ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Ψηφιακή Σχεδίαση

Εργαστήριο 7: Μανταλωτές (Latches), Καταχωρητές, και Διφασικά Ρολόγια

Μανόλης Γ.Η. Κατεβαίνης

Τμήμα Επιστήμης Υπολογιστών

Άδειες Χρήσης

•Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο 3.0 Ελλάδα
(Attribution – Non Commercial – Non-derivatives 3.0 Greece)

CC BY-NC-ND 3.0 GR

•Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

•Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
•Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» 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 & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

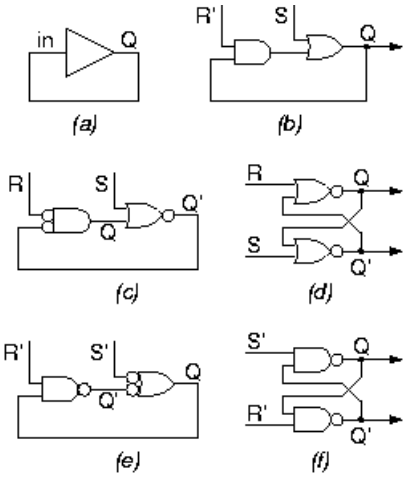
Ψηφιακή Σχεδίαση

Εργαστήριο 7:
Μανταλωτές (Latches), Καταχωρητές, και Διφασικά Ρολόγια

[Βιβλία: *προαιρετικά* μπορείτε να διαβάσετε: Wakerly: το κεφ. 7 μέχρι και την § **7.2.4** (σελ. 625-639)· Mano: § **5.1-5.3** (σελίδες 182-188)].

7.1 Ο βασικός Μανταλωτής (Latch, Flip-Flop) τύπου RS

Στο πείραμα 3.4 είχαμε δεί ότι τα κυκλώματα με θετική ανάδραση έχουν συνήθως δύο σταθερές καταστάσεις, κι έτσι χρησιμοποιούνται σαν μνήμες. Το βασικό τέτοιο κύκλωμα λέγεται flip-flop, ή άλλες φορές, ανάλογα κυρίως με τη χρήση και το χρονισμό, λέγεται "**μανταλωτής**" (**latch**). Στο παρακάτω σχήμα φαίνεται η εξέλιξη του κυκλώματος γιά τη δημιουργία του βασικού μανταλωτή τύπου RS. Το σχήμα (a) δείχνει τη βασική ιδέα της θετικής ανάδρασης: η πολικότητα της ανάδρασης πρέπει να μην έχει αντιστροφή --π.χ. μπορούμε να χρησιμοποιήσουμε δύο αντιστροφείς εν σειρά, όπως στο πείραμα 6.10. Το κύκλωμα αυτό ναι μεν είναι ένας μανταλωτής, πλην όμως δεν έχει είσοδο, άρα δεν έχει και δυνατότητα εγγραφής. Αν στη θέση της πύλης θετικής πολικότητας που φαίνεται βάλουμε μιά σκέτη πύλη AND (ή μιά σκέτη πύλη OR), τότε ναι μεν αποκτάμε δυνατότητα γιά κάποια εγγραφή, αλλά αυτή είναι πολύ περιορισμένη: με μιά πύλη AND μπορούμε να γράψουμε μόνο "0", ενώ με μιά πύλη OR μπορούμε να γράψουμε μόνο "1".



Η λύση είναι να βάλουμε στο βρόχο ανάδρασης και μιά πύλη AND και μιά πύλη OR, η πρώτη γιά να γράφουμε 0 ("reset") και η δεύτερη γιά να γράφουμε 1 ("set"), όπως φαίνεται στο σχήμα (b). Όταν R=S=0 (R'=1), οι πύλες AND και OR απλώς περνούν τη δεύτερη είσοδό τους στην έξοδο, άρα η λογική τιμή που υπήρχε στο βρόχο συνεχίζει ες αεί να "κυκλοφορεί" σε αυτόν, επομένως το κύκλωμα παραμένει στην μία από τις δύο σταθερές καταστάσεις του, δηλαδή απομνημονεύει 1 bit πληροφορίας. Όταν S=1, η έξοδος Q τίθεται σε κατάσταση 1, ανεξαρτήτως της προηγούμενής της κατάστασης. Όταν R=1, δηλ. R'=0 (και S=0), η έξοδος Q επαναφέρεται στην κατάσταση 0, πάλι ανεξαρτήτως της προηγούμενής της κατάστασης. Αυτές οι δύο συνθήκες, S=1 ή R=1 (R'=0), είναι οι συνθήκες εγγραφής· όταν οι είσοδοι εγκαταλείψουν μιά συνθήκη εγγραφής και επιστρέψουν στη συνθήκη απομνημόνευσης (S=R=0), η πληροφορία που είχε εγγραφεί παραμένει. Συνήθως αποφεύγουμε το κύκλωμα να έρχεται στη συνθήκη ταυτόχρονης εγγραφής και 1 και 0 (S=R=1), διότι αν τα S και R εγκαταλείψουν αυτή τη συνθήκη "ταυτόχρονα" είναι απροσδιόριστο το τι τιμή θα μείνει αποθηκευμένη στο μανταλωτή.

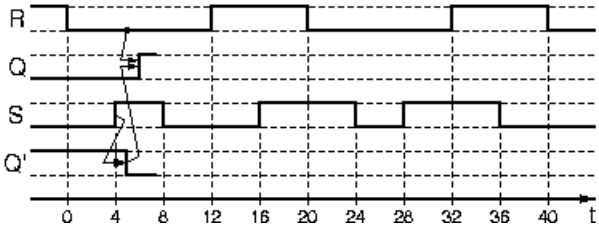
Στην τεχνολογιά CMOS, οι πύλες AND και OR δεν υπάρχουν, και πρέπει να συντεθούν από NAND/NOR· μπορούμε να κερδίσουμε σε κατασκευαστική απλότητα μετατρέποντας το κύκλωμα (b) στο κύκλωμα (c), εισάγοντας δύο αντιστροφές στο βρόχο ανάδρασης. Η μία αντιστροφή προσάπτεται στην έξοδο της πύλης OR, μετατρέποντας την έτσι σε πύλη NOR, η δε άλλη αντιστροφή προσάπτεται στις

εισόδους της πύλης AND (αλλάζοντας και την πολικότητα του σήματος R' σε R), μετατρέποντας την έτσι και αυτήν σε πύλη NOR (κανόνας DeMorgan). Ξανασχεδιάζοντας το κύκλωμα (c) με ισοδύναμη τοπολογιά και σύμβολα, παίρνουμε το "κλασσικό" κύκλωμα μανταλωτή τύπου RS με πύλες NOR που φαίνεται στο σχήμα (d). Ένα άλλο πλεονέκτημα που κερδίζουμε πηγαίνονας από μη αντιστρέφουσες πύλες (AND, OR) σε αντιστρέφουσες (NOR), είναι ότι τώρα έχουμε στη διάθεσή μας και τις δύο πολικότητες της αποθηκευμένης πληροφορίας, Q και Q'. Παρατηρήστε ότι στην πραγματικότητα η έξοδος που ονομάζουμε Q' είναι το συμπλήρωμα της Q μόνον όταν οι είσοδοι R και S δεν είναι στην "ανεπιθύμητη" συνθήκη R=S=1.

Εναλλακτικά, το κύκλωμα (b) μπορεί να μετατραπεί στο κύκλωμα (e), εισάγοντας τις δύο αντιστροφές σε διαφορετικό σημείο του βρόχου ανάδρασης. Το κύκλωμα (f) που προκύπτει, με πύλες NAND αντί NOR, είναι εξ' ίσου "κλασσικό" με το (d), και είναι απλά το δυϊκό του· εδώ, τα σήματα εισόδου, S' και R', έχουν αρνητική πολικότητα.

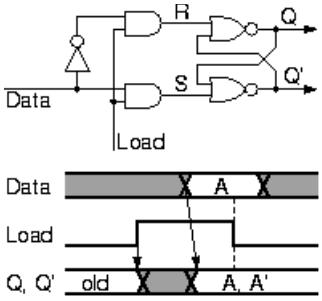
<u>Άσκηση 7.2:</u> Λειτουργία του Μανταλωτή RS

[**Πριν** το εργαστήριο]. Συμπληρώστε στο διπλανό σχήμα τη χρονική συμπεριφορά των κόμβων Q και Q' στον μανταλωτή RS του σχήματος (d) παραπάνω, όταν οι είσοδοι R και S μεταβάλονται όπως δείχνει στο σχήμα. Υποθέτουμε ότι κάθε πύλη έχει καθυστέρηση 1 χρονική μονάδα. Τα βέλη δείχνουν την αιτία των αλλαγών τιμής των κόμβων.



7.3 Ο Μανταλωτής (Latch) τύπου D

Ο μανταλωτής RS βολεύει σε εφαρμογές όπου οι αιτίες "θέσης" (εγγραφής 1) είναι διαφορετικές από τις αιτίες επαναφοράς (εγγραφής 0). Γιά παράδειγμα, έστω ότι υπάρχουν κάμποσοι ανιχνευτές σφαλμάτων ή ανεπιθύμητων καταστάσεων, και ότι χρειαζόμαστε μιάν ένδειξη "συναγερμού" που να μας ενημερώνει ότι κάτι πήγε στραβά από την τελευταία φορά που κοιτάξαμε. Τότε μπορούμε να χρησιμοποιήσουμε έναν μανταλωτή RS και να τροφοδοτήσουμε την είσοδό του S από μιά πύλη Ή που δέχεται σαν εισόδους τα σήματα όλων των ανιχνευτών· η έξοδος Q του μανταλωτή θα θυμάται αν κάτι πήγε στραβά, μέχρις ότου ο χειριστής τον επαναφέρει στην κατάσταση ηρεμίας, ενεργοποιώντας στιγμιαία το σήμα R.



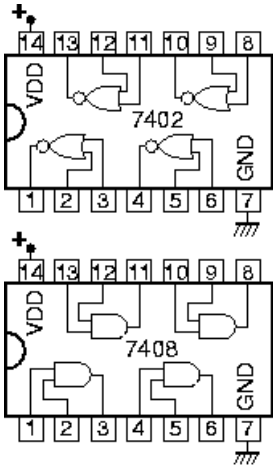
Στις περισσότερες εφαρμογές, όμως, οι ανάγκες είναι διαφορετικές: θέλουμε να αποθηκεύσουμε στο μανταλωτή τη λογική τιμή ενός σήματος (είτε αυτή είναι 0 είτε 1) όταν ένα άλλο σήμα μας λέει ότι ήλθε η ώρα να την αποθηκεύσουμε. Γιά τις εφαρμογές αυτές, το κατάλληλο κύκλωμα μανταλωτή είναι αυτό που φαίνεται δίπλα. Εδώ έχουμε ένα μανταλωτή RS του οποίου προηγείται λίγη συνδυαστική λογική: οι δύο πύλες ΚΑΙ φροντίζουν ώστε έγγραφή στο μανταλωτή να γίνεται τότε και μόνο τότε όταν το σήμα Load ("φόρτωσε") είναι ενεργό (Load=1), και ο αντιστροφέας φροντίζει ώστε, όποτε γίνεται εγγραφή (Load=1), ένα και μόνον ένα από τα σήματα R και S να είναι ενεργό (1), καθορίζοντας έτσι το ποιά τιμή θα γραφτεί, και αποκλείοντας την ανεπιθύμητη συνθήκη R=S=1· το ποιό από τα R ή S θα ενεργοποιηθεί καθορίζεται από την είσοδο Data, που φέρνει τα δεδομένα εγγραφής. Το συνολικό αυτό κύκλωμα του μανταλωτή με τη συνδυαστική λογική στην είσοδό του ονομάζεται **μανταλωτής**

τύπου D (D-type latch), παίρνοντας το όνομά του από την είσοδο D (Data).

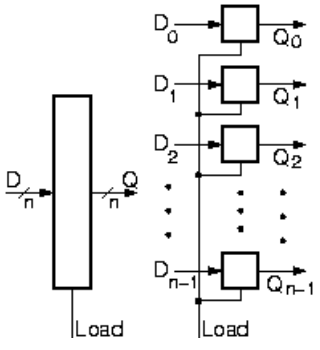
Το "διάγραμμα χρονισμού" (κάτω από το κύκλωμα) δείχνει τη συμπεριφορά του μανταλωτή D στο χρόνο. Ο οριζόντιος άξονας εξυπακούεται ότι είναι ο χρόνος. Όταν Load=0 (αριστερά και δεξιά στο διάγραμμα), οι έξοδοι Q και Q' παραμένουν σταθερές, είτε στο 0 είτε στο 1, όπως δείχνουν οι οριζόντιες γραμμές, ανάλογα με το ποιά τιμή είχε αποθηκευτεί στο μανταλωτή την τελευταία φορά που ήταν αναμένο το Load. Όταν Load=1 (στο μέσον του διαγράμματος), η έξοδος Q ισούται με την είσοδο Data (με μιά μικρή καθυστέρηση), η δε έξοδος Q' είναι το συμπλήρωμά τους. Η σκιασμένη περιοχή στο διάγραμμα της εισόδου Data δείχνει ότι η είσοδος αυτή μπορεί να μεταβάλλεται στο χρονικό αυτό διάστημα, δηλαδή να "ανεβοκατεβαίνει", χωρίς βέβαια να αποκλείεται και να είναι σταθερή, σε όλο ή σε μέρος αυτού του διαστήματος. Η είσοδος Data ζητάμε να παραμένει σταθερή, έχοντας την τιμή "A", μόνο σε ένα "χρονικό παράθυρο" γύρω από την "κατερχόμενη ακμή" του σήματος Load, ούτως ώστε η τιμή αυτή A να προλάβει να μπεί και να αποθηκευτεί με ασφάλεια στο μανταλωτή προτού σβήσει το σήμα Load, και να είναι η τελευταία τιμή που μπήκε εκεί πριν σβήσει το Load, ούτως ώστε να είναι και η τιμή που θα παραμείνει τελικά αποθηκευμένη, ανεξαρτήτως του ότι η είσοδος Data μπορεί να ξαναλλάξει τιμή μετά το σβήσιμο του Load.

<u>Πείραμα 7.4:</u> Μανταλωτής τύπου D

Κατασκευάστε και ελέγξτε τη λειτουργία του μανταλωτή τύπου D της προηγουμένης παραγράφου 7.3. Χρησιμοποιήστε ένα chip πυλών AND, τύπου 7408, κι ένα chip πυλών NOR, τύπου 7402, των οποίων η θέση των ακροδεκτών φαίνεται στο σχήμα. Λεπτομερείς πληροφορίες γιά τα chips 7402 μπορείτε να βρείτε στην §3.7 ή στο http://www.ti.com/lit/ds/symlink/sn74ls02.pdf . Φτιάξτε τον αντιστροφέα που χρειάζεστε μέσω μιάς από τις δύο ελεύθερες πύλες NOR, γειώνοντας τη μία είσοδό της. Τροφοδοτήστε την είσοδο Load π.χ. από τον διακόπτη A, και την είσοδο Data π.χ. από τον M. Συνδέστε τις εξόδους Q' και Q στις LED.0 και LED.1 αντίστοιχα. Κάντε ένα πλήρες σχεδιάγραμμα συνδεσμολογίας, όπως στην §4.10. Μην ξεχάσετε τα pins τροφοδοσίας. Επιβεβαιώστε τη σωστή λειτουργία: έχοντας την είσοδο Load αδρανή, αλλάξτε επανειλημμένα την είσοδο Data, και επαληθεύστε ότι δεν αλλάζουν οι έξοδοι Q και Q'· μετά, με την είσοδο Load ενεργή, επαληθεύστε ότι οι έξοδοι Q και Q' ακολουθούν τις αλλαγές της εισόδου Data. Επίσης, επαληθεύστε ότι οι έξοδοι Q και Q' είναι πάντα συμπληρωματικές μεταξύ τους.



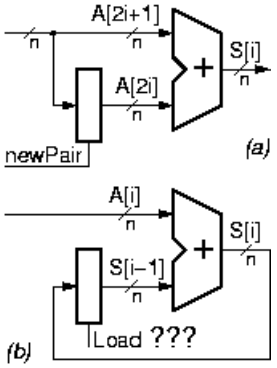
7.5 Μανταλωτές Πολλαπλών Bits (Καταχωρητές - Registers)



Συχνά θέλουμε να αποθηκεύσουμε ολόκληρες λέξεις, αποτελούμενες από πολλαπλά bits. Γιά την αποθήκευση μιάς λέξης των *n* bits απαιτούνται *n* μανταλωτές. Συνήθως, όλα τα bits της λέξης διατίθενται ταυτόχρονα γιά εγγραφή, οπότε αρκεί ένα κοινό σήμα φόρτωσης (Load) και γιά τους *n* μανταλωτές. Το κύκλωμα που προκύπτει φαίνεται στο σχήμα, σε μορφή συμβόλου και περιεχομένου· στη δεξιά πλευρά του σχήματος, κάθε τετράγωνο κουτί παριστά κι από έναν μανταλωτή του 1 bit. Τέτοιοι μανταλωτές πολλαπλών bits λέγονται και **καταχωρητές** (registers).

7.6 Επαναχρησιμοποίηση Πόρων, Σήματα Χρονισμού, Ακολουθιακά Κυκλώματα

Η αξιοποίηση των ψηφιακών κυκλωμάτων απαιτεί αυτά να τα χρησιμοποιούμε ξανά και ξανά --πολλά εκατομμύρια φορές το δεύτερόλεπτο-- προκειμένου να επεξεργαζόμαστε μεγάλο πλήθος διαφορετικών δεδομένων με παρόμοιο τρόπο κάθε φορά. Γιά να επιτευχθεί αυτή η επαναχρησιμοποίηση των κυκλωματικών πόρων χρειαζόμαστε **σήματα χρονισμού** (timing signals), σήματα δηλαδή που μας λένε πότε ήλθε η ώρα να αλλάξουμε τις εισόδους του κυκλώματός μας, ούτως ώστε αυτό να υπολογίσει κάτι καινούργιο. Σχεδόν πάντα, η επαναχρησιμοποίηση αυτή των κυκλωμάτων συνοδεύεται και από την ανάγκη αυτά να έχουν **μνήμη**, προκειμένου να μπορεί η επόμενη λειτουργία τους να επηρεάζεται από τα αποτελέσματα της προηγούμενης. **Συνδυαστικά** κυκλώματα (combinational circuits) είχαμε ονομάσει τα κυκλώματα εκείνα που δεν είχαν μνήμη, και των οποίων επομένως οι έξοδοι είναι συναρτήσεις των τρεχουσών τιμών των εισόδων τους και μόνο· η λειτουργία ενός συνδυαστικού κυκλώματος περιγράφεται πλήρως από τον πίνακα αληθείας του. **Ακολουθιακά** κυκλώματα (sequential circuits) ονομάζουμε τα κυκλώματα που περιέχουν μνήμη, και των οποίων επομένως οι έξοδοι είναι συναρτήσεις τόσο των τρεχουσών τιμών των εισόδων τους όσο και του παρελθόντος τους.



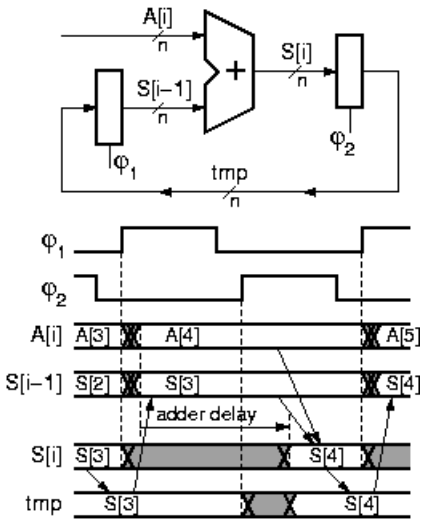
Το σχήμα δίπλα δείχνει δύο απλά παραδείγματα επαναχρησιμοποίησης ενός αθροιστή. Υποθέτουμε ότι η είσοδος δεδομένων (πλάτους *n* bits) που έρχεται από αριστερά μας φέρνει μιά συνεχή (χρονική) σειρά από αριθμούς, A[j]. Γιά την τροφοδότηση αυτών των αριθμών θα χρειαστούν προφανώς σήματα χρονισμού, αλλά αυτά δεν θα μας απασχολήσουν προς το παρόν. Στο σχήμα (a), θεωρούμε ότι η χρονική σειρά αποτελείται από *ζευγάρια* αριθμών, A[2i] και A[2i+1], γιά i=0, 1, 2, ..., και ότι εμείς θέλουμε να υπολογίζουμε το άθροισμα S[i] = A[2i] + A[2i+1] κάθε τέτοιου ζευγαριού αριθμών. Γιά να υπολογίσουμε το S[i], πρέπει να περιμένουμε να έλθει το A[2i+1]· όταν όμως έλθει το A[2i+1], το A[2i] έχει ήδη περάσει και φύγει, άρα γιά να μπορούμε να το προσθέσουμε πρέπει να το έχουμε κρατήσει κάπου --προφανώς σε έναν καταχωρητή. Το κύκλωμα που χρειαζόμαστε λοιπόν φαίνεται στο σχήμα (a). Κάθε φορά που στην είσοδο των αριθμών υπάρχει ο πρώτος από ένα νέο ζευγάρι αριθμών, τον φορτώνουμε (αποθηκεύουμε) στον καταχωρητή (*n* μανταλωτές). Γιά να γίνει η αποθήκευση αυτή, χρειαζόμαστε ένα σήμα χρονισμού, "newPair", που να μας λέει ότι αυτή τη στιγμή έρχεται ο πρώτος αριθμός ενός νέου ζευγαριού αριθμών· υποθέτουμε ότι το σήμα αυτό μας το δίνει η ίδια πηγή που μας δίνει και τους αριθμούς. Μετά από λίγο, όταν στη είσοδο έχει φτάσει ο δεύτερος αριθμός του ζευγαριού, ο αθροιστής θα βλέπει τους δύο αριθμούς του ζευγαριού στις δύο εισόδους του, κι επομένως θα παράγει (μετά από μιά μικρή καθυστέρηση) το άθροισμά τους στην έξοδό του.

Στο σχήμα (b), τα πράγματα είναι δυσκολότερα. Εδώ θέλουμε να υπολογίζουμε το σωρευτικό άθροισμα S[i] όλων των αριθμών που μας ήλθαν από την αρχή του χρόνου: S[i] = A[0] + A[1] + A[2] + ... + A[i]. Γιά να το πετύχουμε αυτό, κρατάμε το μέχρι προ ολίγου σωρευτικό άθροισμα, S[i-1], σ' έναν καταχωρητή, και προσθέτουμε σε αυτό τον νέο αριθμό, A[i], που μας έρχεται αυτή τη στιγμή. Μόλις γίνει η πρόσθεση αυτή, πρέπει το νέο άθροισμα που υπολογίσαμε, S[i], να αντικαταστήσει το προηγούμενο S[i-1] στον καταχωρητή, ούτως ώστε να είναι έτοιμο γιά την επόμενη πρόσθεση, με τον A[i+1]. Το πρόβλημα όμως είναι το εξής. Μόλις ανάψει το σήμα φόρτωσης του καταχωρητή, Load, ο αριθμός S[i] μπαίνει μέσα στους *n* μανταλωτές, και αμέσως εμφανίζεται στις εξόδους τους, *καταστρέφοντας* την τιμή S[i-1] που

υπήρχε εκεί πριν ανάψει το Load.

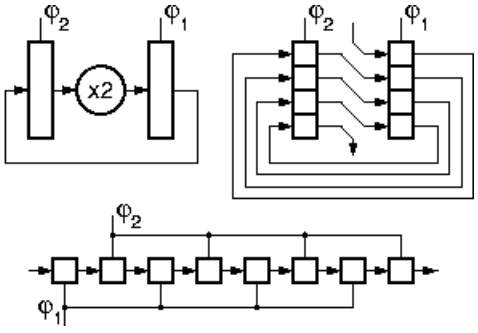
Μόλις όμως καταστραφεί η είσοδος S[i-1] του αθροιστή, καταστρέφεται και η έξοδος του αθροιστή, δεδομένου ότι ο αθροιστής είναι συνδυαστικό κύκλωμα (δεν θυμάται το παρελθόν --κοιτάζει μόνο το παρόν). Με την καταστροφή της εξόδου του αθροιστή, καταστρέφεται και η είσοδος του καταχωρητή, και επειδή το Load δεν πρόλαβε ακόμα να σβήσει, καταστρέφεται και η τιμή S[i] που είχε προς στιγμήν αποθηκευτεί εκεί. Μία λύση θα ήταν το Load να σβήσει "πολύ γρήγορα", πριν προλάβει η καταστροφή της εξόδου του καταχωρητή να περάσει από τον αθροιστή και να επιστρέψει στην είσοδο του καταχωρητή. Σε σπάνιες περιπτώσεις χρησιμοποιείται αυτή η λύση, αλλά είναι πολύ δύσκολο να επιτευχθεί η ακρίβεια της χρονικής διάρκειας του Load που απαιτείται. Αντ' αυτής, υπάρχει μιά απλούστερη λύση (και οι παραλλαγές της που θα πούμε σε λίγο) που την ακολουθούμε στη συντριπτική πλειοψηφία των περιπτώσεων.

Η απλούστερη, ασφαλέστερη, και καλύτερη λύση είναι να χρησιμοποιήσουμε μιάν ενδιάμεση, προσωρινή (temporary) θέση αποθήκευσης, tmp, όπως φαίνεται στο σχήμα δίπλα (η τεχνική αυτή λέγεται "double buffering" σε άλλες εφαρμογές στην επιστήμη υπολογιστών). Γιά να αρχίσει μιά νέα πρόσθεση, αντιγράφουμε το προηγούμενο σωρευτικό άθροισμα από την προσωρινή θέση στον καταχωρητή εισόδου του αθροιστή· η τιμή της προσωρινής θέσης δεν επηρεάζεται. Όταν τελειώσει η πρόσθεση, αντιγράφουμε το νέο άθροισμα στην προσωρινή θέση· η τιμή του καταχωρητή εισόδου δεν επηρεάζεται, άρα ούτε και η έξοδος του αθροιστή επηρεάζεται.



Γιά να γίνουν οι δύο αυτές χωριστές μεταφορές χρειαζόμαστε δύο χωριστά σήματα σήματα χρονισμού, φ_1 και φ_2, σαν σήματα φόρτωσης των καταχωρητών. Επειδή τα σήματα αυτά είναι περιοδικά και μας καθορίζουν το ρυθμό εργασίας του κυκλώματος, τα λέμε "ρολόγια". Επειδή έχουν την ίδια περίοδο και είναι στενά αλληλεξαρτημένα, λέμε ότι πρόκειται γιά τις **δύο φάσεις**, φ_1 και φ_2, ενός **διφασικού ρολογιού** (two-phase clock). Η ιδιότητα που έχουν αυτές οι δύο φάσεις είναι ότι ανάβουν εναλλάξ, και ότι ποτέ δεν είναι αναμμένες και οι δύο ταυτόχρονα. Επειδή ποτέ δεν ανάβουν ταυτόχρονα οι φ_1 και φ_2, ποτέ δεν θα υπάρχει ανοικτός δρόμος διαμέσου ολόκληρου του κύκλου που υπάρχει στο κύκλωμά μας.

Οι λεπτομέρειες της λειτουργίας του κυκλώματος φαίνονται στο διάγραμμα χρονισμού. Όταν ανάβει η φάση φ_1 του ρολογιού, η τιμή tmp, που ήταν το προηγούμενο άθροισμα, έστω S[3] εδώ, μπαίνει στον καταχωρητή S[i-1] και εμφανίζεται στην είσοδο του αθροιστή. Αμέσως, η παλαιά τιμή στην έξοδο του αθροιστή, S[i], καταστρέφεται, αλλά αυτό **δεν** επηρεάζει την αποθηκευμένη τιμή tmp, επειδή η φ_2 είναι σβηστή. Αργότερα, μετά την πάροδο της (σημαντικής) καθυστέρησης του αθροιστή, η έξοδός του, S[i], αποκτά την σωστή τιμή του νέου αθροίσματος, S[4]. Εν τω μεταξύ, η φ_1 έχει σβήσει, και μετά η φ_2 έχει ανάψει. Μόλις ανάψει η φ_2 καταστρέφεται η παλαιά τιμή του tmp, S[3], αλλά αυτό δεν μας πειράζει πιά, διότι την έχουμε ήδη αντιγράψει στον καταχωρητή S[i-1], όπου δεν κινδυνεύει πιά, αφού η φ_1 έχει σβήσει. Όταν υπολογιστεί το νέο άθροισμα, S[4], αυτό αποθηκεύεται στον καταχωρητή tmp, όπου και παραμένει εν ασφαλεία μετά το σβήσιμο της φ_2, ανεξαρτήτως του ότι η έξοδος του αθροιστή θα αλλάξει σε λίγο.

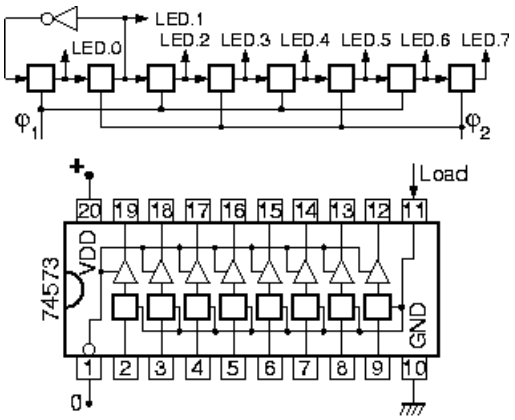


7.7 Εφαρμογή: Καταχωρητής Ολίσθησης

Ας θεωρήσουμε ένα κύκλωμα ανάλογο με το τελευταίο του αθροιστή, παραπάνω, αλλά αντί γιά πρόσθεση να κάνει πολλαπλασιασμό επί 2, δηλαδή ολίσθηση κατά μία θέση. Η ολίσθηση αυτή είναι εξαιρετικά γρήγορη --πρόκειται απλώς γιά σύρματα με κατάλληλη συνδεσμολογία, όπως φαίνεται στο σχήμα. Μ' ένα τόσο γρήγορο κύκλωμα στο βρόχο, η επίλυση του προβλήματος που λύσαμε με τα διαφασικά ρολόγια είναι περισσότερο επιτακτική από οπουδήποτε αλλού. Ξετυλίγοντας τη "σερπαντίνα" που σχηματίζουν οι μανταλωτές, ο ένας πίσω από τον άλλον, προκύπτει το γραμμικό κύκλωμα που φαίνεται στο κάτω μέρος του

σχήματος. Σε αυτό, φαίνεται σαφώς ότι αν ποτέ άναβαν ταυτόχρονα οι 2 φάσεις του ρολογιού, η αριστερή είσοδος του ολισθητή θα διεδίδετο τάχιστα σε όλους τους μανταλωτές, σβήνοντας κάθε παλαιά αποθηκευμένη πληροφορία. Αντ' αυτού, με το να ανάβουν οι δύο φάσεις εναλλάξ, τα bits που είναι αποθηκευμένα στον καταχωρητή (ένα bit γιά κάθε 2 μανταλωτές!) προχωρούν "συντεταγμένα" και τακτικά προς τα δεξιά, με το ρυθμό που ορίζει το διφασικό ρολόϊ.

Πείραμα 7.8: Ολίσθηση με Διφασικό Ρολόϊ



Κατασκευάστε έναν τέτοιο καταχωρητή ολίσθησης με διφασικό ρολόϊ στο εργαστήριο. Θυμηθείτε ότι κάθε "bit" αποθηκευμένης πληροφορίας "καταλαμβάνει" ένα ζευγάρι διαδοχικών μανταλωτών. Σαν σειριακή είσοδο δεδομένων, αριστερά, χρησιμοποιήστε το συμπλήρωμα του αριστερού bit του καταχωρητή (αριστερό bit = δύο αριστεροί μανταλωτές)· με αυτόν τον τρόπο, τα bits που θα ολισθαίνουν μέσα στον καταχωρητή θα είναι πάντα εναλλασόμενα μηδενικα και άσσοι (010101010...) (εναλλάσονται κάθε ένα bit δηλαδή κάθε δύο μανταλωτές). Σαν αντιστροφέα χρησιμοποιήστε μία από τις πύλες NOR του προηγουμένου πειράματος. Τους μισούς μανταλωτές --αυτούς που φορτώνονται στη φάση φ_1-- θα τους πάρετε από ένα chip 74573 (βλ. §3.7), και τους άλλους μισούς από ένα άλλο. Χρειάζεστε δύο chips διότι όλοι οι μανταλωτές ενός chip έχουν ένα, κοινό σήμα φόρτωσης. Οι έξοδοι του chip έχουν "τρικατάστατους οδηγητές" --θέμα γιά το οποίο θα μιλήσουμε αργότερα· γιά να λειτουργήσουν σωστά πρέπει να φέρετε στο μηδέν (να γειώσετε) το pin 1 του chip (Output Enable'). Χρησιμοποιήστε π.χ. τους διακόπτες A και B σαν τις 2 φάσεις του ρολογιού.

Κάντε ένα λεπτομερές σχεδιάγραμμα του κυκλώματος, όπως στην §4.10. Επίσης, σκεφτείτε τι θα συμβεί αν ενεργοποιήστε και τις 2 φάσεις του ρολογιού ταυτόχρονα. Πώς θα συμπεριφερθεί ο βρόχος αριστερά με τον αντιστροφέα; Τι θα κάνει η υπόλοιπη αλυσίδα των μανταλωτών; Επιβεβαιώστε τη σωστή λειτουργία: γεννήστε τις 2 φάσεις του ρολογιού πατώντας τους 2 διακόπτες εναλλάξ, και παρακολουθήστε την "κίνηση" των bits στις LED's. Δοκιμάστε να ενεργοποιήστε και τις 2 φάσεις του ρολογιού ταυτόχρονα, και δείτε αν επιβεβαιώνονται οι προβλέψεις σας.