

Εργαστήριο 6: Προσημασμένοι Ακέραιοι, Προσθαφαιρέτες, Flip-Flops

4 - 7 Νοεμβρίου 2014

Υπενθύμιση: Εξέταση Προόδου: Σάββατο 8 Νοεμβρίου 2014, ώρα 12-2.

[Βιβλία: *προαιρετικά* μπορείτε να διαβάσετε: Wakerly: § 2.5 - 2.6 (σελ. 41-52)· Mano: § 1.5 - 1.6 (σελ. 9-17)].

6.1 Πολλαπλασιασμός, Διαίρεση, και Υπόλοιπο με δυνάμεις του 2

Όπως στο δεκαδικό σύστημα ο πολλαπλασιασμός επί 10, 100, κλπ. είναι πολύ απλός, έτσι και στο δυαδικό ο **πολλαπλασιασμός επί δύναμη του 2** αντιστοιχεί σε απλή **αριστερή ολίσθηση** με γέμισμα μηδενικών από δεξιά. Όντως, αν B είναι ο δυαδικός αριθμός $b_{n-1}b_{n-2}\dots b_2b_1b_0$ με n bits (§5.1), τότε το γινόμενο του επί 2^k είναι:

$$\begin{aligned} B \cdot 2^k &= b_{n-1} \cdot 2^{(n-1)+k} + b_{n-2} \cdot 2^{(n-2)+k} + \dots + b_2 \cdot 2^{2+k} + b_1 \cdot 2^{1+k} + b_0 \cdot 2^{0+k} = \\ &= b_{n-1} \cdot 2^{n+k-1} + b_{n-2} \cdot 2^{n+k-2} + \dots + b_1 \cdot 2^{k+1} + b_0 \cdot 2^k + 0 \cdot 2^{k-1} + 0 \cdot 2^{k-2} + \dots + 0 \cdot 2^1 + 0 \cdot 2^0 \end{aligned}$$

Βάσει της τελευταίας αυτής ιδιότητας, ο αριθμός $B \cdot 2^k$ άναπαρίσταται στο δυαδικό με $n+k$ bits τα οποία είναι: τα n bits του αριθμού B "ολισθημένα" αριστερά κατά k θέσεις (δηλαδή με τη σημαντικότητα καθενός αυξημένη κατά k θέσεις), ακολουθούμενα από k μηδενικά. Για παράδειγμα, ο αριθμός 45 (δεκαδικό) = 101101 (δυαδικό), πολλαπλασιαζόμενος επί $16 = 2^4$ δίνει $45 \times 16 = 720$ (δεκαδικό), που στο δυαδικό είναι: 1011010000. Στο δεκαεξαδικό, ο πρώτος αριθμός (45) είναι ο 2D, και το γινόμενο του επί 16 (επί "10" στο δεκαεξαδικό) είναι: 2D0.

Κατ' αντίστοιχο τρόπο, η **διαίρεση διά δύναμη του 2** αντιστοιχεί σε **δεξιά ολίσθηση**: τα λιγότερο σημαντικά bits του αριθμού, που "εκδιώκονται" από τη δεξιά άκρη του, αποτελούν το **υπόλοιπο** της διαίρεσης. Για τον παραπάνω αριθμό B , η διαίρεσή του διά 2^k δίνει:

$$B / 2^k = (b_{n-1} \cdot 2^{(n-1)-k} + b_{n-2} \cdot 2^{(n-2)-k} + \dots + b_{k+1} \cdot 2^1 + b_k \cdot 2^0) + (b_{k-1} \cdot 2^{k-1} + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0) / 2^k$$

Ο πρώτος όρος του παραπάνω αθροίσματος είναι ακέραιος αριθμός, ενώ ο δεύτερος είναι κλασματικός, μικρότερος της μονάδας ή μηδεν, δεδομένου ότι ο αριθμητής $(b_{k-1} \cdot 2^{k-1} + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0)$ είναι ένας δυαδικός αριθμός με k bits, άρα πάντα μικρότερος του 2^k . Κατά συνέπεια, ο πρώτος όρος (τα αριστερά $n-k$ bits του αρχικού αριθμού) είναι το ακέραιο **πηλίκο** της διαίρεσης, ενώ ο αριθμητής του δευτέρου όρου (τα δεξιά k bits του αρχικού αριθμού) είναι το **υπόλοιπο** της (ακέραιας) διαίρεσης. Για παράδειγμα, ο αριθμός 205 (δεκαδικό) = 11001101 (δυαδικό), διαρνούμενος διά $8 = 2^3$ δίνει πηλίκο 25 (δεκαδικό) = 11001 (δυαδικό), και υπόλοιπο 5 (δεκαδικό) = 101 (δυαδικό). Στο οκταδικό, ο διαιρέτος είναι 315, ο διαιρέτης είναι 10, το πηλίκο είναι 31 (=3x8+1=25 στο δεκαδικό), και το υπόλοιπο είναι 5.

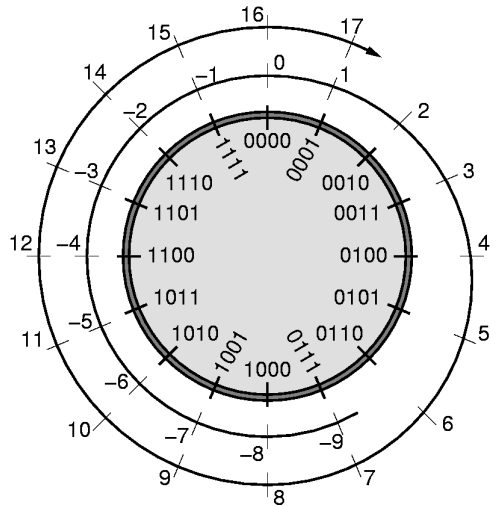
Μιά συνηθισμένη εφαρμογή στους υπολογιστές είναι όταν μιά μεγάλη μνήμη, π.χ. 256 Mbytes, κατασκευάζεται από κάμποσες --π.χ. δεκαέξι (16)-- μικρότερες, μεγέθους δύναμης του 2 η καθεμία --εδώ μεγέθους 16 MBytes = 16,777,216 Bytes καθεμία. Εάν μας ζητηθεί να προσπελάσουμε το Byte με διεύθυνση 167,772,560, σε ποιάν από τις 16 μικρότερες μνήμες βρίσκεται αυτό και τι διεύθυνση μέσα σε αυτήν έχει; Για να βρούμε σε ποιά μνήμη θα το αναζητήσουμε, πρέπει να διαιρέσουμε τη διεύθυνση 167,772,560 διά το μέγεθος 16,777,216 των επιμέρους μνημών, που σ' αυτήν την περίπτωση μας δίνει 10 (δεκαδικό) (πρόκειται για την 11η μνήμη, αφού οι επιμέρους μνήμες αριθμούνται 0, 1, 2, ..., 15)· το υπόλοιπο της διαίρεσης, 400 (δεκαδικό), μας δίνει την επιθυμητή διεύθυνση μέσα στην επιμέρους μνήμη. Η διεύθυνση του επιθυμητού Byte είναι 167,772,560 (δεκαδικό) = A,00,01,90 (δεκαεξαδικό) = 1010,00000000,00000001,10010000 (δυαδικό - 28 bits). Το μέγεθος της κάθε επιμέρους μνήμης (διαιρέτης) είναι $16 \text{ M} = 2^{24}$, άρα το ζητούμενο Byte βρίσκεται στη θέση 00...0110010000 (υπόλοιπο διαίρεσης, δηλαδή δεξιά 24 bits της διεύθυνσης) της μνήμης υπ' αριθμόν 1010 (πηλίκο

διαίρεσης, δηλαδή αριστερά $28-24 = 4$ bits της διεύθυνσης) μετατρέποντας στο δεκαδικό, βλέπουμε ότι όντως πρόκειται για τη θέση $256+128+16 = 400$ της μνήμης υπ' αριθμόν $8+2 = 10$.

6.2 Συστροφή (wrap-around) Αναπαράστασης Αριθμών με n bits

Μία άλλη εφαρμογή της παραπάνω παρατήρησης σχετικά με το υπόλοιπο της διαίρεσης διά δύναμη του 2 είναι η ερμηνεία των αριθμητικών πράξεων με πεπερασμένο πλήθος bits. Ας φανταστούμε ότι οι ακέραιοι αριθμοί των μαθηματικών έχουν πάρα πολλά bits --όσα χρειάζονται για το μέγεθός τους-- αλλά εμείς κάνουμε πράξεις μ' έναν υπολογιστή που έχει πεπερασμένο πλήθος bits, έστω n bits, και γι' αυτό κρατάμε μόνο τα n λιγότερο σημαντικά (δεξιά) bits του "μαθηματικού" αριθμού. Έτσι, όταν προσθέτουμε αριθμούς (§5.4) μ' έναν αθροιστή μεγέθους n bits, αγνοούμε (πετάμε) το κρατούμενο που βγαίνει από την αριστερή άκρη (MSB), διότι δεν έχουμε πού να το αποθηκεύσουμε. Επομένως, αν A είναι το πραγματικό (μαθηματικό) αποτέλεσμα της πράξης μας, ο υπολογιστής μας τελικά θα κρατήσει μόνο το **υπόλοιπο** της διαίρεσης του A διά 2^n (που συνήθως συμβολίζεται: $A \bmod 2^n$).

Εάν σε αυτό τον υπολογιστή, με το πεπερασμένο πλήθος των n bits, αρχίσω να μετρώ από το 0 προς τα πάνω, όταν φτάσω στον αριθμό 2^n-1 , ο επόμενος αριθμός θα φανεί σαν να είναι πάλι ο 0, και μετά θα συνεχίσω να μετρώ πάλι από την αρχή, 1, 10, 11, 100, κλπ. Το φαινόμενο αυτό είναι σαν να έχω ένα τροχό με περίμετρο 2^n και με χαραγμένους επάνω του τους 2^n υπάρχοντες συνδυασμούς των n bits, δεξιόστροφα από το 00...000 μέχρι το 11...111, και να **τύλιγω** επάνω σε αυτόν τον τροχό τον άξονα των αριθμών, όπως φαίνεται στο σχήμα δίπλα για $n=4$ bits. Έστω ότι ξεκινάμε το τύλιγμα έτσι ώστε ο ακέραιος αριθμός 0 να συμπίπτει με τον δυαδικό κώδικα 00...000. Τότε, οι 2^n κώδικες του τροχού θα συμπίψουν με τους ακεραίους αριθμούς από 0 έως



2^n-1 , όπως ακριβώς καθορίζει ο γνωστός μας από την §5.1 κώδικας δυαδικής αναπαράστασης των *μη προσημασμένων ακεραίων*. Όταν όμως εξαντλούνται οι υπάρχοντες συνδυασμοί των n bits, οι περαιτέρω αριθμοί (π.χ. 16, 17,... στο σχήμα) έχουν "τυλιχτεί" πάνω στους ίδιους κώδικες, 000, 001, κλπ, ξανά από την αρχή. Αν λοιπόν μία πρόσθεση δώσει αποτέλεσμα από 2^n και πάνω, το αποτέλεσμα αυτό, στον υπολογιστή με τα n bits, θα μοιάζει σαν ένας μικρότερος αριθμός -- αυτός που προκύπτει από το "τύλιγμα". Το φαινόμενο αυτό ονομάζεται "**wrap around**" -- περιτύλιγμα, περιέλιξη, ή *συστροφή*-- των ακεραίων αριθμών γύρω από τον "τροχό" των πεπερασμένων συνδυασμών που έχει τη δυνατότητα να παραστήσει ο υπολογιστής.

Εάν τώρα θεωρήσουμε και τους αρνητικούς αριθμούς, στον ίδιο άξονα των αριθμών τον συνεστραμμένο γύρω από τον τροχό, όπως φαίνεται στο σχήμα, τότε αποκτάμε μία μέθοδο -- έναν κώδικα-- αναπαράστασης και αρνητικών αριθμών. Ο κώδικας αυτός, που ονομάζεται "συμπλήρωμα ως προς 2", χρησιμοποιείται σήμερα σε όλους τους υπολογιστές για την αναπαράσταση "προσημασμένων ακεραίων" (signed integers), και έχει πολύ σημαντικά πλεονεκτήματα απλότητας έναντι άλλων, εναλλακτικών κωδίκων. Στην καθημερινή μας ζωή παριστάνουμε τους προσημασμένους ακεραίους χρησιμοποιώντας έναν διαφορετικό κώδικα, τον κώδικα προσήμου - απόλυτης τιμής (sign-magnitude representation). Η κατεύθυνση αύξησης της απόλυτης τιμής, όμως, είναι άλλοτε δεξιόστροφα (θετικοί αριθμοί) και άλλοτε αριστερόστροφα (αρνητικοί αριθμοί), πάνω στον συνεστραμμένο άξονα των αριθμών. Αυτή η αλλαγή φοράς αύξησης έχει σαν συνέπεια, όταν προσθέτουμε προσημασμένους ακεραίους στην καθημερινή μας ζωή, άλλοτε να πρέπει να κάνουμε πρόσθεση κι άλλοτε αφαίρεση των απόλυτων τιμών τους, και μάλιστα πριν από την αφαίρεση να πρέπει να συγκρίνουμε τις δύο απόλυτες τιμές για να βρούμε ποιά είναι η μικρότερη και να αφαιρέσουμε αυτήν από την άλλη.

Αντ' αυτού, οι υπολογιστές ακολουθούν τον πολύ απλούστερο τρόπο που πηγάζει από την παραπάνω μέθοδο της "συστροφής": επειδή η (αλγεβρική) αύξηση μιάς τιμής --είτε θετικής είτε αρνητικής-- αντιστοιχεί πάντα σε δεξιόστροφη κίνηση πάνω στον τροχό, προκύπτει ότι αρκεί πάντα να κάνουμε πρόσθεση και μόνο, ανεξαρτήτως του αν προσθέτουμε θετικούς ή αρνητικούς

αριθμούς! Η άλλη βασική παρατήρηση είναι ότι η (αλγεβρική) ελάττωση μιάς τιμής, δηλαδή η πρόσθεση ενός αρνητικού αριθμού --π.χ. του (-1)-- που αντιστοιχεί σε αριστερόστροφη κίνηση πάνω στον τροχό --π.χ. κατά 22.5 μοίρες εδώ-- μπορεί να προκύψει ισοδύναμα και σαν πρόσθεση ενός "μεγάλου" θετικού αριθμού --του 15 στο εδώ παράδειγμα, που αντιστοιχεί σε δεξιόστροφη κίνηση κατά $360-22.5 = 337.5$ μοίρες. Αν λοιπόν κωδικοποιήσουμε το -1 με τον ίδιο κώδικα όπως και το 15, τότε η πρόσθεση αυτού του κώδικα με έναν τετράμπιτο αθροιστή θα φέρνει το ίδιο αποτέλεσμα όπως η πρόσθεση του -1.

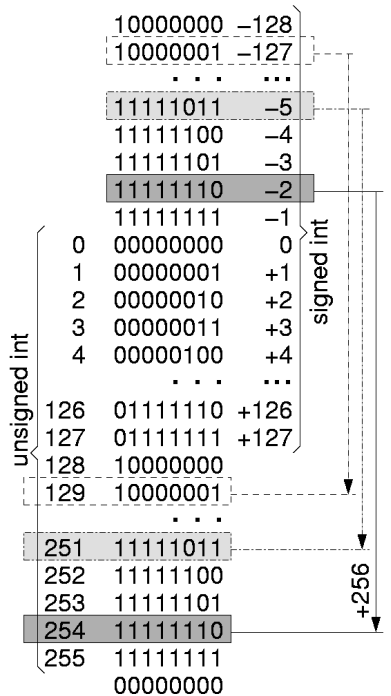
Ένας άλλος τρόπος να καταλάβουμε το τι γίνεται όταν κάνουμε προσθέσεις με n bits και κρατάμε μόνο τα n bits του αποτελέσματος, δηλαδή πετάμε (αγνοούμε) το κρατούμενο εξόδου, το $(n+1)$ -οστό bit, είναι ο εξής. Εάν προσθέσουμε τους αριθμούς A και B και το αποτέλεσμα της πρόσθεσης $A+B$ είναι $\geq 2^n$, τότε, με το να πετάξουμε (αγνοήσουμε) το $(n+1)$ -οστό bit του αποτελέσματος, που είναι άσος και έχει αξία 2^n , σημαίνει ότι το αποτέλεσμα που τελικά κρατάμε, χωρίς αυτόν τον άσο στη $(n+1)$ -οστή θέση, θα είναι κατά 2^n μικρότερο από το πραγματικό άθροισμα. Επομένως, όταν το άθροισμα $A+B \geq 2^n$ κι εμείς κόβουμε το κρατούμενο εξόδου, τότε μας μένει το αποτέλεσμα $(A+B-2^n)$, που μπορούμε να το ερμηνεύσουμε είτε σαν $A + (B-2^n)$, είτε σαν $(A-2^n) + B$. Όμως, οι αριθμοί $(B-2^n)$ ή $(A-2^n)$ είναι αρνητικοί! Άρα, το αποτέλεσμα που βγάλαμε θα είναι το σωστό αποτέλεσμα εάν όταν κάναμε την πρόσθεση θέλαμε στην πραγματικότητα να προσθέσουμε έναν αρνητικό αριθμό, N (negative), και αντί να προσθέσουμε τον N προσθέσαμε τον $N+2^n$, δηλαδή $B = N+2^n$, άρα $N = B-2^n$, άρα το αποτέλεσμα $(A+B-2^n)$ που βγάλαμε είναι $A + (B-2^n) = A + N$, όπου N αρνητικός.

6.3 Κώδικας Συμπληρώματος-2 Προσημασμένων Ακεραίων

Σ' έναν υπολογιστή που λειτουργεί με λέξεις των n bits καθεμία, η συστροφή των ακεραίων αριθμών προκαλεί την επανεμφάνιση της ίδιας αναπαράστασης κάθε φορά που προχωρούμε κατά 2^n προς τα πάνω ή προς τα κάτω. Στο σχήμα δεξιά φαίνεται ένα παράδειγμα για έναν οκτάμπιτο υπολογιστή. Ο κώδικας 11111111, ερμηνευόμενος σαν μη προσημασμένος ακεραίος (unsigned integer), παριστά τον αριθμό 255 (δεκαδικό): όμως, ο ίδιος κώδικας επανεμφανίζεται 256 θέσεις πύ πάνω, στο $255+256 = 511$, όπως και 256 θέσεις πύ κάτω, στο $255-256 = -1$. Η παρατήρηση αυτή είναι η βάση της δημιουργίας του κώδικα αναπαράστασης των προσημασμένων ακεραίων.

Ο κώδικας "συμπλήρωματός ως προς 2" (2's Complement) που χρησιμοποιείται σήμερα στους υπολογιστές για την αναπαράσταση προσημασμένων ακεραίων (signed integers), κωδικοποιεί τον αρνητικό αριθμό $(-A)$, όπου A μεταξύ 1 και 2^{n-1} , με τον κώδικα μη προσημασμένου του ακεραίου $(-A)+2^{n-1} = 2^{n-1}-A$. Έτσι, στο σχήμα δεξιά, ο αριθμός -2 κωδικοποιείται όπως ο $2^{n-1} - 2 = 256 - 2 = 254$, δηλαδή 11111110. Ομοίως, ο -3 κωδικοποιείται όπως ο 253, ο -4 όπως ο 252, ο -5 κωδικοποιείται 11111011, σαν τον 251, κ.ο.κ. Βλέπουμε ότι οι κώδικες των αρνητικών αριθμών αυξάνουν προς την ίδια κατεύθυνση προς την οποία αυξάνουν και οι κώδικες των θετικών αριθμών. Τους θετικούς αριθμούς από 0 έως $2^{n-1}-1$, ο κώδικας συμπλήρωματός ως προς 2 τους παριστά πανομοιότυπα όπως και ο κώδικας των μη προσημασμένων ακεραίων.

Σ' έναν οκτάμπιτο υπολογιστή όπως του παραπάνω παραδείγματος, έχουμε τη δυνατότητα να κωδικοποιήσουμε μέχρι 256 διαφορετικούς ακεραίους αριθμούς. Ποιούς 256 ακεραίους αριθμούς θα διαλέξουμε να κωδικοποιούμε; Η απάντηση εξαρτάται από τον κώδικα που επιλέγουμε. Είδαμε στην §5.1 ότι ο οκτάμπιτος κώδικας μη προσημασμένων ακεραίων παριστάνει τους αριθμούς από το 0 ως το 255· γενικότερα, με n bits, ο κώδικας αυτός παριστάνει τους ακεραίους από το 0 έως και το 2^n-1 . Αντ' αυτού, ο κώδικας συμπλήρωματός ως προς 2, με 8



bits, επιλέγουμε να παριστά μονοσήμαντα τους ακεραίους από -128 έως και $+127$. γενικότερα, με n bits, ο κώδικας αυτός παριστάνει τους ακεραίους από τον -2^{n-1} έως και τον $+2^{n-1}-1$. Τις περιοχές αυτές τις βλέπουμε σημειωμένες στο σχήμα με αγκύλες. Όπως βλέπουμε, η περιοχή αναπαράστασης του κώδικα των προσημασμένων ακεραίων είναι σχεδόν συμμετρική γύρω από το μηδέν, και έχει επιλεγεί ούτως ώστε το **αριστερό (MS) bit** του κώδικα να είναι **1** για όλους τους **αρνητικούς** αριθμούς και μόνο, και να είναι **0** για όλους τους **μη αρνητικούς** αριθμούς και μόνο, δηλαδή για τον αριθμό μηδέν και όλους τους θετικούς αριθμούς.

Ας δούμε τώρα με επίσημο τρόπο τον ορισμό των κωδικοποιήσεων ακεραίων σαν μη προσημασμένους (unsigned) και σαν προσημασμένους (signed) σύμφωνα με τον κώδικα συμπληρώματος ως προς 2. Έστωσαν τα n bits: $a_{n-1} a_{n-2} \dots a_2 a_1 a_0$. Σύμφωνα με τον κώδικα των μη προσημασμένων (unsigned), που είχαμε δει στην § 5.1, αυτά παριστάνουν τον μη αρνητικό ακέραιο αριθμό: $A_u = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$. Τώρα, σύμφωνα με τον κώδικα συμπληρώματος-2 για προσημασμένους (signed) ακέραιους αριθμούς, τα ίδια αυτά n bits παριστάνουν τον εξής αριθμό A_s :

- $A_s = A_u$, όταν $0 \leq A_u \leq 2^{n-1}-1$, δηλαδή όταν το MS bit του A_u (το a_{n-1}) είναι μηδέν (0)
- $A_s = A_u - 2^n$, όταν $2^{n-1} \leq A_u \leq 2^n-1$, δηλαδή όταν το MS bit του A_u (το a_{n-1}) είναι ένα (1)

Από τα παραπάνω προκύπτει επίσης και η εξής ιδιότητα ή εναλλακτικός ορισμός:

$$A_s = -a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

δηλαδή το περισσότερο σημαντικό (MS - most significant) bit, a_{n-1} , έχει συντελεστή αρνητικό, -2^{n-1} , ενώ τα υπόλοιπα bits έχουν τους ίδιους, κλασικούς, θετικούς συντελεστές. Επίσης, μπορούμε να αντιστρέψουμε τις παραπάνω σχέσεις μεταξύ A_s και A_u , ως εξής. Ο προσημασμένος (signed) ακέραιος A_s αναπαρίσταται στον κώδικα συμπληρώματος-2 με τα ίδια bits όπως και ο εξής μη προσημασμένος (unsigned) A_u :

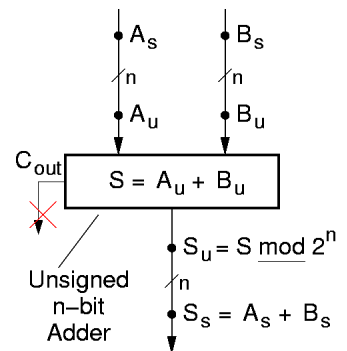
- $A_u = A_s + 2^n$, όταν $-2^{n-1} \leq A_s \leq -1$, δηλαδή όταν το MS bit του A_s (το a_{n-1}) είναι ένα (1)
- $A_u = A_s$, όταν $0 \leq A_s \leq 2^{n-1}-1$, δηλαδή όταν το MS bit του A_s (το a_{n-1}) είναι μηδέν (0)

6.4 Πρόσθεση Προσημασμένων Ακεραίων σε Συμπλήρωμα-2

Θεωρήστε έναν υπολογιστή με λέξεις των n bits, και έναν αθροιστή μη προσημασμένων ακεραίων όπως αυτός της §5.4, του οποίου όμως αγνοούμε το κρατούμενο εξόδου, κρατάμε δηλαδή μόνο το άθροισμα **mod** 2^n , δηλαδή μόνο τα n LS bits. Θα αποδείξουμε ότι εάν στις δύο εισόδους αυτού του αθροιστή τροφοδοτήσουμε τους κώδικες συμπληρώματος-2 δύο προσημασμένων ακεραίων, A_s και B_s , μεταξύ -2^{n-1} και $+2^{n-1}-1$ ο καθένας, και εάν το (προσημασμένο)

άθροισμα των δύο ακεραίων βρίσκεται επίσης στην περιοχή από -2^{n-1} έως και $+2^{n-1}-1$, τότε στην έξοδο του παραπάνω αθροιστή θα εμφανιστεί ο κώδικας συμπληρώματος-2 του (προσημασμένου) αυτού αθροίσματος. Με άλλα λόγια, ο αθροιστής **μη** προσημασμένων ακεραίων, όπως τον ξέρουμε, είναι επίσης και **αθροιστής προσημασμένων αριθμών**, όταν αυτοί παρίστανται με κώδικα συμπληρώματος-2, και όταν το άθροισμά τους χωρά να παρασταθεί και αυτό με τον ίδιο κώδικα και το ίδιο πλήθος bits.

Γιά την απόδειξη θα χρησιμοποιήσουμε, όπως φαίνεται στο παραπάνω σχήμα, τους ακεραίους A_u και B_u , που αποτελούν την ερμηνεία των A_s και B_s ως μη προσημασμένων αριθμών· δηλαδή, όπως ξέρουμε, $A_u = A_s + 2^n$ όταν $A_s < 0$, αλλιώς $A_u = A_s$ --και ομοίως για τον B_u . Ο αθροιστής μη προσημασμένων υπολογίζει το $S_u = (A_u + B_u) \bmod 2^n$, το οποίο στη συνέχεια ερμηνεύουμε σαν τον προσημασμένο αριθμό S_s , δηλ. $S_s = S_u$ όταν $S_u < 2^{n-1}$, αλλιώς $S_s = S_u - 2^n$. Η βασική ιδέα της



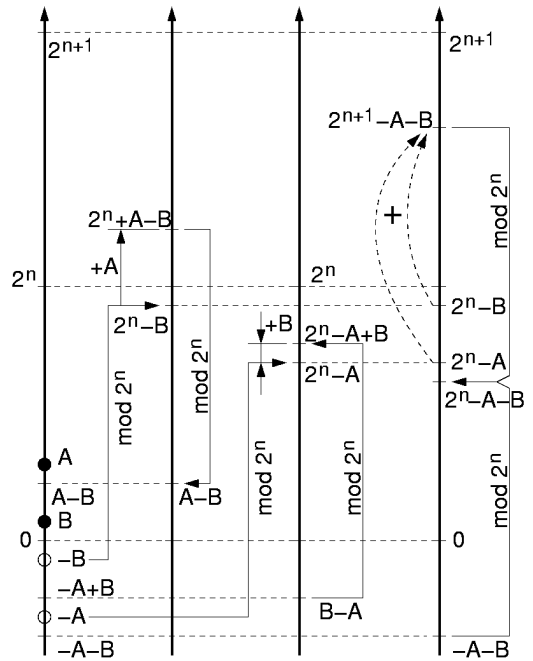
απόδειξης είναι η εξής: αφού ο A_u είναι είτε A_s είτε $A_s + 2^n$, και ο B_u είναι είτε B_s είτε $B_s + 2^n$, τότε το άθροισμα (A_u+B_u) θα είναι: A_s+B_s+P , όπου $P =$ είτε 0 , είτε 2^n , είτε 2^{n+1} . Επομένως, αφού το άθροισμα S_u υπολογίζεται **mod** 2^n , ο όρος P θα φεύγει, και θα μας μένει το άθροισμα A_s+B_s . Για μιά πλήρη απόδειξη, πρέπει να εξετάσουμε προσεκτικά τις περιοχές τιμών των προσθετών και του αθροίσματος, όπως θα κάνουμε τώρα, χωριστά για τις τέσσερις περιπτώσεις:

(α) Θετικός συν Θετικό:

Όταν οι δύο προσθετέοι, A_s και B_s , είναι θετικοί ή μηδέν και δεν ξεπερνούν τον αριθμό $+2^{n-1}-1$, τότε $A_u = A_s$ και $B_u = B_s$, άρα και $A_u+B_u = A_s+B_s$. Η υπόθεση του θεωρήματός μας είναι ότι το άθροισμα A_s+B_s μπορεί να παρασταθεί με n bits σε μορφή συμπληρώματος 2, άρα το A_s+B_s δεν ξεπερνά το $+2^{n-1}-1$. Κατά συνέπεια, και το $S = A_u+B_u$ δεν ξεπερνά το $2^{n-1}-1$, επομένως $S_u = S < 2^{n-1}$. Σε αυτήν την περιοχή, όμως, $S_s = S_u$, άρα $S_s = S = A_u+B_u = A_s+B_s$ [ΟΕΔ].

(β) Θετικός συν Αρνητικό, με Άθροισμα Θετικό ή Μηδέν:

Έστω ότι $A_s = +A$ και $B_s = -B$, όπου A και B είναι θετικοί που δεν ξεπερνούν το $+2^{n-1}-1$, και ο A είναι μεγαλύτερος ή ίσος του B , όπως φαίνεται στο σχήμα (πρώτοι δύο άξονες). Επειδή $A_s > 0$, έχουμε $A_u = A_s = A$. απ' την άλλη μεριά, $B_s < 0$, άρα $B_u = B_s + 2^n = 2^n - B$. Ο μη προσημασμένος αθροιστής υπολογίζει το άθροισμα $S = A + (2^n - B) = 2^n + (A - B)$, το οποίο όμως είναι μεγαλύτερο ή ίσο του 2^n , επειδή ο A είναι μεγαλύτερος ή ίσος του B . Άρα, $S_u = S \bmod 2^n = (2^n + (A - B)) \bmod 2^n = A - B$. Επειδή $S_u = A - B$ δεν ξεπερνά το $+2^{n-1}-1$, θα είναι $S_s = S_u$. επομένως, $S_s = A - B = A_s + B_s$ [ΟΕΔ].



(γ) Θετικός συν Αρνητικό, με Άθροισμα Αρνητικό:

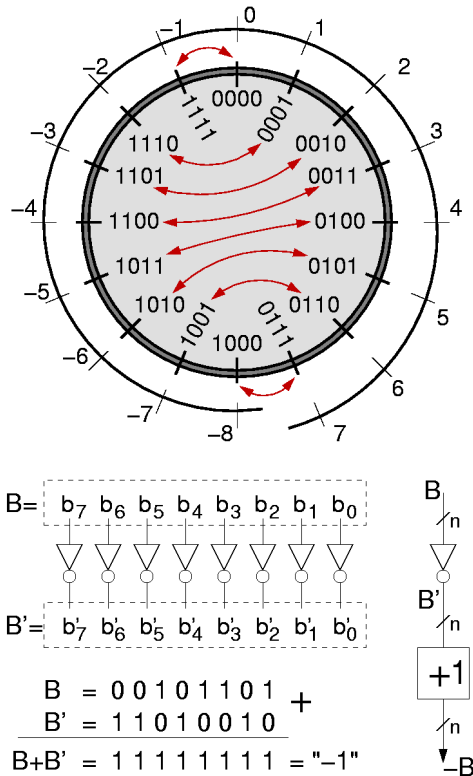
Έστω ότι $A_s = -A$ και $B_s = +B$, όπου A θετικός, B θετικός ή μηδέν, $A > B$, ο A δεν ξεπερνά το 2^{n-1} , και ο B δεν ξεπερνά το $2^{n-1}-1$. Σε αυτή την περίπτωση, έχουμε: $A_u = A_s + 2^n = 2^n - A$, και $B_u = B_s = B$, όπως φαίνεται στο σχήμα, στη μετάβαση από τον πρώτο στον τρίτο άξονα των αριθμών. Ο μη προσημασμένος αθροιστής υπολογίζει το άθροισμα $S = (2^n - A) + B = 2^n - (A - B)$. επειδή $A > B$, το άθροισμα αυτό είναι μικρότερο του 2^n , άρα ο αθροιστής το βγάζει αυτούσιο: $S_u = S = 2^n - (A - B)$. Επειδή ο A δεν ξεπερνά το 2^{n-1} , το ίδιο ισχύει και για τον $(A - B)$. επομένως, ο S_u είναι μεγαλύτερος ή ίσος του 2^{n-1} . Ένας τέτοιος μη προσημασμένος αριθμός, ερμηνευόμενος σε κωδικοποίηση συμπληρώματος-2, θα ερμηνευτεί σαν ο αρνητικός αριθμός $S_s = S_u - 2^n = (2^n - (A - B)) - 2^n = -A + B = A_s + B_s$ [ΟΕΔ].

(δ) Αρνητικός συν Αρνητικό:

Έστω ότι $A_s = -A$ και $B_s = -B$, όπου A και B είναι θετικοί που δεν ξεπερνούν, ούτε αυτοί ούτε το άθροισμά τους, τον αριθμό 2^{n-1} . Τότε: $A_u = A_s + 2^n = 2^n - A$, και $B_u = B_s + 2^n = 2^n - B$, όπως φαίνεται στο δεξιό άξονα του σχήματος. Ο μη προσημασμένος αθροιστής υπολογίζει το άθροισμα $S = (2^n - A) + (2^n - B) = 2^{n+1} - (A + B)$, το οποίο όμως είναι μεταξύ $2^{n+1} - 2^{n-1} = 2^n + 2^{n-1}$ και $2^{n+1} - 1$. Άρα, επειδή ο αθροιστής βγάζει μόνο n bits, θα βγάλει τελικά: $S_u = S \bmod 2^n = (2^{n+1} - (A + B)) \bmod 2^n = 2^n - (A + B)$, το οποίο είναι μεταξύ 2^{n-1} και $2^n - 1$. Σε αυτήν την περιοχή τιμών, $S_s = S_u - 2^n = (2^n - (A + B)) - 2^n = -A - B = A_s + B_s$ [ΟΕΔ].

6.5 Εύρεση του Αντιθέτου ενός Αριθμού

Διαπιστώσαμε ότι μπορούμε να προσθέτουμε προσημασμένους ακέραιους σε μορφή συμπληρώματος ως προς 2 χρησιμοποιώντας τον ίδιο αθροιστή μη προσημασμένων ακεραίων που ήδη είχαμε. Προκειμένου, τώρα, να κάνουμε και αφαιρέσεις, το μόνο που χρειαζόμαστε είναι μία μέθοδος να βρίσκουμε τον αντίθετο αριθμό, $-B$, ενός δοθέντα αριθμού B . Όταν αποκτήσουμε μία τέτοια μέθοδο, θα μπορούμε να κάνουμε την αφαίρεση $A-B$ μέσω της πρόσθεσης $A+(-B)$. Τη ζητούμενη μέθοδο μας τη δίνει η παρατήρηση του σχήματος: Έστω B ένας προσημασμένος ακεραίος σε μορφή συμπληρώματος-2 με n bits, και έστω B' ο επίσης προσημασμένος ακεραίος συμπληρώματος-2 με n bits που προκύπτει από τον B αντιστρέφοντας το κάθε bit του. Τον αριθμό B' τον λέμε και "συμπλήρωμα του B ως προς 1" (*1's complement*), επειδή κάθε νέο bit είναι 1 μείον το παλιό bit. Στο επάνω μέρος του σχήματος φαίνεται ο τροχός των προσημασμένων αριθμών της §6.2 με σημειωμένα επάνω τα ζευγάρια $B - B'$ αριθμών που είναι ο ένας το συμπλήρωμα-ως-προς-1 του άλλου.



Εύκολα βλέπει κανείς ότι το προσημασμένο άθροισμα των $B + B'$ ισούται πάντα με -1 , αφού έχει την αναπαράσταση με όλο άσους: η πρόσθεση ενός bit 0 με ένα bit 1, χωρίς κρατούμενο εισόδου, δίνει πάντα άθροισμα 1 και κρατούμενο 0 (χρησιμοποιήσαμε την ιδιότητα ότι οι προσημασμένοι αριθμοί B και B' προστίθενται με τον ίδιο τρόπο όπως προσθέτουμε και τους μη προσημασμένους αριθμούς, όπως αποδείξαμε στην προηγούμενη παράγραφο, 6.4). Αφού $B+B' = -1$, προκύπτει ότι $B+B'+1 = B + (B'+1) = 0$, άρα οι αριθμοί B και $(B'+1)$ είναι αλγεβρικά αντίθετοι: $(B'+1) = -B$ --υπό την προϋπόθεση ότι το άθροισμα $(B'+1)$ μπορεί να παρασταθεί σε συμπλήρωμα-2 με n bits, δηλαδή ότι το $(B'+1)$ βρίσκεται μεταξύ -2^{n-1} και $+2^{n-1}-1$, πράγμα που ισχύει πάντα εκτός της περίπτωσης $B' = +2^{n-1}-1 = 0111...111$, δηλαδή εκτός $B = 1000...000 = -2^{n-1}$, αφού τότε ο $-B = +2^{n-1}$ δεν μπορεί να παρασταθεί με n bits σε μορφή συμπληρώματος-2.

Επομένως, συνολικά, όταν B είναι ακεραίος στο διάστημα $[-(2^{n-1}-1), +(2^{n-1}-1)]$, τότε ο αλγεβρικός αντίθετός του, $-B$ (που επίσης ανήκει στο ίδιο διάστημα), σε παράσταση συμπληρώματος-2, είναι ο $B'+1$, δηλαδή ο αριθμός που προκύπτει από την παράσταση συμπληρώματος-2 του B αν αντιστρέψουμε το κάθε bit της και στη συνέχεια προσθέσουμε τον αριθμό 1 (μέσω προσημασμένης πρόσθεσης, που, όπως έχουμε δείξει, είναι η ίδια με την μη προσημασμένη όπου αγνοούμε το κρατούμενο εξόδου).

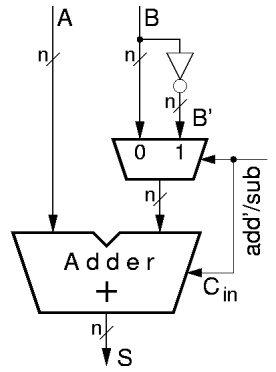
Άσκηση 6.6: Αρνητικοί Αριθμοί και Αφαιρέσεις

- [Κάντε την πριν το εργαστήριο και παραδώστε την με την αναφορά σας.]
- (α) Χρησιμοποιήστε τις παραστάσεις συμπληρώματος-2 του σχήματος της §6.3 για να κάνετε τις παρακάτω προσθέσεις μέσω του γνωστού αλγόριθμου **μη προσημασμένης πρόσθεσης**, με 8 bits και αγνοώντας το κρατούμενο εξόδου, και επιβεβαιώστε στο δεκαδικό ότι το αποτέλεσμα είναι σωστό: $32+(-1)$, $15+(-3)$, $2+(-5)$, $(-1)+(-1)$, $(-4)+(-1)$, $(-127)+126$, $(-127)+127$.
- (β) Χρησιμοποιήστε τις παραστάσεις συμπληρώματος-2 του σχήματος της §6.3 και την παραπάνω μέθοδο εύρεσης του αντίθετου ενός αριθμού, για να βρείτε τα εξής αντίθετα, και επιβεβαιώστε την ορθότητα του αποτελέσματος: (-127) , (-126) , (-5) , (-2) , (-1) , (-0) (-1) , (-2) , (-3) , (-124) , (-126) , και (-127) .

6.7 Προσθαφαιρέτες

Στους υπολογιστές συνήθως βρίσκει κανείς "προσθαφαιρέτες" (adder/subtractor) ακεραίων, κυκλώματα δηλαδή που μπορούν να κάνουν είτε πρόσθεση είτε αφαίρεση. Το κύκλωμα αυτό

κατασκευάζεται πολύ εύκολα και με χαμηλό κόστος, με βάση έναν αθροιστή μη προσημασμένων ακεραίων (§5.4), όπως δείχνει το σχήμα. Ο αθροιστής φαίνεται στο κάτω μέρος του σχήματος: το σύμβολό του είναι ένα τραπέζιο με μία εσοχή, έτσι που να θυμίζει ότι παίρνει δύο λέξεις δεδομένων, τις συνδυάζει, και βγάζει μία λέξη-απάντηση: μέσα στο τραπέζιο γράφεται το σύμβολο "+" για να ξεχωρίζουμε αυτόν τον αθροιστή από άλλα κύκλωμα που κάνουν (και) άλλες πράξεις πάνω στις δύο εισόδους τους. Το κύκλωμα έχει δύο εισόδους δεδομένων, A και B, μία είσοδο ελέγχου, add/sub, και μία έξοδο δεδομένων, S. Συνήθως τα δεδομένα είναι προσημασμένοι αριθμοί, αλλά μπορεί να χρησιμοποιηθεί και με μη προσημασμένους. Η είσοδος A δίδεται στον αθροιστή ως έχει: η είσοδος B, όμως, φτάνει στον αθροιστή με μία από δύο διαφορετικές μορφές, σύμφωνα με το τι επιλέγει ένας πολυπλέκτης 2-σε-1: είτε αυτούσια η λέξη B, είτε το συμπλήρωμά της ως προς 1, B'. Ο βασικός αθροιστής πρέπει να δέχεται κρατούμενο εισόδου, C_{in}. Όταν κάναμε πρόσθεση το κρατούμενο αυτό εισόδου ήταν πάντα 0, και γι' αυτό μπορούσαμε να το αγνοήσουμε χρησιμοποιώντας έναν ημαθροιστή για τα δεξιά (LS) bits, όμως εδώ θα το χρειαστούμε αυτό το κρατούμενο εισόδου, (άρα χρειαζόμαστε πλήρεις αθροιστές στις θέσεις όλων των bits --και του LS). Για να κάνει το κύκλωμα πρόσθεση, S = A+B, θέτουμε την είσοδο ελέγχου add/sub = 0: αυτό κάνει τη δεύτερη είσοδο του αθροιστή να ισούται με B, και το κρατούμενο εισόδου να είναι 0, όπως ακριβώς δηλαδή πρέπει για να γίνει η πρόσθεση A+B. Για να κάνει το κύκλωμα αφαίρεση, S = A-B, θέτουμε την είσοδο ελέγχου add/sub = 1: αυτό κάνει τη δεύτερη είσοδο του αθροιστή να είναι το συμπλήρωμα B', και το κρατούμενο εισόδου να είναι 1. Δεδομένου ότι το κρατούμενο εισόδου έχει την ίδια σημαντικότητα, 2⁰, με τα δεξιά (LS) bits των αριθμών εισόδου, A₀ και B₀, στα οποία και προστίθεται, το να ισούται το κρατούμενο εισόδου αυτό με 1 ισοδυναμεί με το να προστίθεται ο αριθμός 1 μαζί με τους δύο προσθετέους, A και B'. Επομένως, η έξοδος S = A + B' + 1· ξέρουμε όμως, από την §6.5, ότι B'+1 = -B, άρα η έξοδος S = A + (B'+1) = A + (-B) = A - B. Το όνομα του σήματος ελέγχου, add/sub, είναι κατάλληλα γραμμένο ώστε να μας θυμίζει ότι όταν αυτό είναι αληθές (1) το κύκλωμα κάνει "sub" (αφαίρεση), ενώ όταν αυτό είναι ψευδές (0), δηλαδή add' ψευδές άρα add αληθές, τότε το κύκλωμα κάνει "add" (πρόσθεση).



Πείραμα 6.8: Τετράμπιτος Αθροιστής

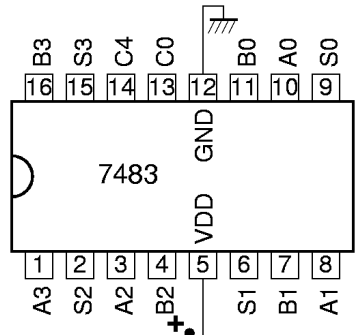
ΠΡΟΣΟΧΗ!: το chip αυτού του πειράματος έχει τα pins τροφοδοσίας σε παράξενη, μη συμβατική θέση. Συνδέστε τα πολύ προσεκτικά, αλλιώς θα κάψετε το chip!

Στο πείραμα αυτό θα χρησιμοποιήσετε το chip "7483" το οποίο είναι ένας τετράμπιτος δυαδικός αθροιστής: περιέχει 4 πλήρεις αθροιστές (του ενός bit καθένας), με τα κρατούμενά τους συνδεδεμένα σε μιά αλυσίδα. Λεπτομερείς πληροφορίες για το chip αυτό μπορείτε να βρείτε στο pdf.datasheetcatalog.com/datasheet/motorola/SN74LS83D.pdf

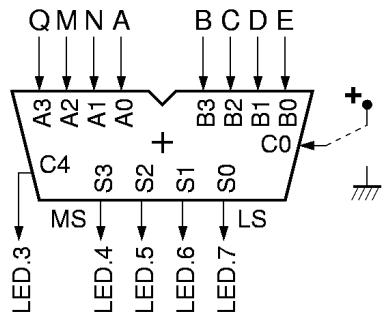
Τροφοδοτήστε τις εισόδους δεδομένων από τους 8 διακόπτες της πλακέτας εισόδων/εξόδων, και συνδέστε 5 LED's στις εξόδους, όπως στο σχήμα. Ελλείπει επαρκών διακοπών εισόδου, συνδέστε την είσοδο κρατουμένου του αθροιστή σε μία από τις τάσεις τροφοδοσίας με ένα σύρμα που μπορείτε να το εναλλάσσετε χειροκίνητα μεταξύ γής (0) και θετικού (1).

Πριν φτάσετε στο εργαστήριο, συμπληρώστε τον εξής πίνακα:

A _{in}	B _{in}	C _{in}	Au+Bu = Su	S (5)	As+B _s = S _s	S (4)
0010	0011	0	2 + 3 = 5	00101	2+3 = 5	0101
0010	1011	0	2 +11 = 13	01101	2-5 = -3	1101
1110	1111	0				
1111	1111	0				
1111	0001	0				
0101	1110	0				
0101	0001	0				



Caution: unconventional GND/VDD!

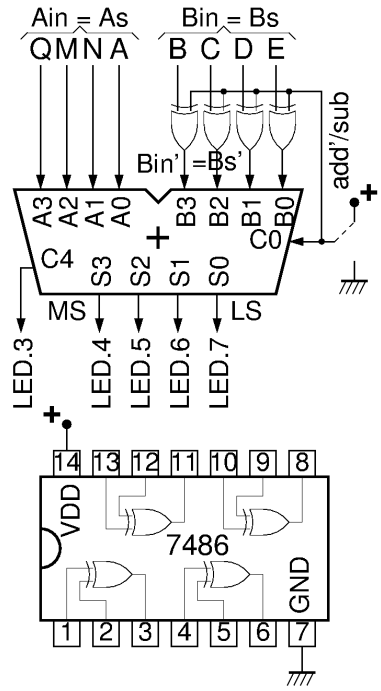


όπου: A_{in}, B_{in}, και C_{in} είναι οι δυαδικές εισόδους του αθροιστή· Au+Bu = Su είναι η ερμηνεία των

εισόδων και της αναμενόμενης εξόδου στο δεκαδικό και σύμφωνα με τον κώδικα μη προσημασμένων αριθμών· $S(5)$ είναι η αναμενόμενη πεντάμπιτη έξοδος του αθροιστή (άθροισμα μη προσημασμένων εισόδων), και πρέπει να συμφωνεί με την προηγούμενη στήλη· $A+B = S$ είναι η ερμηνεία των εισόδων και της αναμενόμενης εξόδου στο δεκαδικό και σύμφωνα με τον κώδικα συμπληρώματος-2 προσημασμένων αριθμών· και $S(4)$ είναι η αναμενόμενη τετράμπιτη έξοδος του αθροιστή για άθροισμα προσημασμένων εισόδων, και πρέπει αφ' ενός να προκύπτει από την στήλη $S(5)$ παραλείποντας το αριστερό (MS) bit, και αφ' ετέρου να συμφωνεί με την προηγούμενη στήλη S . **Στο εργαστήριο**, επαληθεύστε πειραματικά τις εξόδους $S(5)$ και $S(4)$ του πίνακα, ή και άλλες πρόσθετες που εσείς θα δοκιμάσετε. **Μην** χαλάστε το κύκλωμά σας όταν τελειώσετε: θα το χρειαστείτε στο πείραμα 6.9.

Πείραμα 6.9: Προσθαφαιρέτης με XOR

Ο προσθαφαιρέτης της §6.7 μπορεί εναλλακτικά να υλοποιηθεί με πύλες αποκλειστικού-Η (XOR), όπως φαίνεται στο σχήμα δίπλα, αντί αντιστροφέων και πολυπλέκτη. Ο λόγος είναι ότι οι πύλες XOR δίνουν στην έξοδό τους την πρώτη είσοδο όταν η δεύτερη είσοδος είναι 0, ενώ δίνουν στην έξοδό τους το συμπλήρωμα της πρώτης εισόδου τους όταν η δεύτερη είσοδος είναι 1. Φτιάξτε έναν τετράμπιτο προσθαφαιρέτη, χρησιμοποιώντας τον αθροιστή του προηγούμενου πειράματος 6.8 και ένα chip πυλών XOR (7486) όπως αυτό του πειράματος 5.7 για διευκόλυνσή σας, η διάταξη ακροδεκτών του επαναλαμβάνεται στο σχήμα εδώ.



Πριν φτάσετε στο εργαστήριο, κάντε το σχεδιάγραμμα συνδεσμολογίας που δείχνει ποιά συγκεκριμένα pins ποιού chip πρέπει να συνδέσετε πού, και συμπληρώστε τον παρακάτω πίνακα, κατ' αναλογία εκείνου του προηγούμενου πειράματος 6.8. Εδώ, Bin' είναι το συμπλήρωμα (NOT) του Bin · Bu' είναι η ερμηνεία του Bin' σαν unsigned· Su είναι η ερμηνεία του τι θα βγάλει ο αθροιστής σαν unsigned, και πρέπει να συμφωνεί με την επόμενη στήλη, $S(5)$, που πρέπει να είναι το άθροισμα των Ain , Bin' , και $Cin=1$ · και Bs' είναι η ερμηνεία του Bin' σαν signed. **Στο εργαστήριο**, κατασκευάστε τον προσθαφαιρέτη, και δώστε του τις εισόδους Bin του πίνακα, πρώτα με το σήμα add/sub στο 0 (πρόσθεση), οπότε πρέπει να βλέπετε τα αποτελέσματα του πειράματος 6.8, και μετά με το σήμα $add/sub = 1$ (αφαίρεση), ούτως ώστε να διαπιστώσετε πως οι ίδιες πράξεις αυτή τη φορά σαν αφαιρέσεις γίνονται σωστά και δίνουν τα αποτελέσματα του εδώ πίνακα.

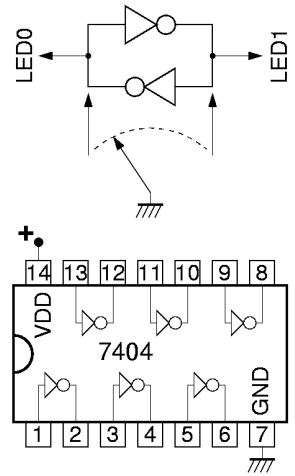
Ain	Bin	$As-Bs = Ss$	Bin'	Cin	$Au+Bu'+1=Su$	$S(5)$	$As+Bs'+1=Ss$	$S(4)$
0010	0011	2-3 = -1	1100	1	2+12+1 =15	01111	2+(-4)+1=-1	1111
0010	1011	2-(-5)= 3	0100	1	2+ 4+1 = 7	00111	2+(+4)+1= 7	0111
1110	1111		0000	1				
1111	1111		0000	1				
1111	0001		1110	1				
0101	1110		0001	1				
0101	0001		1110	1				

Πείραμα 6.10: Η στοιχειώδης Ιδέα του Flip-Flop

Στο πείραμα 3.4 είχαμε δει ότι τα κυκλώματα με θετική ανάδραση έχουν συνήθως δύο σταθερές καταστάσεις, κι έτσι χρησιμοποιούνται σαν μνήμες. Το απλούστερο κύκλωμα λογικών πυλών που έχει θετική ανάδραση είναι δύο αντιστροφείς (πύλες NOT) συνδεδεμένοι κυκλικά ώστε να τροφοδοτούν ο ένας τον άλλον, όπως στο σχήμα. Η ανάδραση είναι θετική επειδή δύο αρνήσεις κάνουν μία κατάφαση. Προφανώς, το κύκλωμα αυτό έχει δύο σταθερές καταστάσεις: (i) ο αριστερός κόμβος μπορεί να είναι 0 και ο δεξιός 1, ή (ii) ο αριστερός κόμβος μπορεί να είναι 1 και ο δεξιός 0. Το κύκλωμα αυτό αποτελεί τη βασική ιδέα του *flip-flop*, δηλαδή του στοιχειώδους κυττάρου ψηφιακής μνήμης.

Κατασκευάστε το κύκλωμα αυτό στο εργαστήριο, και συνδέστε δύο LED's στις εξόδους του.

Χρησιμοποιήστε δύο από τις πύλες ενός chip 7404 (βλ. §3.7). Το κύκλωμα αυτό δεν έχει εξωτερικές εισόδους, αφού κάθε είσοδος πύλης του οδηγείται από μιά άλλη πύλη του! Επομένως, δεν μπορείτε να του αλλάξετε την κατάσταση. Σβήστε και ανάψτε την τροφοδοσία πολλές φορές, και παρατηρήστε αν το κύκλωμα "σηκώνεται" πάντα στην ίδια κατάσταση, ή τότε στη μιά και τότε στην άλλη. (Αν το κύκλωμα είναι εντελώς "συμμετρικό", από ηλεκτρική άποψη, θα "πέφτει" με πιθανότητα 50% στη μιά κατάσταση και 50% στην άλλη, σαν μιά μπίλια που την τοποθετείτε ακριβώς ισορροπημένη στην κόψη ενός ξυραφιού, κι αυτή πέφτει τότε από τη μιά και τότε από την άλλη· αν όμως το κύκλωμα έχει έστω και μία ανεπαίσθητη (ηλεκτρική) ασυμμετρία, τότε θα πέφτει συνήθως από την ίδια "πλευρά").



Ένας --ημιπαράνομος...-- τρόπος να επηρεάσουμε την κατάσταση του flip-flop (με την τροφοδοσία αναμένη) είναι να ακουμπήσουμε "στιγμαία" τον έναν από τους δύο κόμβους του με ένα σύρμα γειωμένο, όπως δείχνει το σχήμα με τα κατακόρυφα βέλη. (Εναλλακτικά, το σύρμα θα μπορούσε να συνδέονταν και στην θετική τροφοδοσία). (Γενικά, σ' ένα τυχόν κύκλωμα, είναι πολύ κακό να ακουμπάς την έξοδο μιάς πύλης στη γή ή στην τροφοδοσία, διότι όταν η έξοδος προσπαθεί να δώσει την αντίθετη τιμή από αυτήν που της ακουμπάμε, κατ' ουσίαν προκαλούμε βραχυκύκλωμα και υπερθέρμανση, με πιθανό κάψιμο αν συνεχιστεί για πολλήν ώρα· εδώ, ευτυχώς, τα πράγματα δεν είναι τόσο κακά: το βραχυκύκλωμα διαρκεί ελάχιστα ns μόνο, μέχρις ότου η νέα τιμή που αυτό επιβάλλει εξωτερικά κάνει την "βόλτα" του κυκλώματος και αλλάξει κατά την ίδια φορά την τιμή που η πύλη προσπαθεί να δώσει στην έξοδό της· έτσι, η υπερθέρμανση είναι πολύ μικρή). Ακουμπήστε τον έναν κόμβο, και δείτε ότι αμέσως η αντίστοιχη LED σβήνει και η άλλη ανάβει. Απομακρύνετε το σύρμα και παρατηρήστε ότι το κύκλωμα μένει εκεί που το αφήσατε, δηλαδή έχει **μνήμη**. Επαναλάβετε από την άλλη πλευρά. Όταν το ακούμπημα είναι από την ίδια πλευρά που είναι ήδη σβηστή, αυτό δεν έχει επίδραση· όταν είναι από την άλλη, τότε έχει. Αυτή είναι η βασική (χειροκίνητη) ιδέα του flip-flop τύπου RS, την αυτοματοποίηση της οποίας (με πύλες αντί ακουμπήματος συρμάτων) θα δούμε στο επόμενο εργαστήριο.

[Up to the Home Page of CS-120](#)

© copyright University of Crete, Greece.
last updated: 12 Nov. 2013, by [M. Katevenis](#).