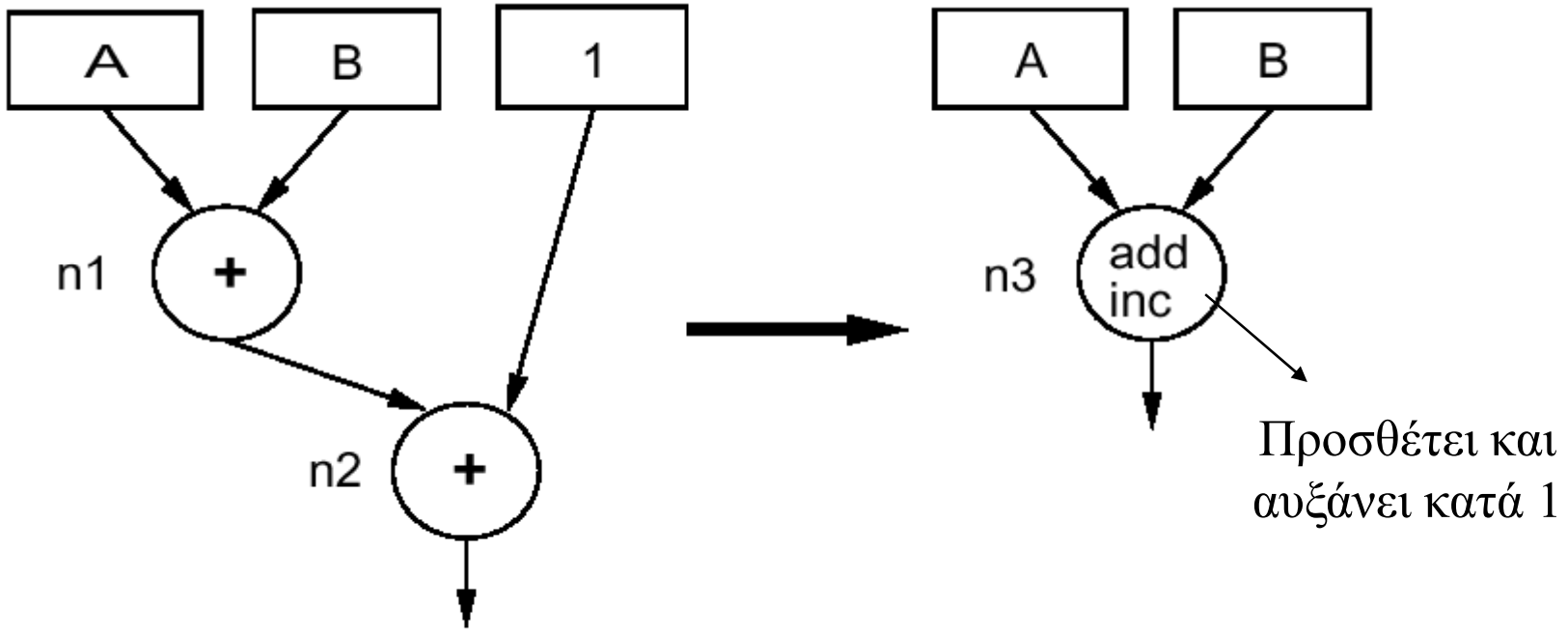
Hardware Transformations

Στο λογικό επίπεδο εφαρμόζουμε τεχνικές βελτιστοποίησης Boolean.



Hardware Transformations

Στο επίπεδο RT χρησιμοποιούμε pattern matching για να αντικαταστήσουμε τμήματα του flow graph με απλούστερα τα οποία υποστηρίζονται από υπάρχουσες δομές hardware.

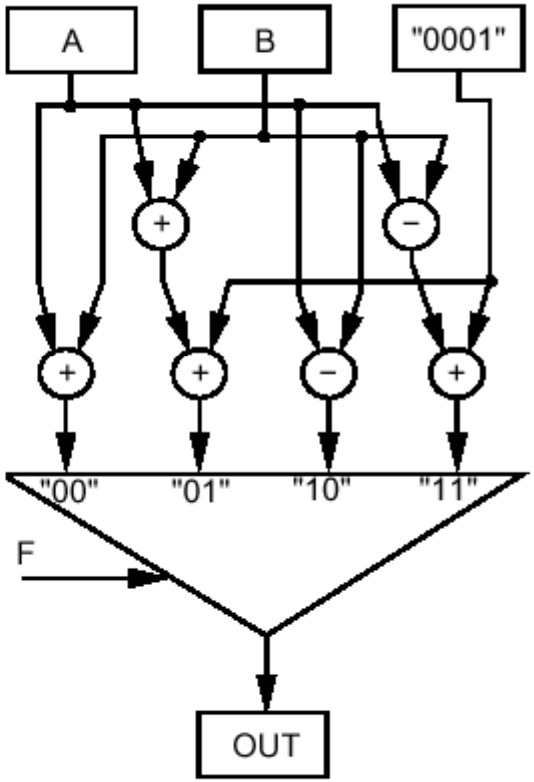


Adder/Incrementer Transformation

Hardware Transformations

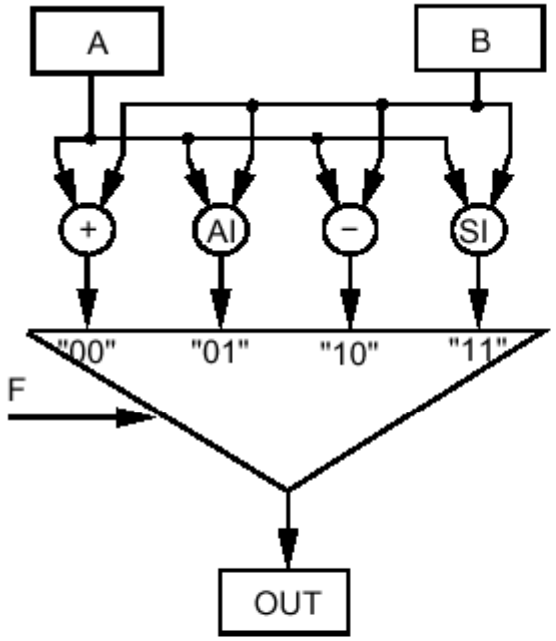
Ομάδες λειτουργιών μπορούν να ανατεθούν σε multifunctional nodes

```
case F is
        when "00" => OUT <= A + B;
        when "01" => OUT <= A + B + "0001";
        when "10" => OUT <= A − B;
        when "11" => OUT <= A − B + "0001";
end case;
```

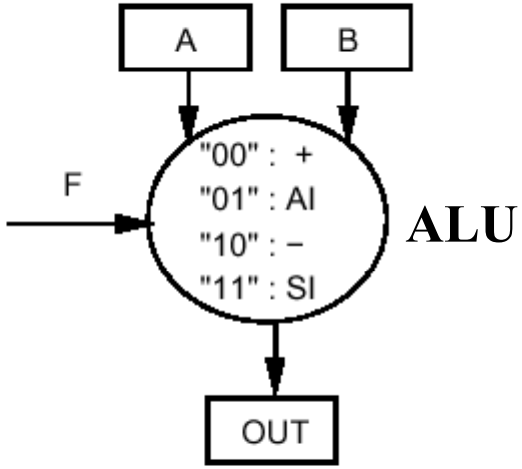


VHDL **DF graph**

Hardware Transformations



Simplified DF graph



Complex−node DF graph