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

Data Path Allocation

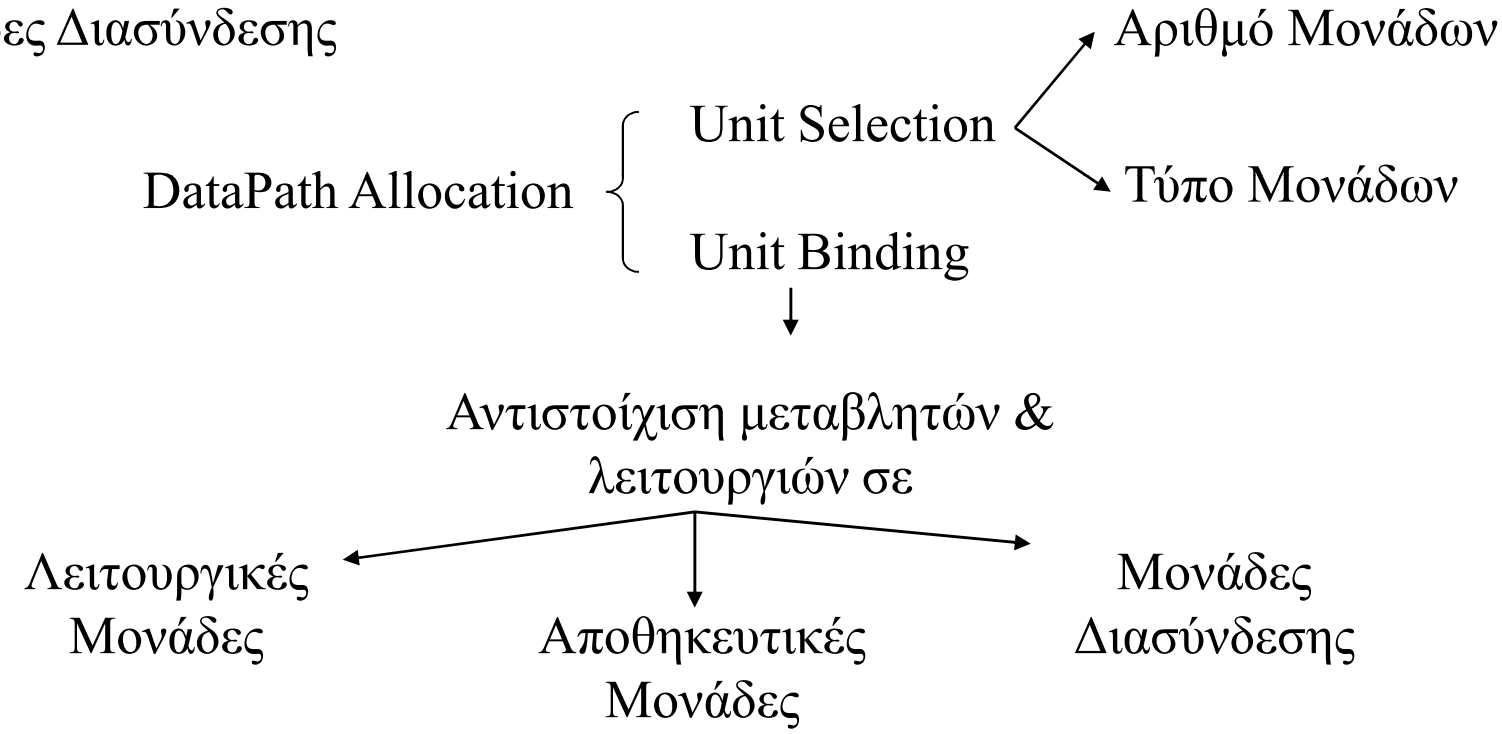
Σύνθεση Data Path

Το DataPath είναι ένα netlist που αποτελείται από τρεις τύπους μονάδων:

(α) Λειτουργικές Μονάδες,

(β) Μονάδες Αποθήκευσης και

(γ) Μονάδες Διασύνδεσης



Σύνθεση Data Path

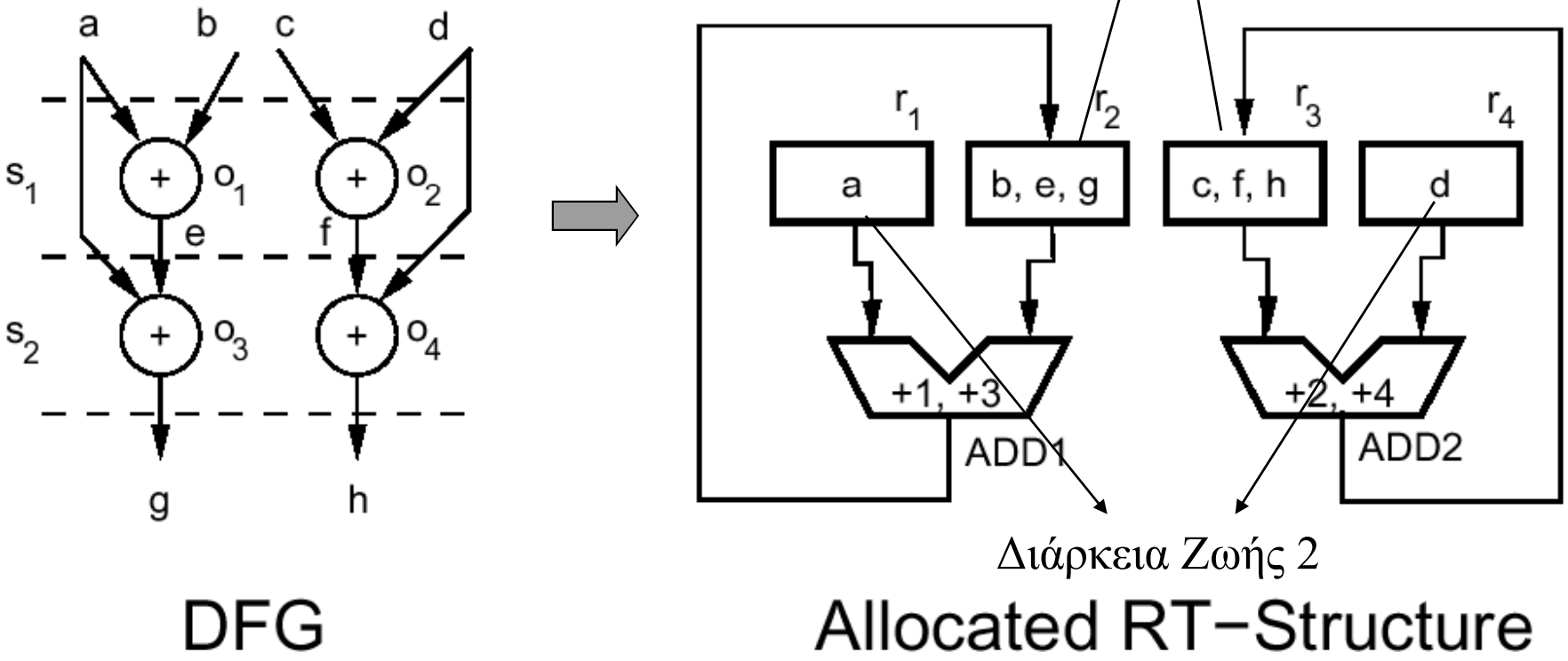
- ✓ Για κάθε λειτουργία απαιτείται η αντίστοιχη λειτουργική μονάδα.
- ✓ Για κάθε μεταβλητή που χρησιμοποιείται σε πολλά βήματα απαιτείται αποθηκευτική μονάδα (για τον χρόνο ζωής της).
- ✓ Για κάθε μεταφορά δεδομένων απαιτείται η αντίστοιχη μονάδα διασύνδεσης.

Περιορισμοί:

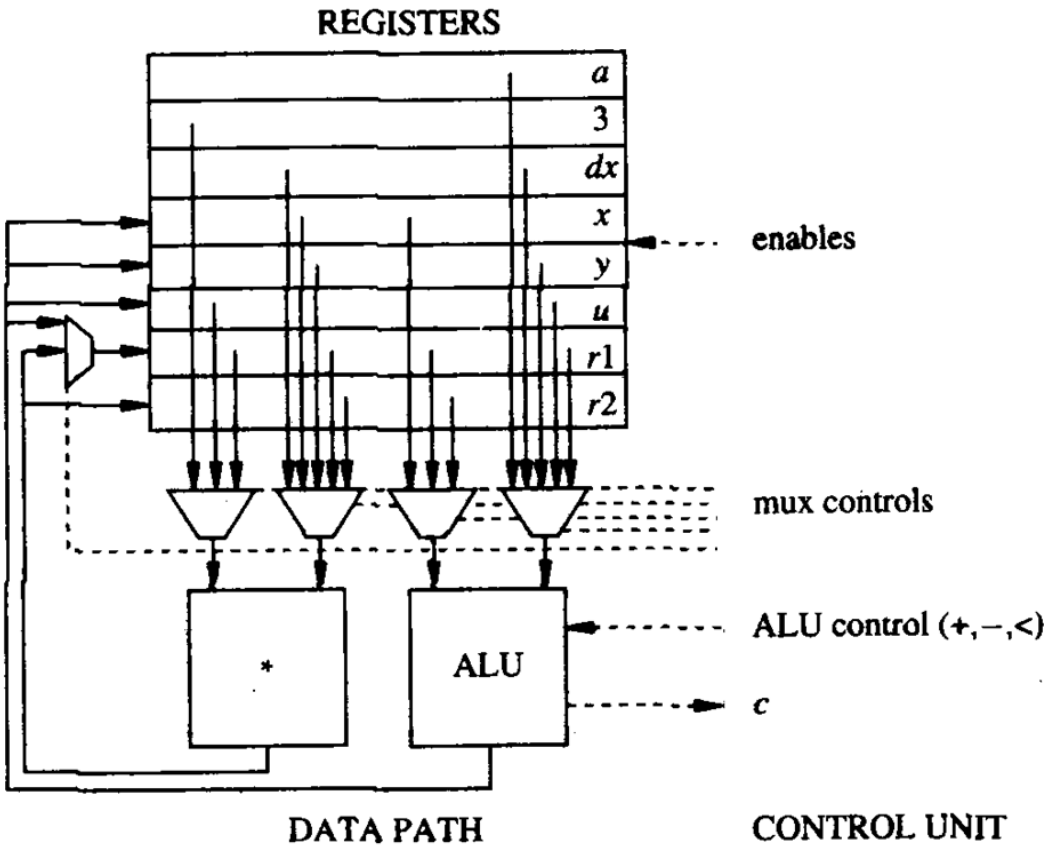
- ➢ Κάθε λειτουργική μονάδα μπορεί να εκτελέσει μόνο μία λειτουργία σε κάθε βήμα ελέγχου.
- ➢ Μπορούμε να έχουμε πολλαπλή πρόσβαση σε αποθηκευτικές μονάδες ανάλογα με τον αριθμό των θυρών.

Σύνθεση Data Path

Πρέπει να τηρούνται και οι περιορισμοί



Σύνθεση Data Path



xl=x + dx

ul=u-3*x*u*dx-3*y*dx

yl=y+u*dx

c=xl<a

x=xl

u=ul

y=yl

Σύνθεση Data Path

- ✓Η σύνθεση για Data Path καθορίζει και την διασύνδεση του Data Path με το περιβάλλον μέσω των θυρών διασύνδεσης.
- ✓Το Data Path μπορεί να συμπεριλαμβάνει μνήμες πολλαπλών θυρών (register file).
- ✓Η βελτιστοποίηση της χρήσης τέτοιων μνημών έγκειται στην βελτιστοποίηση των μεταφορών δεδομένων από και προς τις θύρες.
- ✓Η διασύνδεση με την λογική ελέγχου γίνεται μέσω των γραμμών ελέγχου.
- ✓Οι ακολουθιακές resources απαιτούν ένα σήμα start (και ίσως ένα reset). Έτσι απαιτείται ένα σύνολο σημάτων ενεργοποίησης (activation).

Σύνθεση Data Path

Μέθοδοι Επίλυσης

1. **Αλγόριθμοι Απληστίας (Greedy)**

   Κατασκευάζουν το DataPath σταδιακά ενώ διαπερνούν τον γράφο.

2. **Αλγόριθμοι Διάσπασης (Decomposition)**

   Διασπούν το πρόβλημα Allocation σε υποπροβλήματα τα οποία προσπαθούν να λύσουν χωριστά.

3. **Επαναληπτικές Μέθοδοι (Iterative Refinement)**

   Συνδυάζουν τις λύσεις υποπροβλημάτων

Αρχιτεκτονικές Data Path

Η τοπολογία διασύνδεσης είναι σημαντικός παράγοντας που επηρεάζει την απόδοση της αρχιτεκτονικής.

Παράδειγμα

Για την υλοποίηση των ακόλουθων μετακινήσεων

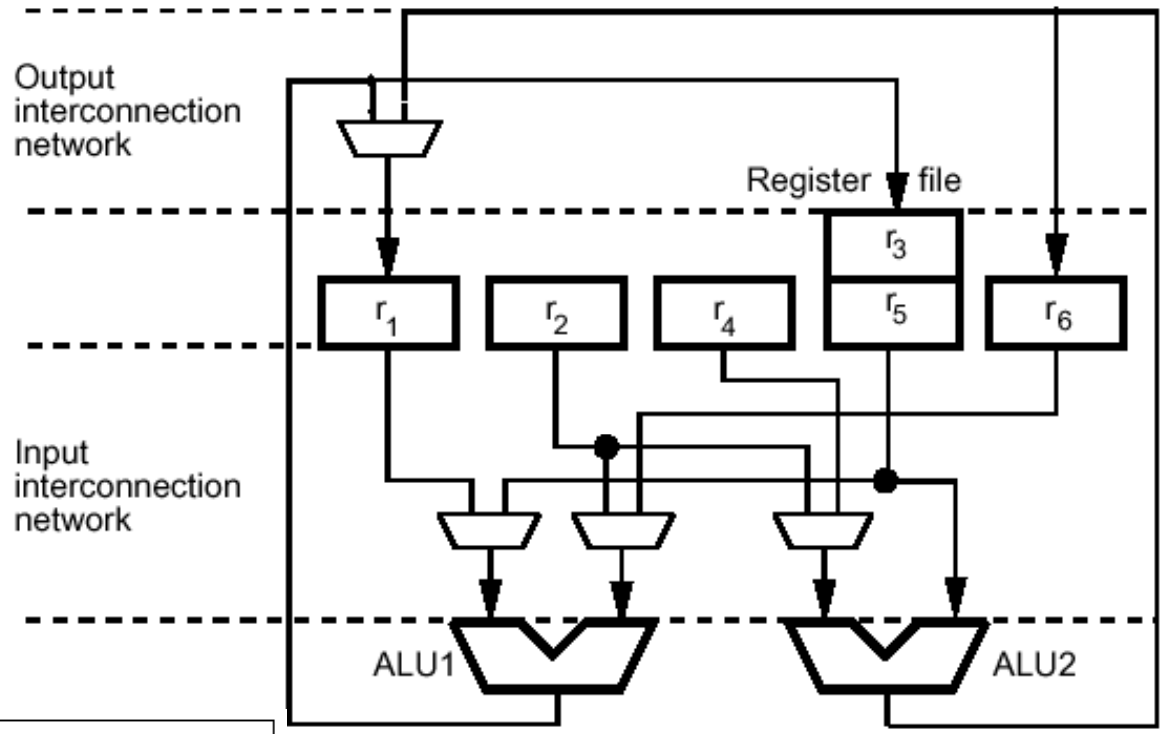
```
s1:  r3 <= ALU1(r1,r2);   r1 <= ALU2(r3,r4);
s2:  r1 <= ALU1(r5,r6);   r6 <= ALU2(r2,r5);
s3:  r3 <= ALU1(r1,r6)
```

μπορεί να χρησιμοποιηθεί:

(α) αρχιτεκτονική πολυπλεκτών ή

(β) αρχιτεκτονική διασύνδεσης διαύλων

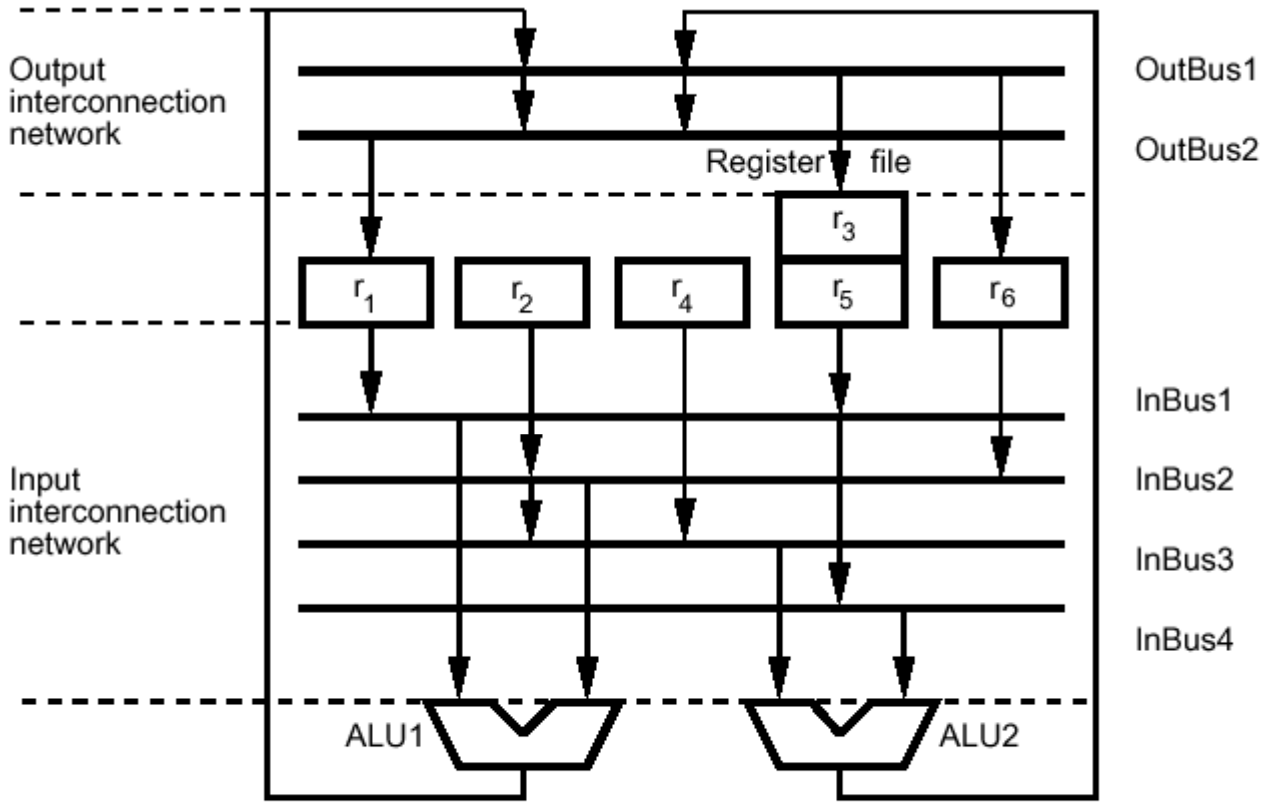
Αρχιτεκτονική Πολυπλεκτών



s1: r3 <= ALU1(r1,r2); r1 <= ALU2(r3,r4);
s2: r1 <= ALU1(r5,r6); r6 <= ALU2(r2,r5);
s3: r3 <= ALU1(r1,r6)

Mux−oriented DP

Αρχιτεκτονική Διαύλων

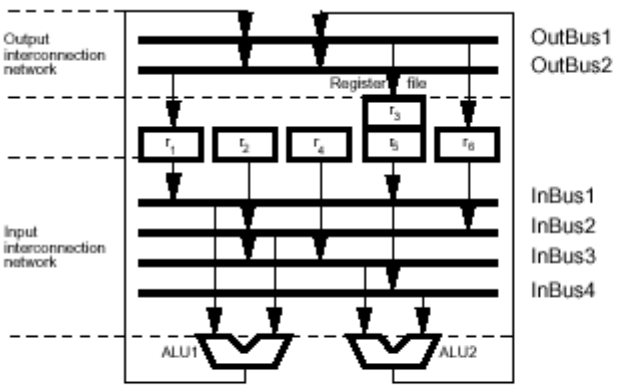


Bus−oriented DP

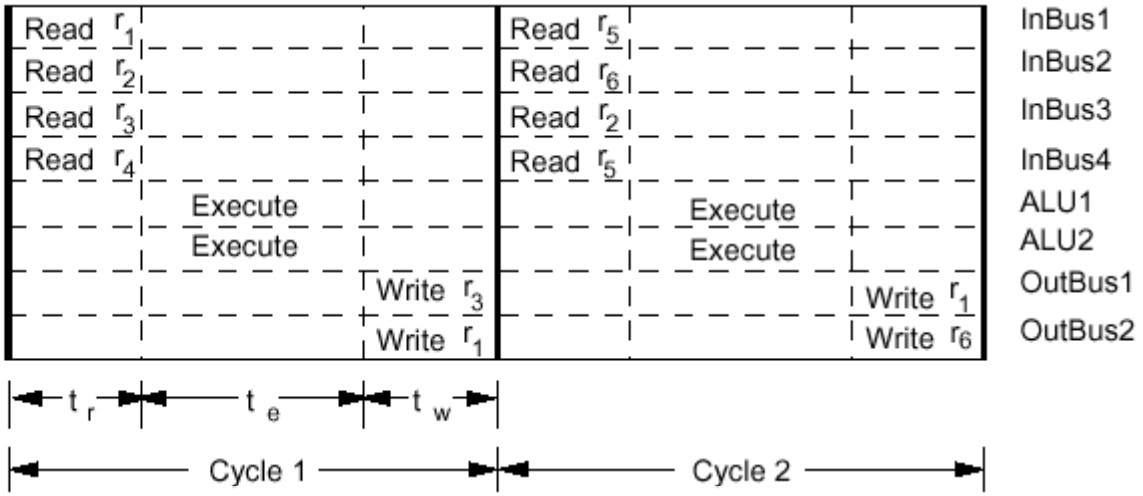
Αρχιτεκτονικές Data Path

- ✓ Μία τοπολογία ονομάζεται **Point-To-Point** αν υπάρχει μία μονάδα διασύνδεσης ανάμεσα σε κάθε δύο θύρες που διασυνδέονται.
- ✓ Για περισσότερες από μία διασυνδέσεις χρησιμοποιούνται δίαυλοι.
- ✓ Για ελαχιστοποίηση των διασυνδέσεων μπορούμε να συνδυάσουμε καταχωρητές σε αρχεία καταχωρητών με πολλαπλές θύρες.
- ✓ Τα δίκτυα διασύνδεσης διαχωρίζονται σε δίκτυα εισόδου (στις λειτουργικές μονάδες) και δίκτυα εξόδου (από τις λειτουργικές μονάδες).
- ✓ Τα δίκτυα εισόδου/εξόδου δεν είναι απαραίτητα το ίδιο πολύπλοκα.

Χρονισμός



Bus−oriented DP



- *Requirement:* Cycle Time > tr + te + tw
- *Sequential execution*

Η ανάγνωση-εκτέλεση-εγγραφή δεν μπορούν να επικαλυφθούν λόγω των κοινών μέσων.

```
s1:  r3 <= ALU1(r1,r2);   r1 <= ALU2(r3,r4);
s2:  r1 <= ALU1(r5,r6);   r6 <= ALU2(r2,r5);
s3:  r3 <= ALU1(r1,r6)
```

Τοποθέτηση Latches

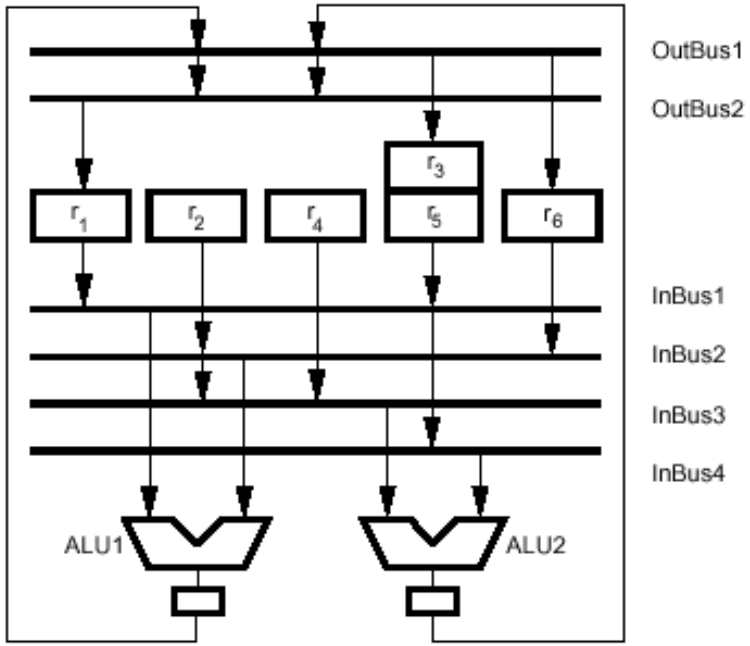
- ✓ Για την βελτίωση της απόδοσης των Data Path τοποθετούμε latches στις θύρες εισόδου/εξόδου.
- ✓ Όταν τοποθετούμε latches μόνο στις εξόδους των λειτουργικών μονάδων έχουμε επικάλυψη των ακόλουθων εργασιών:

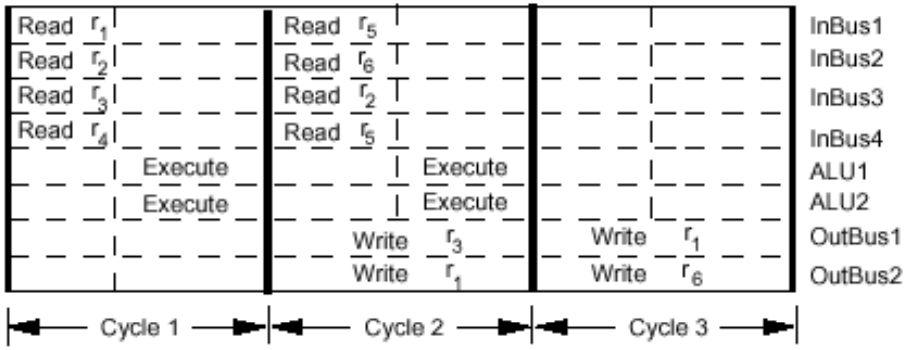
  1. Εγγραφή βήματος n-1

  2. Ανάγνωση και εκτέλεση λειτουργίας βήματος n.

- ✓ Ο κύκλος ρολογιού μειώνεται
- ✓ Δεν υπάρχει εξισορρόπηση των μετακινήσεων μεταξύ των καταχωρητών.
- ✓ Τα ίδια ισχύουν και όταν τοποθετούμε latches μόνο στις εισόδους των λειτουργικών μονάδων.

One Phase Pipelined DataPath





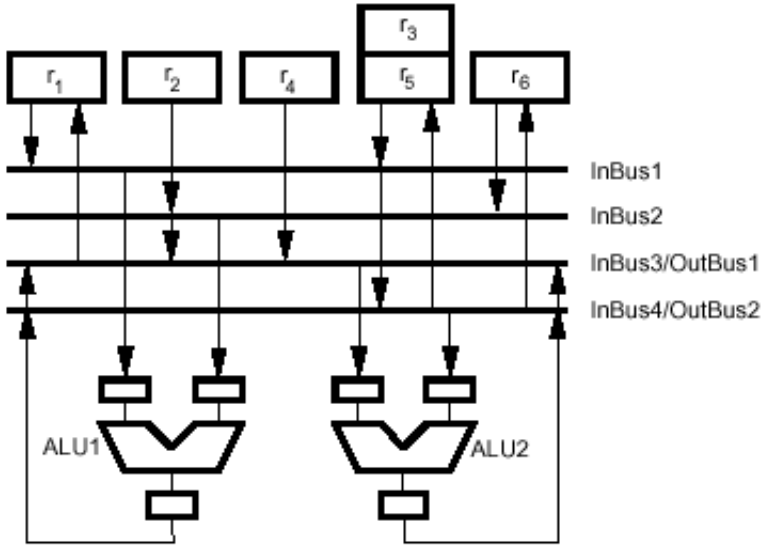
Η αποθήκευση των αποτελεσμάτων επιτρέπει την επικάλυψη εγγραφής με ανάγνωση-εκτέλεση.

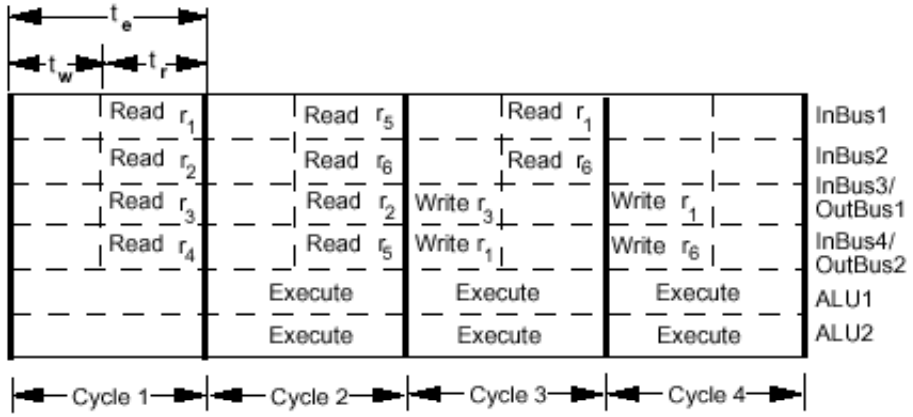
- Clock cycle = max (tr + te, tw)
- *Overlapping read and write data transfers*

Fully Pipelined DataPath

- ✓ Η μέγιστη επικάλυψη επιτυγχάνεται όταν έχουμε latches και στις εισόδους και τις εξόδους
- ✓ Το pipelining τότε έχει τρεις βαθμίδες (input, functional unit, output).
- ✓ Απαιτείται ένα διφασικό μη επικαλυπτόμενο ρολόι.
- ✓ Η μία φάση χρησιμοποιείται για τον έλεγχο των latches εισόδου/εξόδου.
- ✓ Η άλλη φάση χρησιμοποιείται για την εγγραφή στο αρχείο καταχωρητών.
- ✓ Ο κύκλος ρολογιού μειώνεται αισθητά.

Two Phase Pipelined DataPath

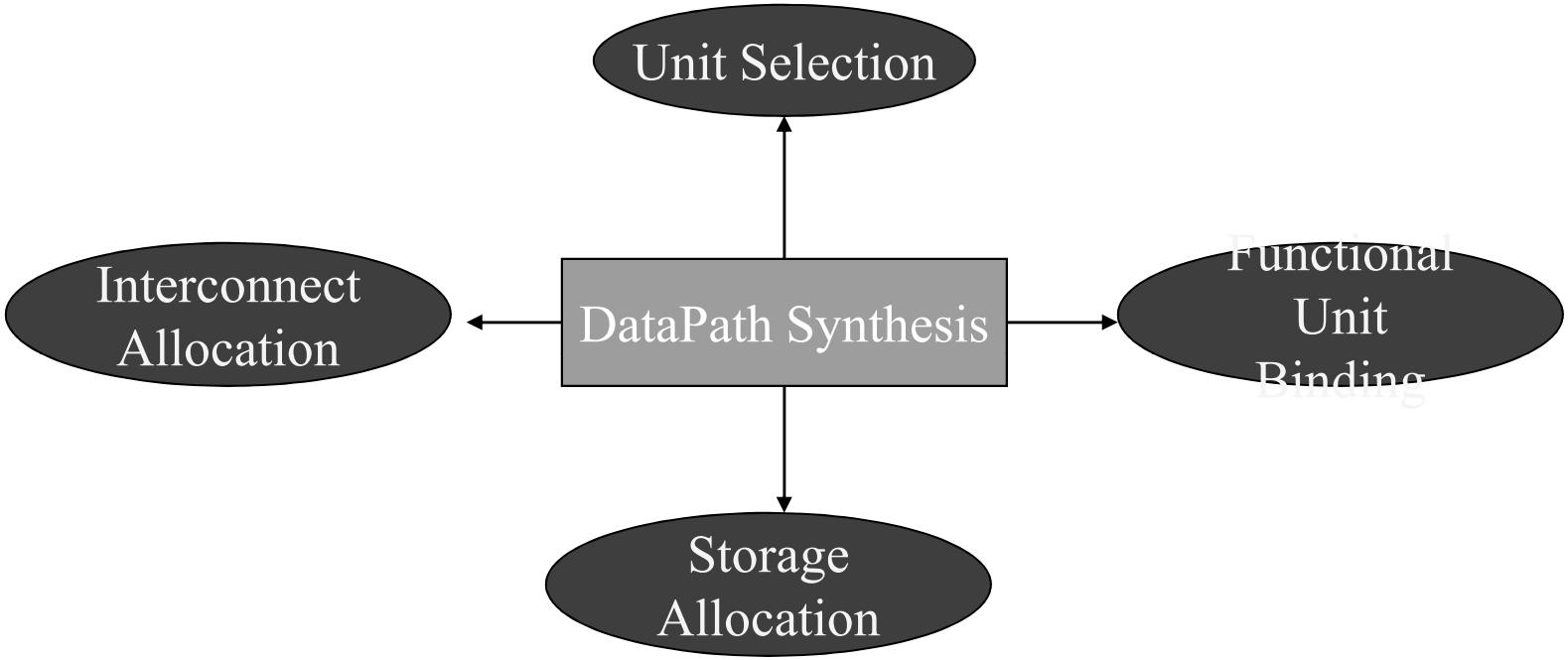




Η αποθήκευση δεδομένων και αποτελεσμάτων επιτρέπει την επικάλυψη εκτέλεσης με ανάγνωση-εγγραφή.

- Clock cycle = max (te, tr + tw)
- *Overlapping data transfers with FU execution*

Allocation Tasks



Unit Selection

- ✓ Μία βιβλιοθήκη περιέχει πολλαπλούς τύπους μονάδων με διαφορετικά χαρακτηριστικά.
- ✓ Πολλές μονάδες έχουν πολλαπλές χρήσεις.
- ✓ Πρέπει οι μονάδες που θα επιλεγούν να καλύπτουν τις απαιτούμενες λειτουργίες.
- ✓ Συχνά η επιλογή μονάδων συνδυάζεται με την δέσμευση ανά λειτουργία (binding).

Unit Binding

Functional

- ✓ Αντιστοίχιση των λειτουργιών με τις μονάδες που έχουν επιλεγεί.
- ✓ Συνήθως έχουμε λίγες μονάδες για πολλές λειτουργίες, οπότε χρειάζεται αλγόριθμος επιλογής.

Storage

- ✓ Αντιστοιχεί δεδομένα σε αποθηκευτικές μονάδες.
- ✓ Οι σταθερές αποθηκεύονται σε ROM.
- ✓ Οι μεταβλητές αντιστοιχίζονται σε καταχωρητές ή μνήμες.
- ✓ Μεταβλητές με μη επικαλυπτόμενους χρόνους ζωής μπορούν να μοιραστούν καταχωρητές.
- ✓ Οι καταχωρητές τελικά μπορούν να σχηματίσουν ένα αρχείο καταχωρητών.

Unit Binding

Interconnection

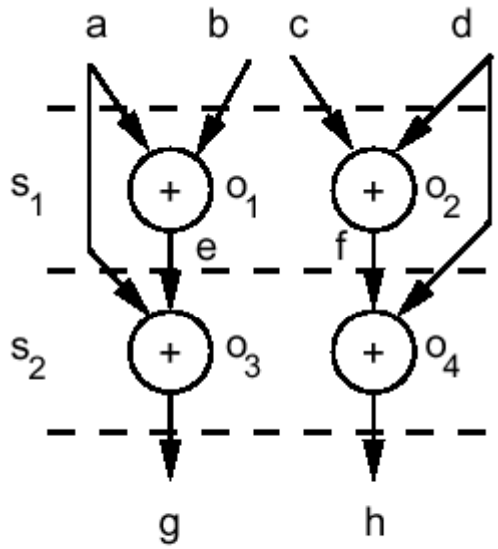
- ✓ Κάθε μεταφορά δεδομένων απαιτεί μονοπάτι διασύνδεσης.
- ✓ Δύο διαφορετικές μεταφορές δεδομένων μπορούν να μοιραστούν το ίδιο ή τμήμα μονοπατιού διασύνδεσης, εάν δεν γίνονται ταυτόχρονα.
- ✓ Στόχος είναι η μεγιστοποίηση της διαμοίρασης μονάδων διασύνδεσης (και άρα ελαχιστοποίηση του κόστους)

Εξάρτηση και σειρά

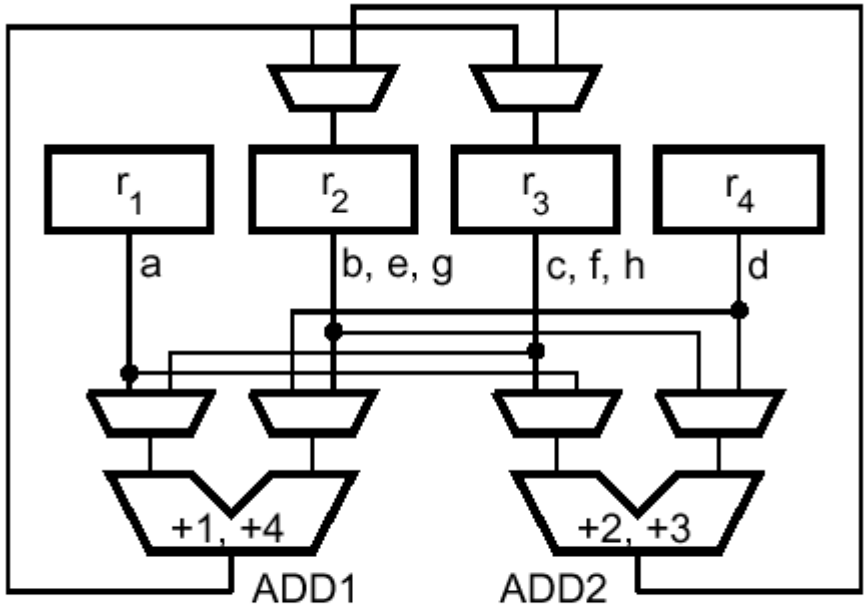
- ✓ Οι παραπάνω λειτουργίες επηρεάζουν η μία την άλλη.
- ✓ Ακόμη και η σειρά που εκτελούνται είναι πολύ σημαντική

Unit Binding

Παράδειγμα



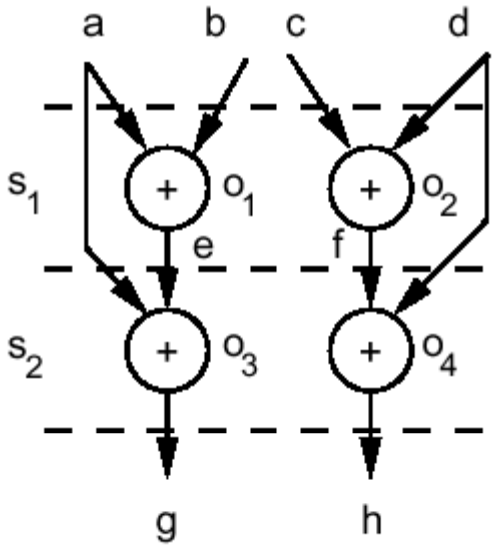
Scheduled DFG



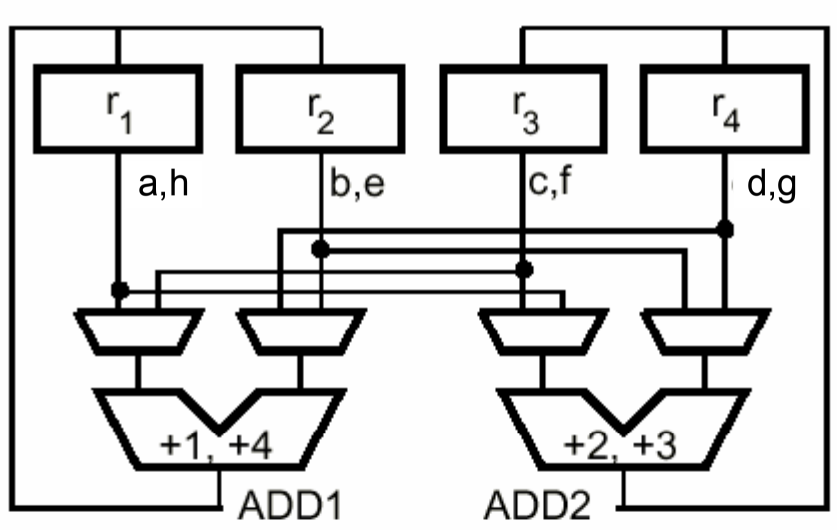
FU Binding with 6 Muxes

Για το παραπάνω Functional Unit/Storage Binding απαιτούνται 6 πολυπλέκτες

Unit Binding



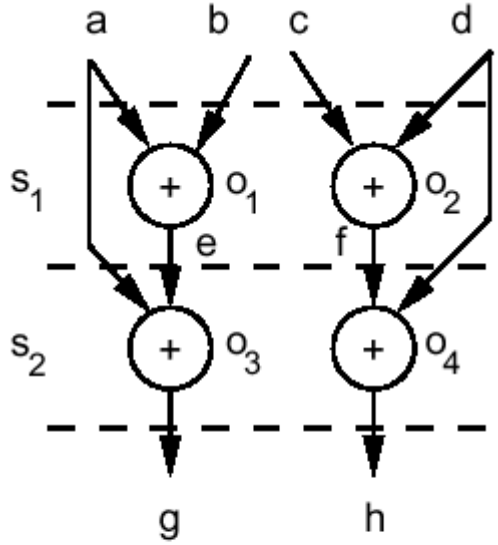
Scheduled DFG



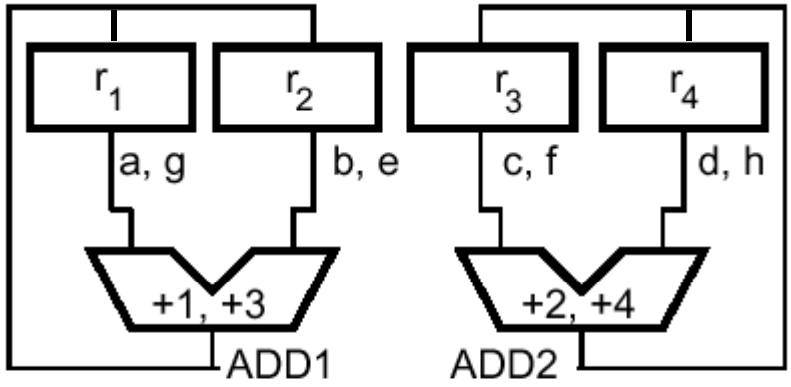
Register Reallocation: 4 Muxes

Με άλλο Storage Binding απαιτούνται 4 πολυπλέκτες. Ουσιαστικά κατανέμονται τα αποτελέσματα πράξεων σε όλους τους καταχωρητές και δεν απαιτούνται πολυπλέκτες εγγραφής.

Unit Binding



Scheduled DFG



FU Rebinding: Optimal Design

Αλλάζοντας και το Functional Binding δεν απαιτούνται πολυπλέκτες

Αλγόριθμοι Greedy

- ✓ Η ανάθεση των λειτουργιών σε components γίνεται βηματικά.
- ✓ Ξεκινά ο αλγόριθμος με κενό datapath και σταδιακά προσθέτει components.
- ✓ Σε κάθε βήμα προσπαθεί να βρει μία λειτουργική μονάδα η οποία είναι ελεύθερη την στιγμή που απαιτείται από την λειτουργία.
- ✓ Εάν υπάρχουν περισσότερες της μία μονάδες τότε επιλέγει αυτήν που ελαχιστοποιεί το κόστος διασύνδεσης.
- ✓ Εάν καμία μονάδα δεν είναι διαθέσιμη προσθέτουμε μία νέα.
- ✓ Η ίδια διαδικασία γίνεται και για τους καταχωρητές.
- ✓ Για τις μεταβλητές λαμβάνεται υπόψη ο χρόνος ζωής τους.
- ✓ Όταν υπάρχουν πολλοί διαθέσιμοι καταχωρητές, τότε επιλέγεται εκείνος που ελαχιστοποιεί το κόστος.

Αλγόριθμοι Διάσπασης (Decomposition)

- ✓ Προσπαθούν να δώσουν καλύτερα αποτελέσματα από τους αλγόριθμους απληστίας.
- ✓ Η διαδικασία Allocation διαιρείται σε επιμέρους τμήματα και κάθε ένα από αυτά επιλύεται χωριστά.
- ✓ Τα προβλήματα του allocation (storage/funct. unit/interconnection binding) επιλύονται χωριστά.
- ✓ Λόγω της συσχέτισης των προβλημάτων, δεν εγγυάται βέλτιστη λύση ακόμη και αν τα επιμέρους προβλήματα επιλυθούν βέλτιστα.

Μέθοδοι
- Διαμέριση Κλίκας
- Αλγόριθμος Αριστερής Ακμής
- Ταίριασμα Διμερή Γράφου με βάρη

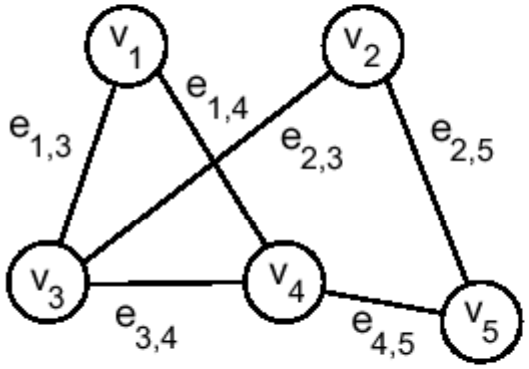
Διαμέριση Κλίκας

Στόχος η διαμέριση ενός γράφου σε ελάχιστο αριθμό από κλίκες

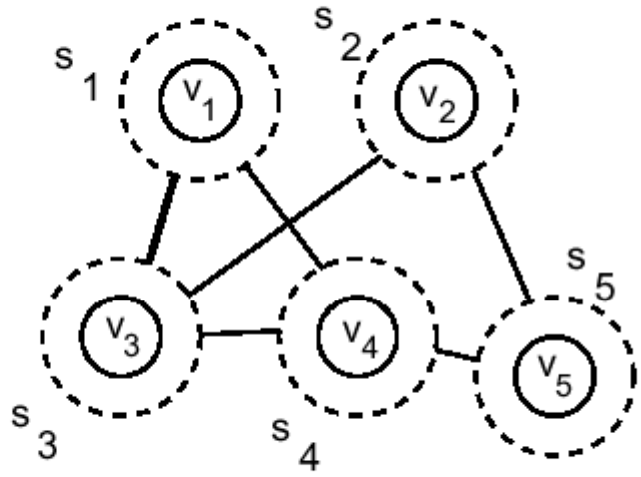
Επίλυση Προβλήματος Διαμέρισης σε κλίκες:

- ✓ Από τον αρχικό γράφο κατασκευάζουμε ένα υπεργράφο.
- ✓ Ο υπεργράφος αποτελείται από υπερ-κορυφές (σύνολα κορυφών).
- ✓ Μία υπερκορυφή είναι και κλίκα.
- ✓ Σε κάθε βήμα οι δύο υπερκορυφές που έχουν τους περισσότερους κοινούς γείτονες και συνδέονται με ακμή συνδυάζονται σε μία.
- ✓ Η νέα υπερκορυφή συνδέεται με ακμές μόνο με το σύνολο των κοινών γειτονικών άλλων υπερκορυφών.
- ✓ Η διαδικασία συνεχίζεται έως ότου να μην υπάρχουν άλλες ακμές.

Διαμέριση Κλίκας



Graph G



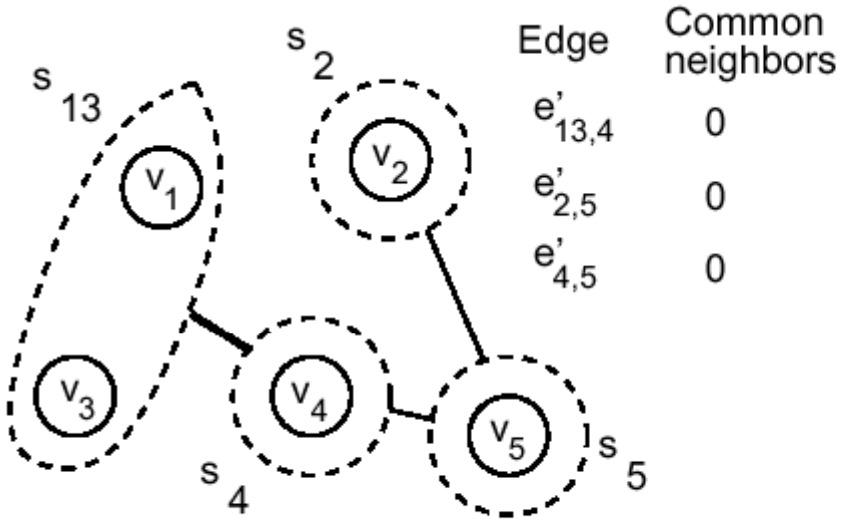
Edge	Common neighbors
$e'_{1,3}$	1
$e'_{1,4}$	1
$e'_{2,3}$	0
$e'_{2,5}$	0
$e'_{3,4}$	1
$e'_{4,5}$	0

Common Neighbors

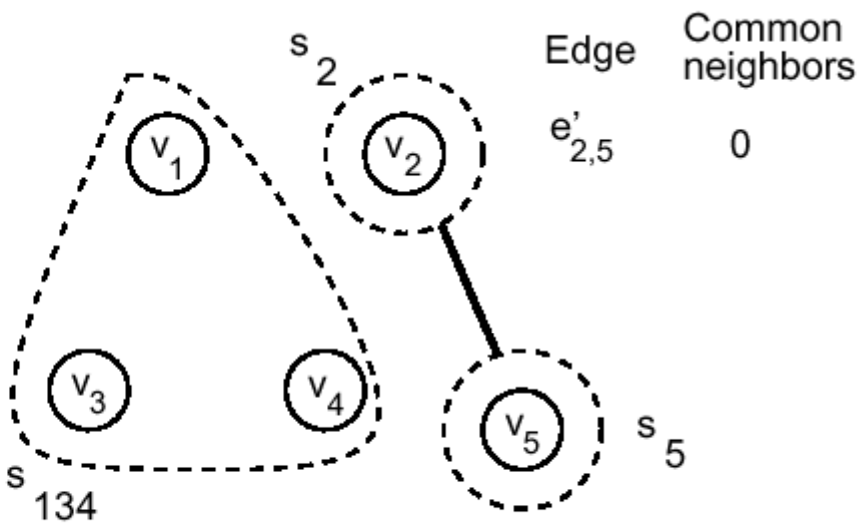
Αρχικός Γράφος

Μετατροπή σε υπεργράφο

Διαμέριση Κλίκας

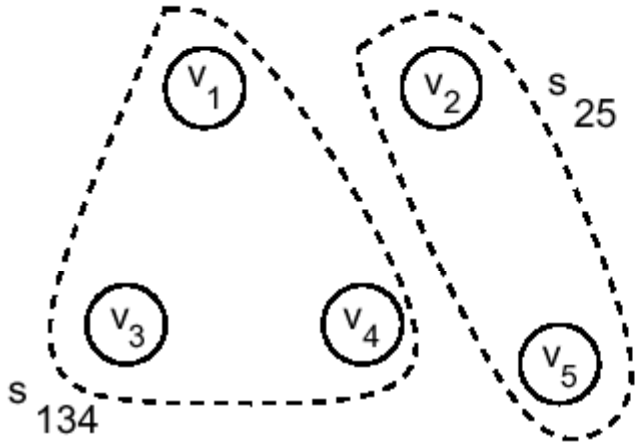


Supernode Creation 1



Supernode Creation 2

Διαμέριση Κλίκας



Supernode Creation 3

$s_{134} = \{v_1, v_3, v_4\}$

$s_{25} = \{v_2, v_5\}$

Cliques for Graph G

Διαμέριση Κλίκας

Χρήση Γράφων στο πρόβλημα Register Allocation:

- ✓ Στόχος του Register Allocation είναι η ελαχιστοποίηση του κόστους των καταχωρητών, με την μέγιστη δυνατή διαμοίραση.
- ✓ Κάθε κορυφή του γράφου αντιστοιχεί σε μία μεταβλητή
- ✓ Μία ακμή υπάρχει όταν δύο μεταβλητές μπορούν να αποθηκευτούν στον ίδιο καταχωρητή.
- ✓ Κάθε κλίκα αντιστοιχίζεται σε ένα καταχωρητή, καθώς όλες οι μεταβλητές μπορούν να συνυπάρξουν σε αυτόν.

Διαμέριση Κλίκας

Χρήση Γράφων στο πρόβλημα Functional Allocation:

- ✓ Κάθε κορυφή του γράφου αντιστοιχεί σε μία λειτουργία
- ✓ Μία ακμή υπάρχει όταν

  (α) Οι δύο λειτουργίες δεν εκτελούνται ταυτόχρονα.

  (β) Χρησιμοποιείται τύπος μονάδας που μπορεί να τις εκτελέσει και τις δύο.
- ✓ Κάθε κλίκα αντιστοιχίζεται σε μία λειτουργική μονάδα.

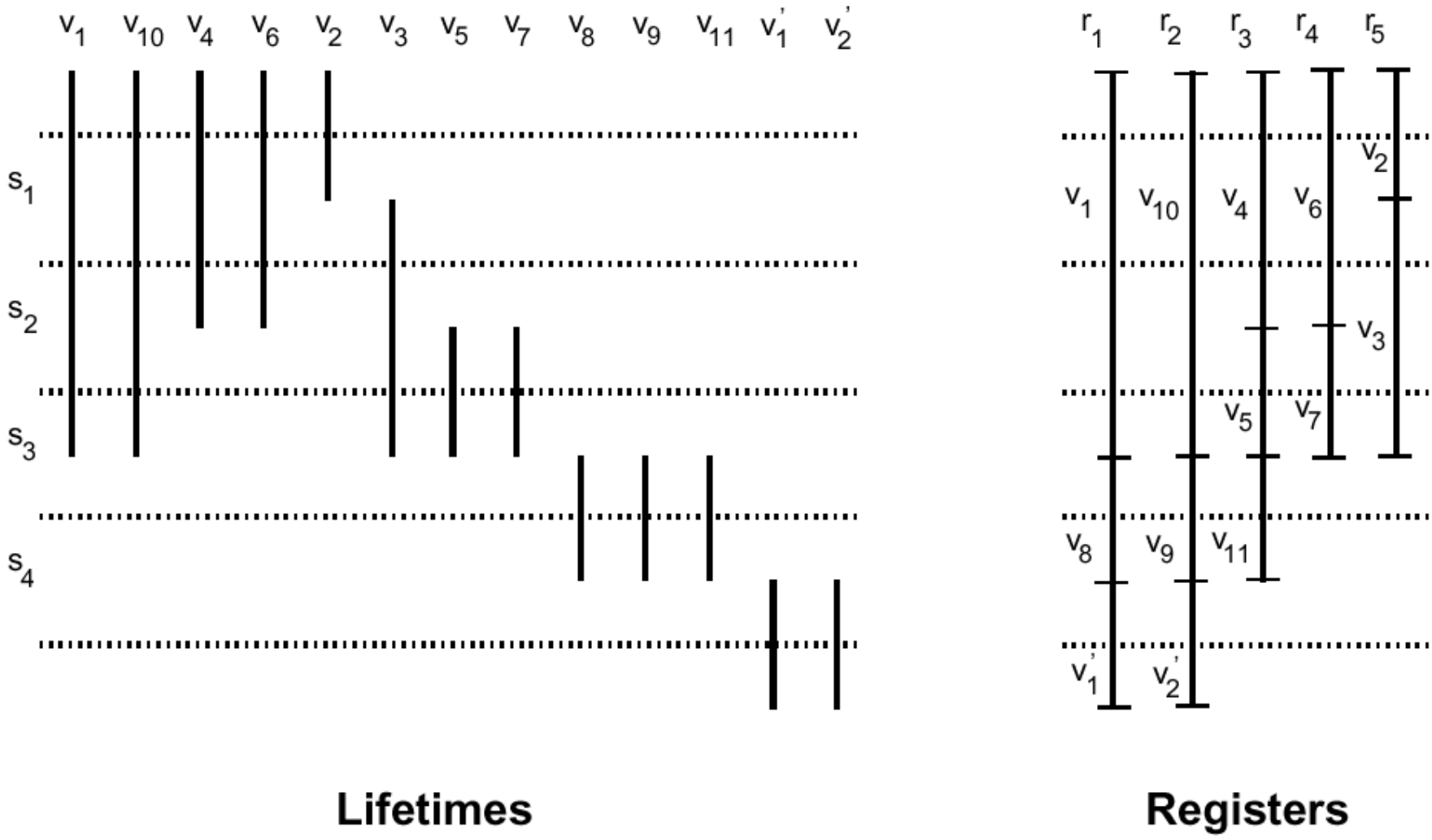
Χρήση Γράφων στο πρόβλημα Interconnection Allocation:

- ✓ Κάθε κορυφή του γράφου αντιστοιχεί στην διασύνδεση δύο μονάδων.
- ✓ Μία ακμή υπάρχει όταν οι διασυνδέσεις που αντιστοιχούν στις κορυφές δεν χρησιμοποιούνται ταυτόχρονα.
- ✓ Κάθε κλίκα αντιστοιχεί σε διαύλους ή πολυπλέκτες

Αλγόριθμος Αριστερής Ακμής

- ✓ Δέχεται σαν είσοδο μία λίστα μεταβλητών L.
- ✓ Για κάθε μεταβλητή υπάρχει και ένα διάστημα ζωής.
- ✓ Ο αλγόριθμος κάνει πολλαπλά περάσματα και αναθέτει μεταβλητές σε καταχωρητές.
- ✓ Σε κάθε πέρασμα εξαντλεί την χρήση ενός καταχωρητή με όσο το δυνατόν περισσότερες μεταβλητές.
- ✓ Ο αλγόριθμος έχει πολυωνυμική πολυπλοκότητα.
- ✓ Η λύση που δίνει είναι βέλτιστη.

Αλγόριθμος Αριστερής Ακμής



Weighted Bipartite-Matching Algorithm

- ✓ Αντιστοιχίζει πολλαπλές λειτουργίες/μεταβλητές σε πολλαπλές μονάδες ταυτόχρονα.
- ✓ Λαμβάνει υπόψη και το κόστος διασυνδέσεων.

Βήματα:

- ➢ Ταξινομούνται οι μεταβλητές σύμφωνα με τον χρόνο έναρξης τους.
- ➢ Διαιρούνται σε ομάδες με το ακόλουθο κριτήριο: Η ζωή κάθε μεταβλητής σε μία ομάδα επικαλύπτεται με **όλες** τις άλλες μεταβλητές της ομάδας.
- ➢ Αναθέτουμε τις μεταβλητές της μεγαλύτερης ομάδας σε ισάριθμους καταχωρητές.
- ➢ Κάθε επόμενη ομάδα ανατίθεται σε αυτούς τους καταχωρητές αν δεν υπάρχει επικάλυψη με τις μεταβλητές που έχουν ήδη ανατεθεί.

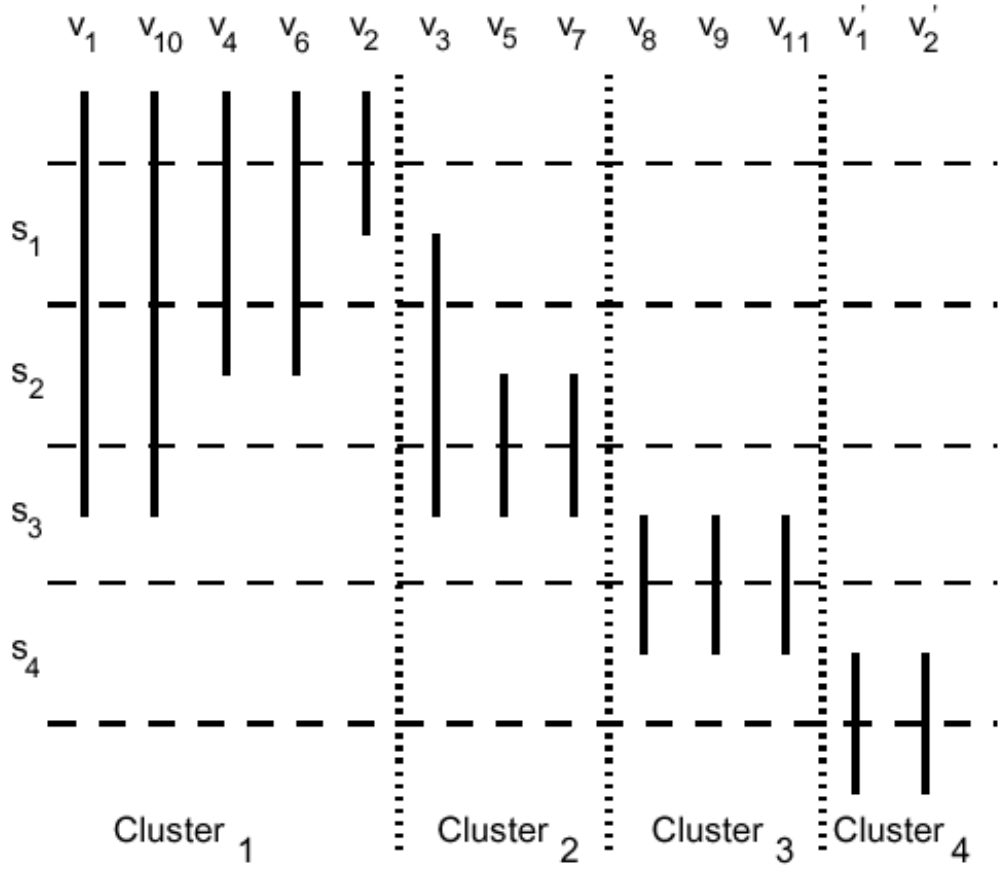
Weighted Bipartite-Matching Algorithm

- ✓ Σε κάθε βήμα δημιουργείται ένας διμερής γράφος, με

  (α) το σύνολο των καταχωρητών και

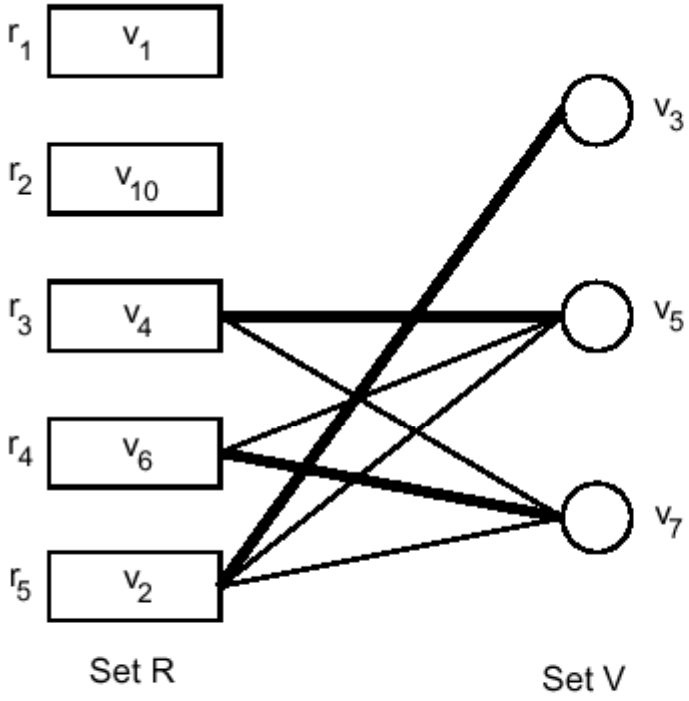
  (β) το σύνολο των μεταβλητών της επόμενης ομάδας.
- ✓ Κάθε ακμή του γράφου αναπαριστά πιθανή αντιστοίχηση μεταβλητής σε καταχωρητή.
- ✓ Οι ακμές μπορούν να έχουν βάρη.
- ✓ Μία ακμή έχει μεγάλο βάρος αν η αντίστοιχη ανάθεση έχει μειωμένο κόστος διασύνδεσης.

Weighted Bipartite-Matching Algorithm



Sorted Lifetime Intervals with Clusters

Weighted Bipartite-Matching Algorithm



Bipartite Graph for Binding Vars in Cluster2 after Cluster1

$r_1 = \{ v_1 , v_8 , v_1' \}$

$r_2 = \{ v_9 , v_{10} \}$

$r_3 = \{ v_4 , v_5 , v_{11} \}$

$r_4 = \{ v_6 , v_7 \}$

$r_5 = \{ v_2 , v_3 , v_2' \}$

Final Register Binding

Επαναληπτικές Μέθοδοι

- ✓ Ξεκινούν με ένα αρχικό DataPath το οποίο δημιουργήθηκε με κάποια κατασκευαστική μέθοδο ή μέθοδο διάσπασης.
- ✓ Προσπαθούν να μειώσουν το κόστος διασύνδεσης με απλή εναλλαγή των αναθέσεων μονάδων-λειτουργιών.
- ✓ Το ίδιο επιτυγχάνουν και με εναλλαγή αναθέσεων μεταβλητών-καταχωρητών.

Απλή Προσέγγιση: Εναλλαγή ζεύγους.

Δύο αναθέσεις εναλλάσσονται μεταξύ τους.

Επιλέγεται από όλα τα πιθανά ζεύγη πάντα το καλύτερο (greedy).

Στόχος η μείωση του κόστους του DataPath.

Η απόδοση του είναι φτωχή

Υπάρχουν πιο καλές προσεγγίσεις.