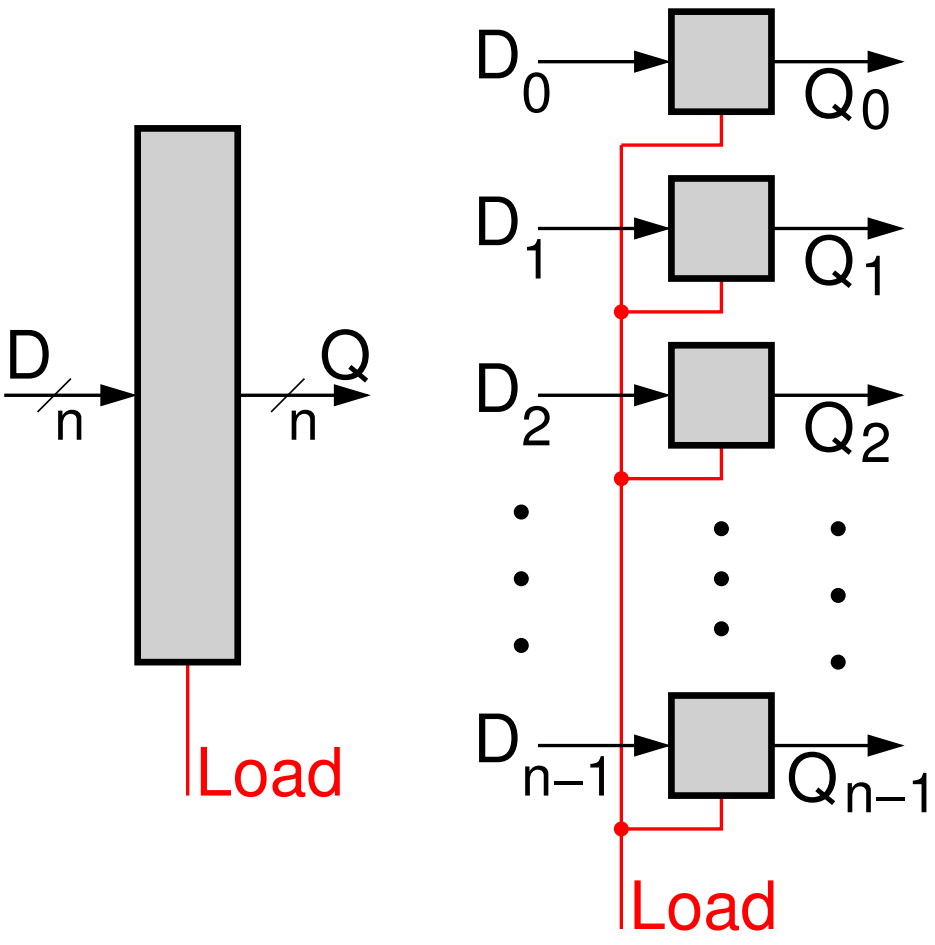
Καταχωρητές (Registers), Επαναχρησιμοποίηση Πόρων, Σήματα Χρονισμού

07b (§ 7.5 - 7.8) – 4 Νοε. 2026 – Μανόλης Κατεβαίνης

© copyright University of Crete – https://www.csd.uoc.gr/~hy120/26f/copyright.html

Καταχωρητές (Registers): πολύμπιτοι μανταλωτές

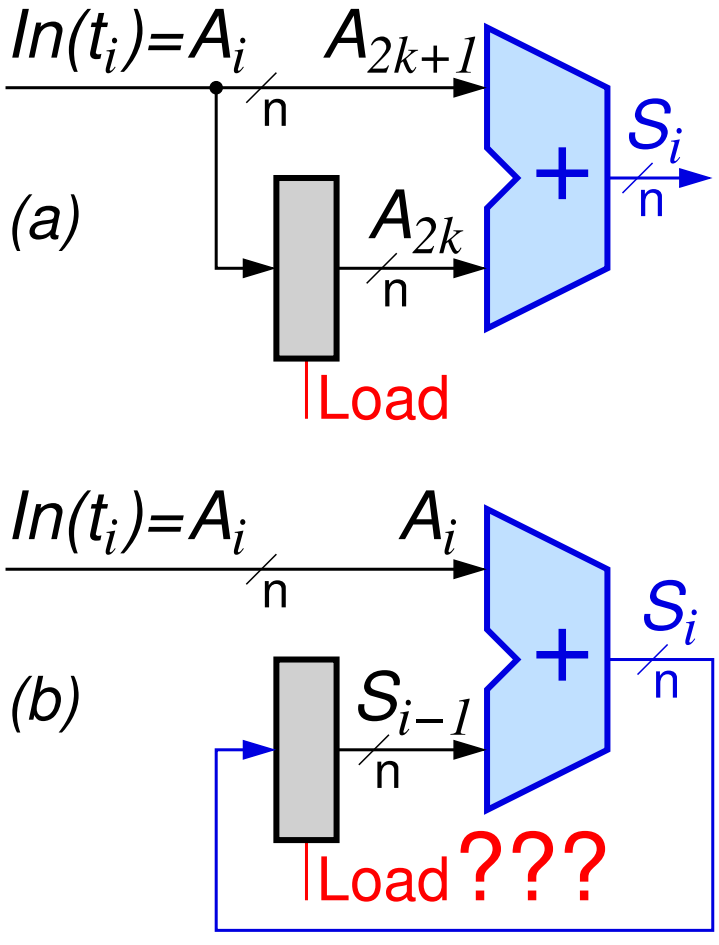


- Εγγραφή (αποθήκευση) μιάς ολόκληρης λέξης (*n* bits) όλης μαζί
- Κοινό σήμα ελέγχου φόρτωσης (*Load*) και γιά τους *n* μανταλωτές
- «Καταχωρητής»: πολύμπιτη λέξη από flip-flops
  - υπάρχουν και άλλων μορφών («ακμοπυροδότητα») flip-flops & καταχ.

Επαναχρησιμοποίηση Πόρων (resource reuse)

- Δισεκατομμύρια πράξεις το δευτερόλεπτο – όλες στην ίδια αριθμητική μονάδα, επαναχρησιμοποιούμενη
- Δισεκατομμύρια λέξεις το δευτερόλεπτο – όλες από τα ίδια σύρματα της ίδιας μνήμης, επαναχρησιμοποιούμενης
- Δισεκατομμύρια bits το δευτερόλεπτο – όλα από το ίδιο σύρμα δικτύου, επαναχρησιμοποιούμενο κάθε κλάσμα ns
- Οι επόμενες λειτουργίες εξαρτώνται από τα προηγούμενα αποτελέσματα, και τα χρησιμοποιούν:
  - Πρέπει να κρατηθούν κάπου τα προηγούμενα:
  - Καταχωρητές, Ακολουθιακά Κυκλώματα/συστήματα!

Ακολουθιακή Επεξεργασία Αριθμών – δύο παραδείγματα



- Είσοδος *In(t)*: ροή (ακολουθία) δεδομένων (λέξεων - αριθμών)
- Σε χρόνο t_i: ο νέος αριθμός A_i
- Δύο παραδείγματα/περιπτώσεις:

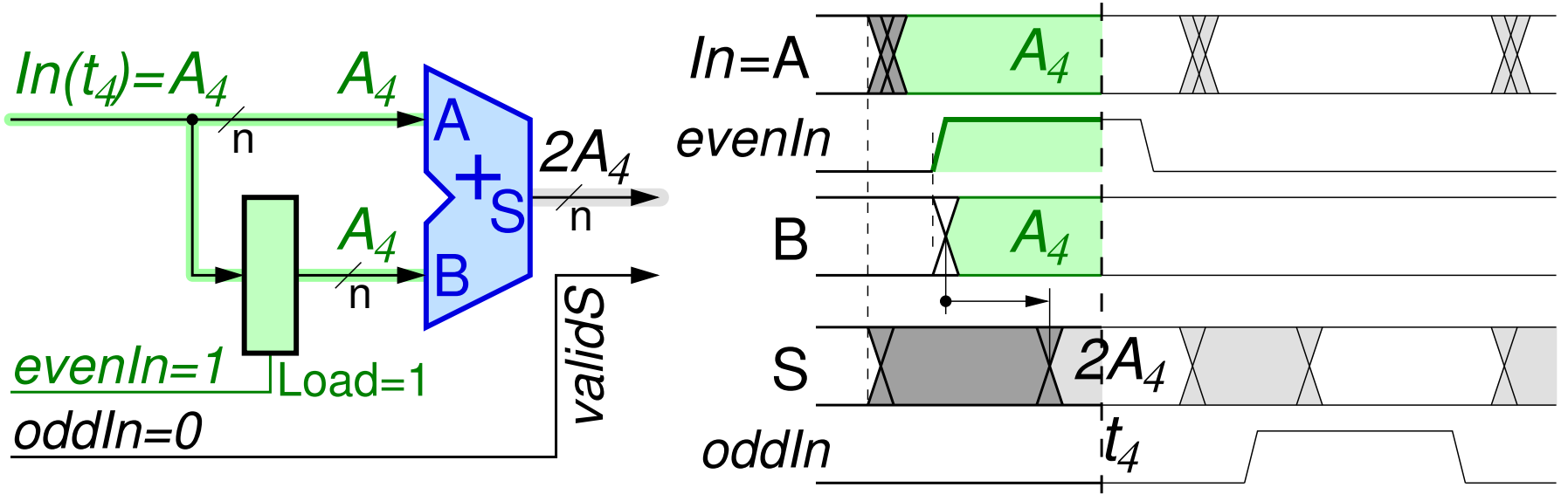
  a) Αθροίσματα ανά δύο:

  $$A_{2k}+A_{2k+1}$$

  b) Άθροισμα όλων των A_k από

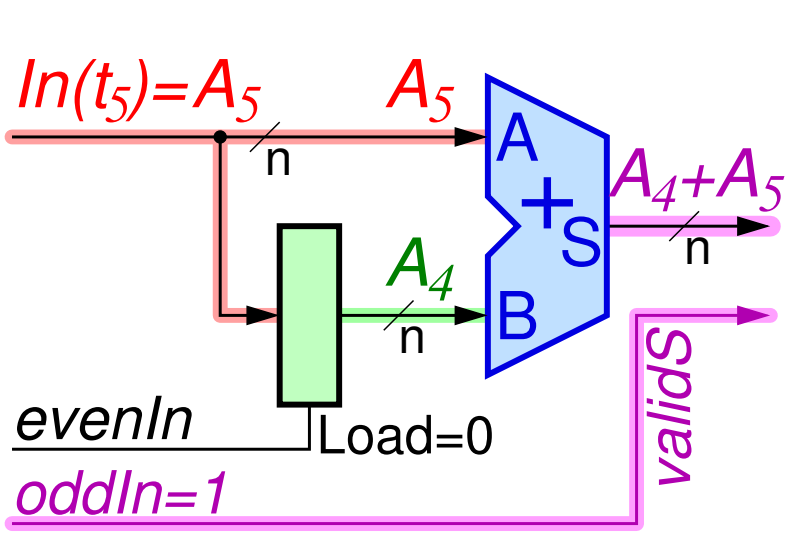
  $k=0$ έως και το τρέχον i @t_i

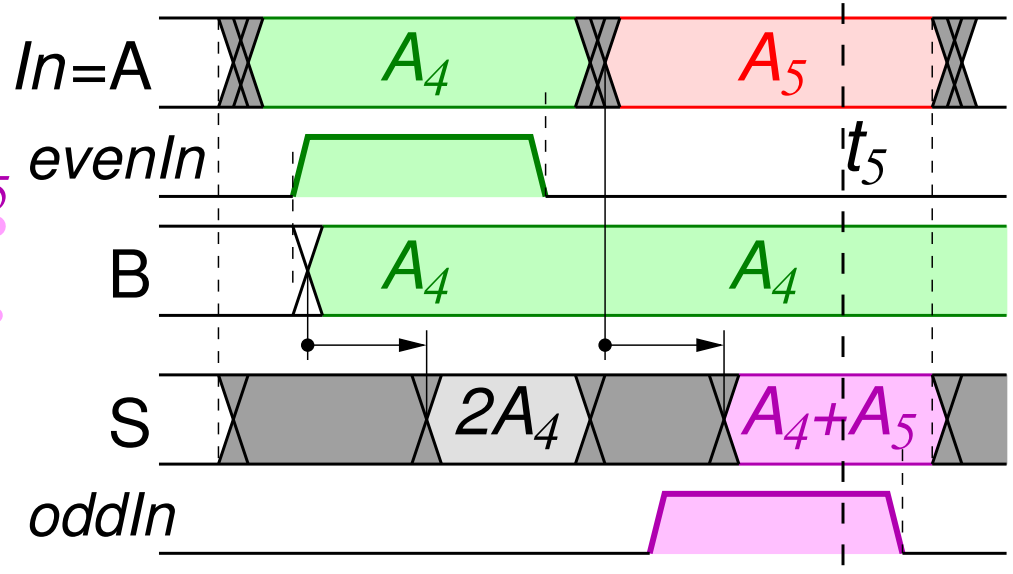
Άθροισμα ζεύγους: μέρος πρώτο



- Αποθηκεύουμε τα A_0, A_2, A_4,… γιά να τα έχουμε μετά
- Σήμα χρονισμού *evenIn*: ανάβει όταν A_0, A_2, A_4,… έγκυρα
- Εδώ, ακόμα, η έξοδος του αθροιστή μας είναι άχρηστη

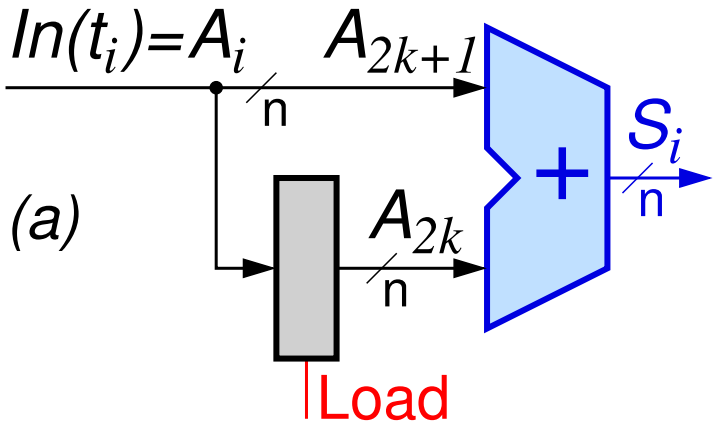
Άθροισμα ζεύγους: μέρος δεύτερο

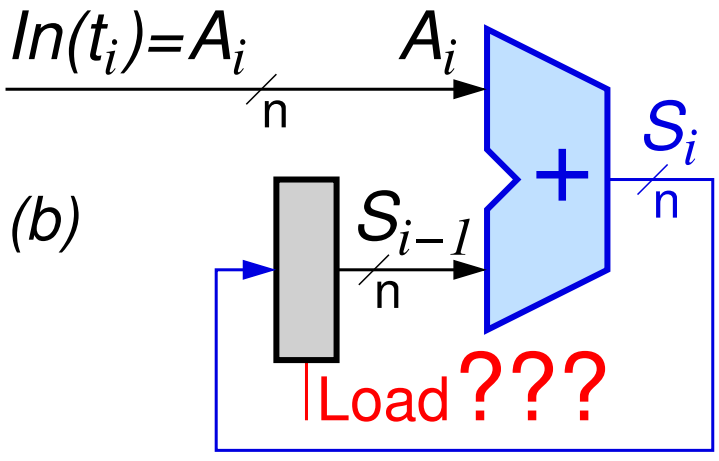




- Έχουμε το A_4 από πριν: το κρατήσαμε όταν έσβησε το Load
- Ο αθροιστής προσθέτει ό,τι βλέπει: A_5 τώρα + A_4 από πριν
- Σήμα χρονισμού *validS=oddIn*: δηλώνει έγκυρο άθροισμα

Μανταλωτές σε βρόχο: Πρόβλημα Χρονισμού





a) Ο καταχωρητής/μανταλωτής δεν είναι σε βρόχο: έξοδός του δεν επηρεάζει την είσοδό του

b) Άθροισμα όλων των A_k από $k=0$ έως και το τρέχον i @t_i

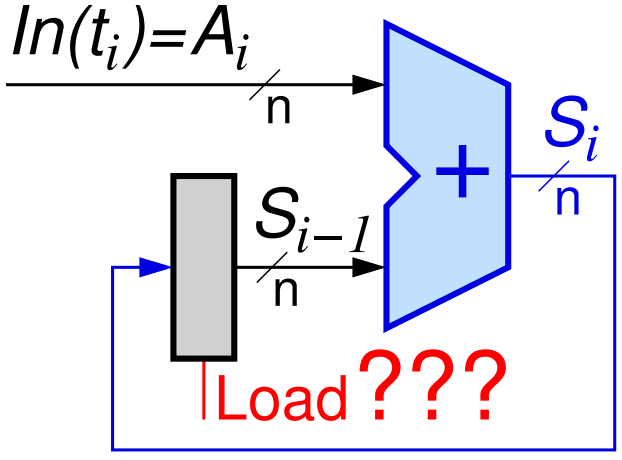
→ Μανταλωτής σε βρόχο!

→ Είσοδός του = *f(* έξοδός του *)*

→ Όταν ανάψει το Load, το περιεχόμενό του *αυτοκαταστρέφεται* από την ανάδραση!!

Πόση ώρα αναμένο το Load, αν μπορούσαμε;

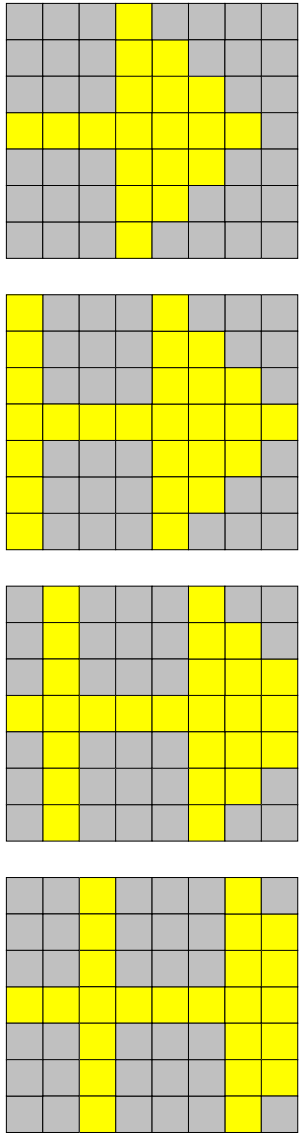
- Θα θέλαμε Load=1 τόσο ώστε:
  - εγγραφή μανταλωτή με σιγουριά
  - αλλά να *μην* προλάβουν τα σήματα να «κάνουν το γύρο» του βρόχου

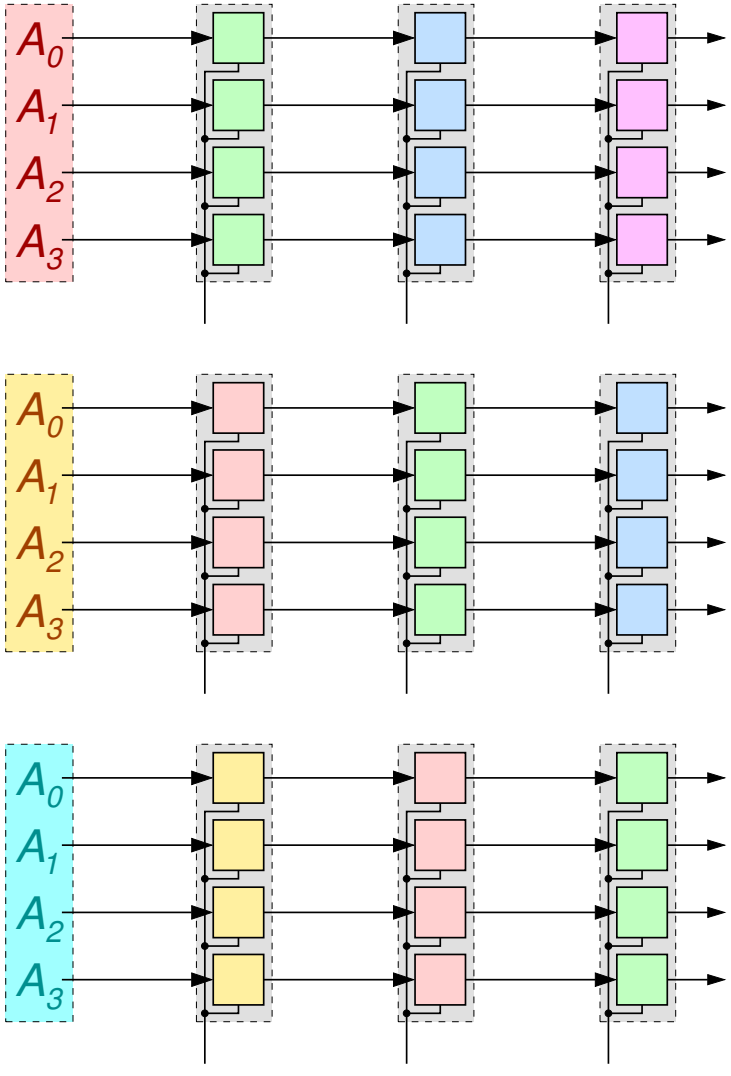


- Δύσκολο να μην «κάνουν το γύρο»:
  - ορισμένοι δρόμοι σημάτων πολύ συντομότεροι από άλλους
  - η λογική του βρόχου μπορεί και «μηδενική» – επόμενο π.χ.
- Δύσκολο να φτιαχτούν παλμοί τόσο «στενοί» (2-4 καθ. πυλών)
- Καθυστερήσεις πυλών αλλάζουν πολύ με θερμοκρασία, τάση τροφοδοσίας, κατασκευαστική μεταβλητότητα τρανζίστορς, κλπ.

Παράδειγμα: Ολισθαίνουσα επιγραφή

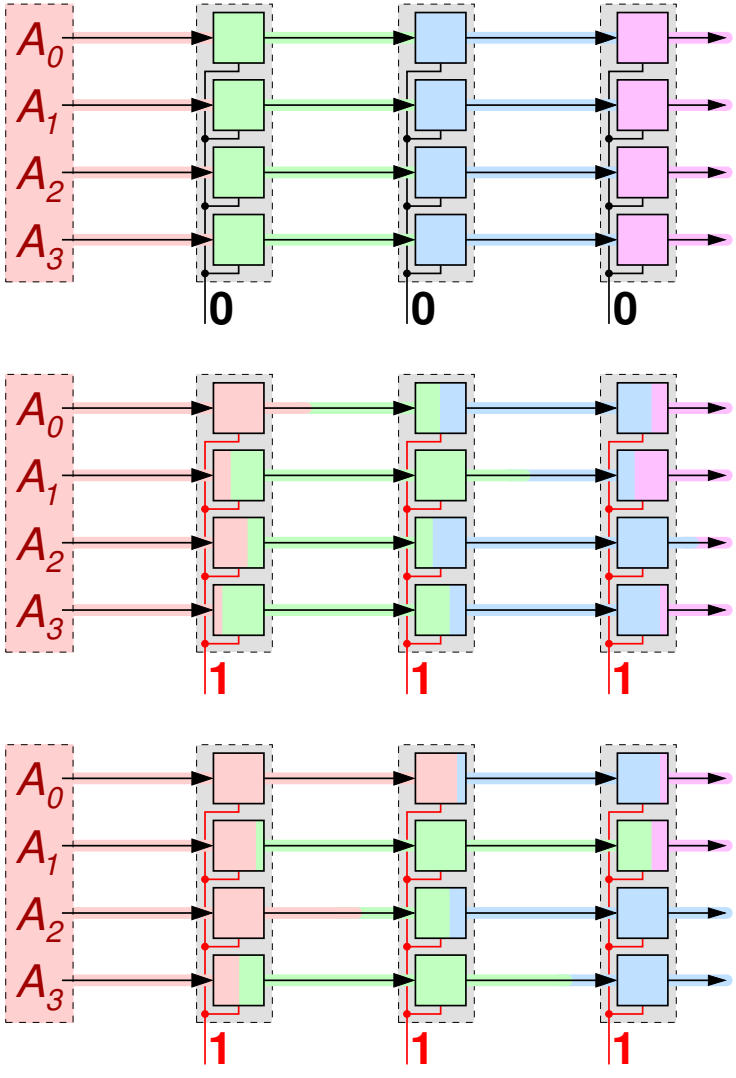
- Διδιάστατος πίνακας από LED
- Κάθε LED τροφοδ. από έναν μανταλωτή
- Κάθε στήλη τροφοδ. από έναν καταχωρητή
- Στην κάθε ολίσθηση, ο κάθε καταχωρητής τροφοδ. από το διπλανό του αριστερά
- Αλλά και ο διπλανός από τον διπλανό του
- Πώς θα καταφέρουμε μόνον ένα «βήμα» ο κάθε καταχωρητής την κάθε φορά;
- «Μηδενική» λογική μεταξύ καταχωρητών





Τι θέλουμε να πετύχουμε

- Σε κάθε «κύκλο ολίσθησης», τα bits της κάθε στήλης να μετακινηθούν *ακριβώς μία* στήλη δεξιά
- Σημείωση: τα bits δεν έχουν «χρώμα» – το χρώμα το βάλαμε γιά εμάς τους ανθρώπους, γιά να βλέπουμε ποιό bit πήγε πού



Τι θα πάθουμε

- Με μανταλωτές στη σειρά, όπου ανάβουν μαζί τα Load όλων, τα Data προωθούνται με ανεξέλεγκτη ταχύτητα
- Εάν το Load μείνει αναμένο ελάχιστα, δεν ξέρουμε πόσο θα προοχωρήσει το κάθε bit
- Εάν το Load μείνει αναμένο πολύ, οι αρ. είσοδοι θα δια-δοθούν και σβήσουν τα πάντα

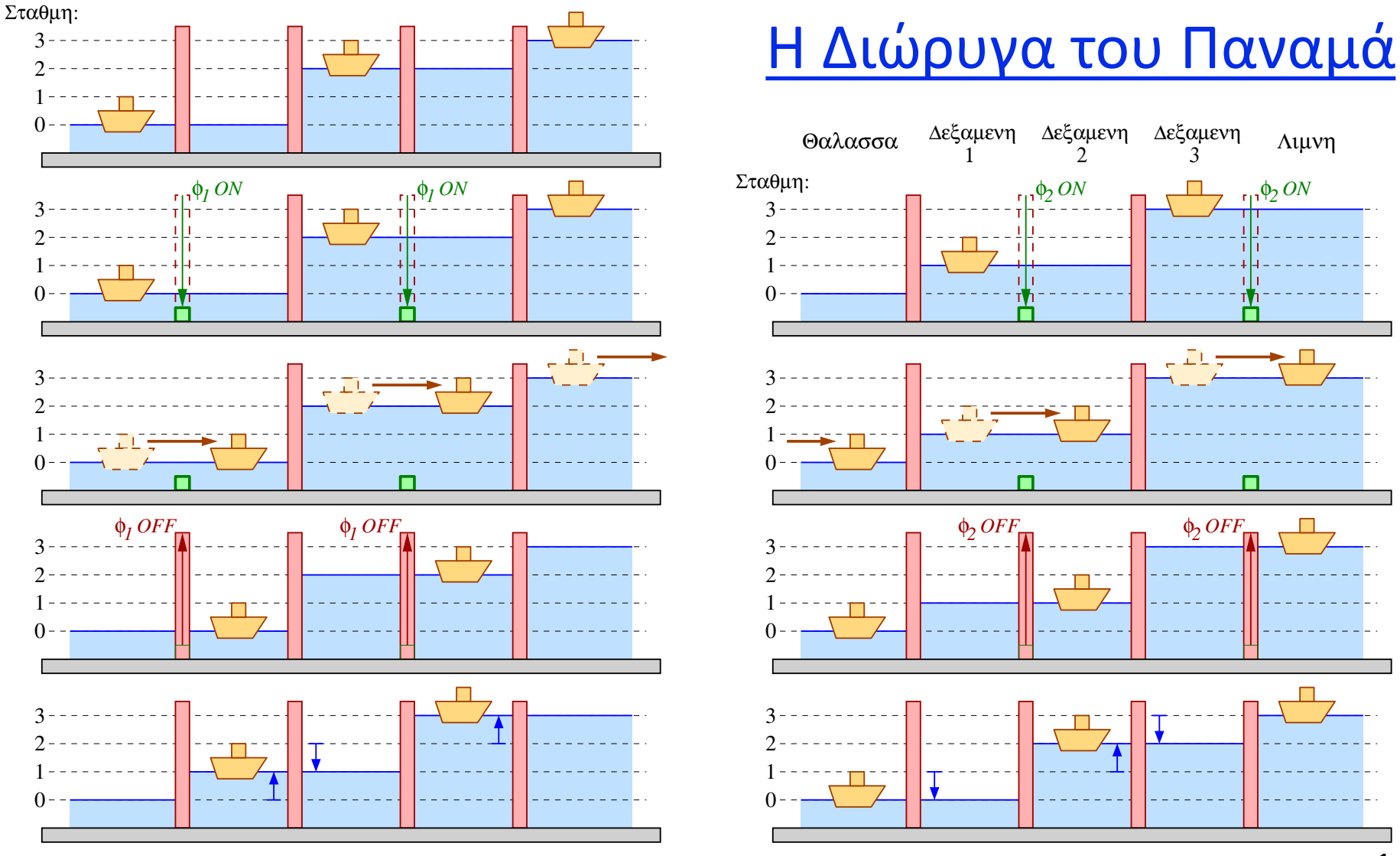
Η λύση: Διφασικά μη Αλληλοκαλυπτόμενα Ρολόγια

- Χρειάζονται ενδιάμεσες ("tmp") θέσεις αποθήκευσης
  - Όπως στις γλώσσες προγραμ. το "swap" μεταβλητών a, b:
  - δεν γίνεται απλώς γράφοντας: a=b; b=a;
  - απαιτεί ενδιάμεση tmp: tmp=a; a=b; b=tmp;
- Πριν φορτώσουμε νέα τιμή σε έναν μανταλωτή, κρατάμε αντίτυπο της παλαιάς του τιμής σε ενδιαμ. βοηθητικό μαντ.
- Η φόρτωση στους κύριους και στους βοηθητικούς μανταλωτές με εναλλασόμενες φάσεις ρολογιού, $\varphi_1 - \varphi_2$
- Μη αλληλοκαλυπτόμενες φάσεις $\varphi_1 - \varphi_2$:
  - πρώτα σβήνει η μία, μετά ανάβει η άλλη – όπως… ⟶

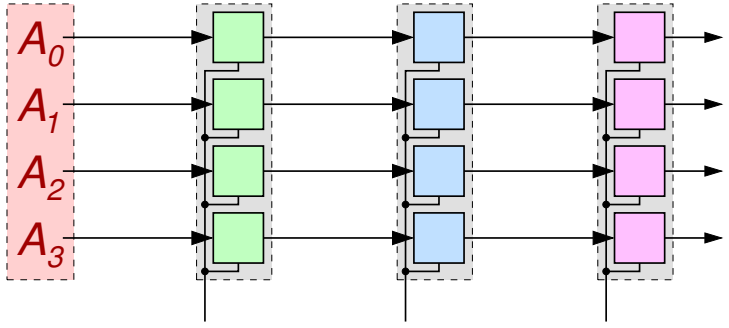
Μία αναλογία: Η Διώρυγα του Παναμά

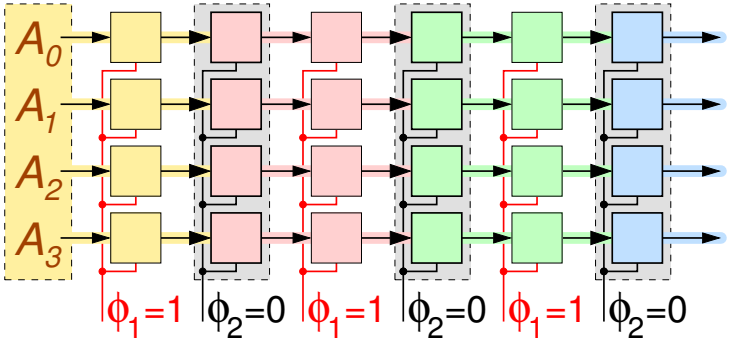
- Περνά ενδιάμεσα από μια λίμνη, πάνω σε ύψωμα
  - Τα πλοία ανεβαίνουν μέσω τριών Δεξαμενών, μεταξύ τεσσάρων Πυλών, με ενδιάμεσες στάθμες νερού
  - Κατεβαίνουν από την άλλη μέσω ανάλογων δεξαμενών-πυλών
- Πύλες ανοίγουν μόνον όταν Δεξαμενές στην ίδια στάθμη
  - Αλλιώς τα νερά και τα πλοία θα κατρακυλούσαν ανεξέλεγκτα
  - όπως εάν άναβαν ταυτόχρονα τα Load σε διαδοχικές βαθμίδες
  - *Ανοιγοκλείνουν **εναλλάξ**, όπως και τα **Διφασικά Ρολόγια***
- Διαδραστική Προσομοίωση: www.csd.uoc.gr/~hy120/26f/07b_panamaLocks.html
  - ή όπως τα βήματα στην επόμενη διαφάνεια… →

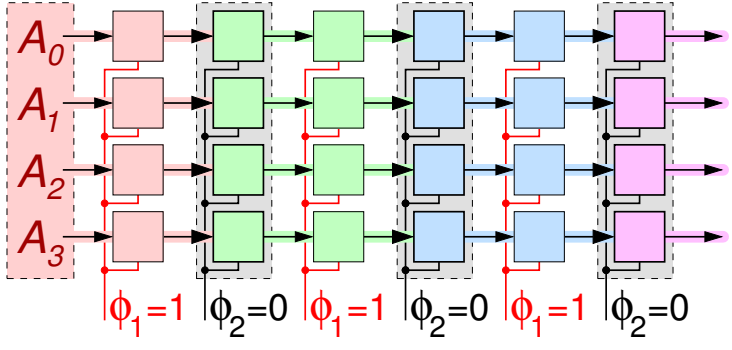
Η Διώρυγα του Παναμά

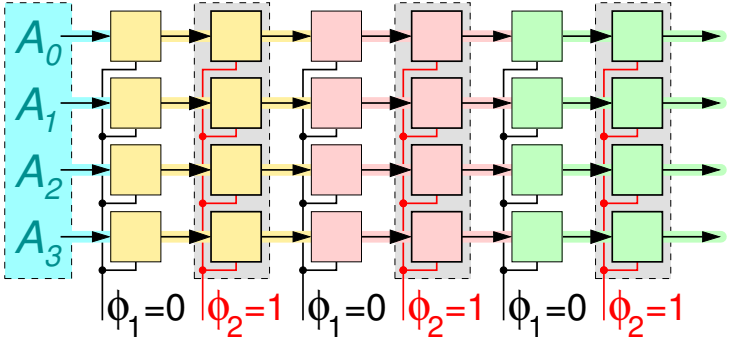


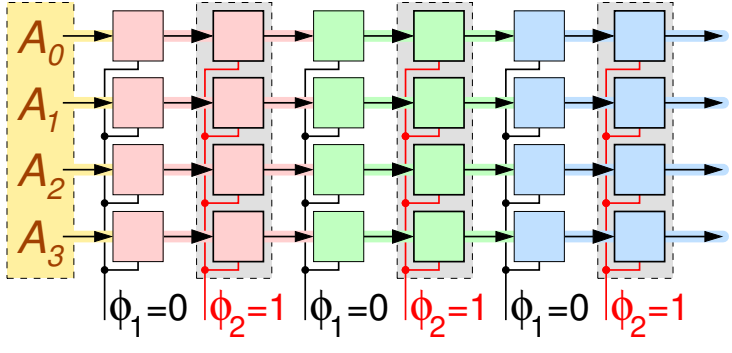
Διφασικά μη Αλληλοκαλ. Ρολόγια

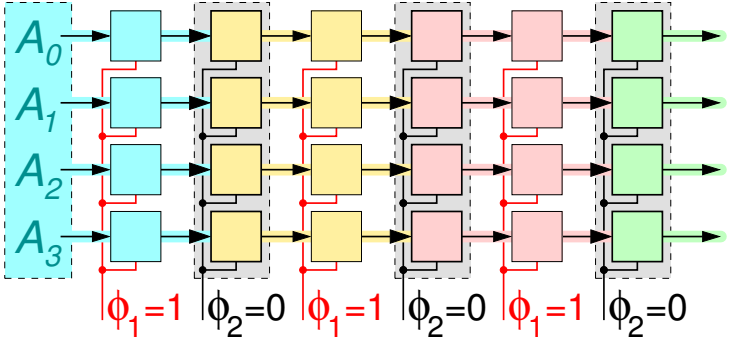




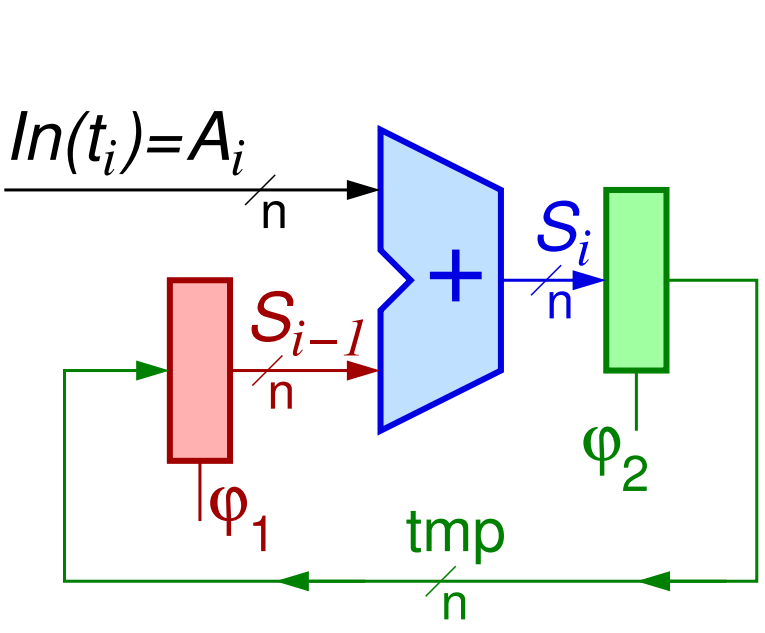


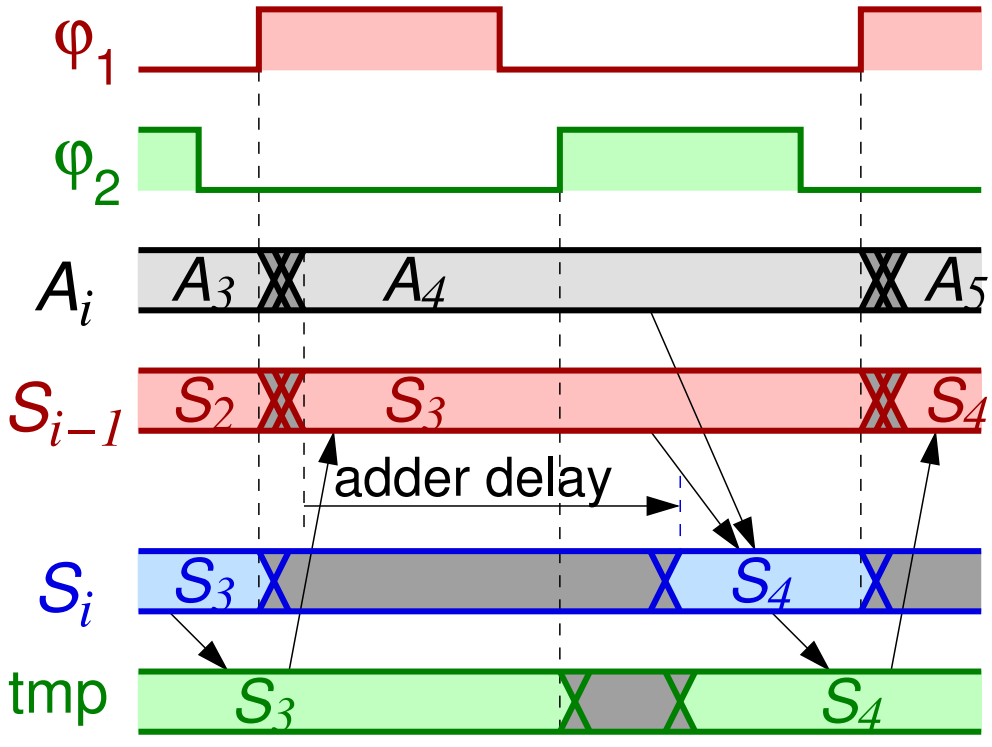






Το Σωρευτικό Άθροισμα με Διφασικό Ρολόϊ





- φ_1: άφιξη νέου αριθμού και προηγ. αθροίσματος στον αθροιστή
- φ_2: αποθήκευση νέου αθροίσματος με ασφάλεια γιά το μέλλον