10.23 Ένας συγχρονιστής χρησιμοποιεί ένα flip-flop με τ_s=54 ps και T_0=21 ps. Εάν υποθέσουμε ότι η είσοδος εναλλάσσεται με συχνότητα 10 MHz και ο χρόνος αποκατάστασης είναι αμελητέος, ποια είναι η ελάχιστη περίοδος ρολογιού για την οποία ο μέσος χρόνος μεταξύ αστοχιών υπερβαίνει τα 100 χρόνια;

10.24 Προσομοιώστε το συγχρονιστή του Σχήματος 10.45 και σχεδιάστε ένα γράφημα ανάλογο με αυτό του Σχήματος 10.43. Από το γράφημα, βρείτε τα Δ_{DQ}, h, τ και T_0.

10.25 Η εταιρεία InferiorCircuits Inc θέλει να σας πουλήσει τον ΤΕΛΕΙΟ συγχρονιστή, ο οποίος, κατά τα λεγόμενά της, δεν παράγει ποτέ μετασταθερή έξοδο. Ο συγχρονιστής αποτελείται από ένα κανονικό flip-flop το οποίο ακολουθείται από ένα συγκριτή μεγάλου κέρδους που παράγει υψηλή έξοδο για εισόδους πάνω από $V_{DD}/4$ και χαμηλή έξοδο για εισόδους κάτω από αυτό το σημείο. Ο αντιπρόεδρος του τμήματος μάρκετινγκ ισχυρίζεται ότι ακόμα κι αν το flip-flop εισέλθει σε κατάσταση μετασταθερότητας, η έξοδός του θα παραμένει κοντά στο $V_{DD}/2$, οπότε ο συγχρονιστής θα παράγει "καλή" υψηλή έξοδο μετά από το συγκριτή. Γιατί δεν θα αγοράζατε αυτό το συγχρονιστή;

11 Υποσυστήματα Χειριστών Δεδομένων

11.1 Εισαγωγή

Γενικά, οι λειτουργικές μονάδες των ολοκληρωμένων μπορούν να ταξινομηθούν στις ακόλουθες κατηγορίες:

- Χειριστές δεδομένων
- Στοιχεία μνήμης
- Δομές ελέγχου
- Κύτταρα ειδικού σκοπού
  - Εισόδου/Εξόδου
  - Διανομής ισχύος
  - Δημιουργίας και διανομής ρολογιού
  - Αναλογικά και RF

Η διαδικασία σχεδίασης συστημάτων CMOS συνίσταται στο διαχωρισμό του σχεδιαζόμενου συστήματος σε υποσυστήματα των παραπάνω κατηγοριών. Υπάρχουν πολλά εναλλακτικά σχήματα τα οποία επιτρέπουν στους σχεδιαστές να κάνουν τους επιθυμητούς ή αναγκαίους συμβιβασμούς μεταξύ ταχύτητας, πυκνότητας, δυνατότητας προγραμματισμού, ευκολίας σχεδίασης και διάφορων άλλων παραμέτρων. Το παρόν κεφάλαιο εξετάζει διάφορες επιλογές σχεδίασης για τους ευρύτερα χρησιμοποιούμενους τελεστές χειριστών δεδομένων. Το επόμενο κεφάλαιο πραγματεύεται τις διατάξεις (arrays), δίνοντας ιδιαίτερη έμφαση σ' αυτές που χρησιμοποιούνται για κυκλώματα μνήμης. Οι δομές ελέγχου κωδικοποιούνται συνήθως με τη χρήση κάποιας γλώσσας περιγραφής hardware και κατόπιν ακολουθεί η σύνθεσή τους. Στο Κεφάλαιο 13 θα εξετάσουμε ορισμένα υποσυστήματα ειδικού σκοπού.

Όπως αναφέραμε στο Κεφάλαιο 1, οι χειριστές δεδομένων ωφελούνται από τις αρχές της ιεραρχίας, της κανονικότητας, της τμηματοποίησης και της τοπικότητας, οι οποίες διέπουν τη δομημένη σχεδίαση συστημάτων. Μπορούν να χρησιμοποιούν N πανομοιότυπα κυκλώματα για την επεξεργασία δεδομένων των N bit. Οι σχετιζόμενοι χειριστές δεδομένων τοποθετούνται σε φυσικά γειτονικές θέσεις, έτσι ώστε να μειώνεται το μήκος των αγωγών διασύνδεσης και η καθυστέρηση. Γενικά, τα δεδομένα διευθετούνται ώστε να ρέουν μόνο προς μία κατεύθυνση, ενώ τα σήματα ελέγχου διευθετούνται ώστε να ρέουν σε κατεύθυνση ορθογώνια προς τη ροή δεδομένων.

Στους κοινούς τελεστές χειριστών δεδομένων που θα εξετάσουμε σ' αυτό το κεφάλαιο περιλαμβάνονται οι αθροιστές, οι ανιχνευτές 0/1, οι συγκριτές, οι μετρητές, οι μονάδες Boolean λογικής, οι μονάδες κώδικα διόρθωσης σφαλμάτων, οι ολισθητές και οι πολλαπλασιαστές.

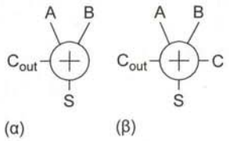
11.2 Πρόσθεση/Αφαίρεση

«Σχεδιάστηκε πλειάδα μηχανισμών και δημιουργήθηκαν σχεδόν ατελείωτα σχεδιαγράμματα, με στόχο την εξοικονόμηση χρόνου και την απλοποίηση του μηχανισμού κρατουμένου».

-Charles Babbage, για την Difference Engine No.1, 1864 [Morrison61]

Η πρόσθεση αποτελεί τη βάση για πολλές λειτουργίες επεξεργασίας – από τις αριθμητικές λογικές μονάδες (ALU) και την παραγωγή διευθύνσεων, έως τον πολλαπλασιασμό και το φιλτράρισμα. Για το λόγο αυτό, τα κυκλώματα αθροιστών που εκτελούν την πρόσθεση δύο δυαδικών αριθμών έχουν ιδιαίτερο ενδιαφέρον για τους σχεδιαστές ψηφιακών συστημάτων. Υπάρχει ευρύτατη γκάμα αρχιτεκτονικών για αθροιστές, η οποία καλύπτει διαφορετικές απαιτήσεις ταχύτητας, κατανάλωσης ισχύος και επιφάνειας. Θα ξεκινήσουμε αυτή την ενότητα περιγράφοντας ημιαθροιστές και πλήρεις αθροιστές για πρόσθεση μεμονωμένων bit. Στη συνέχεια θα εξετάσουμε διάφορους τύπους αθροιστών διάδοσης κρατουμένου (carry-propagate adders, CPA) για την πρόσθεση λέξεων πολλαπλών bit. Τέλος, θα διερευνήσουμε ορισμένες άλλες δομές που σχετίζονται με τις προαναφερθείσες, όπως οι αφαιρέτες και οι αθροιστές πολλαπλών εισόδων.

11.2.1 Πρόσθεση Μεμονωμένων Bit



ΣΧΗΜΑ 11.1 Ημιαθροιστής και πλήρης αθροιστής.

Ο *ημιαθροιστής* (half adder) του Σχήματος 11.1(α) προσθέτει δύο εισόδους του ενός bit, τις A και B. Το αποτέλεσμα είναι 0, 1 ή 2, πράγμα το οποίο σημαίνει ότι απαιτούνται δύο bit για την αναπαράσταση της τιμής: το άθροισμα (S) και το κρατούμενο εξόδου (C_{out}). Το κρατούμενο εξόδου ισοδυναμεί με το κρατούμενο εισόδου της επόμενης σημαντικότερης βαθμίδας ενός αθροιστή πολλών bit, οπότε μπορεί να θεωρηθεί ότι έχει διπλάσια αξία (βαρύτητα) από τα υπόλοιπα bit. Εάν πρόκειται να συνδεθούν εν σειρά πολλοί αθροιστές, ο καθένας θα πρέπει να μπορεί να λαμβάνει ένα κρατούμενο εισόδου. Ένας τέτοιος *πλήρης αθροιστής* (full adder), όπως παρουσιάζεται στο Σχήμα 11.1(β), περιλαμβάνει μια τρίτη είσοδο που αποκαλείται C ή C_{in}.

Οι πίνακες αληθείας για τον ημιαθροιστή και τον πλήρη αθροιστή παρουσιάζονται στους Πίνακες 11.1 και 11.2 αντίστοιχα. Για έναν πλήρη αθροιστή είναι πολύ συχνά χρήσιμο να ορίζονται τα σήματα *Generate (G)*, *Propagate (P)* και *Kill (K)* – δηλαδή, τα σήματα γέννησης, διάδοσης και θανάτωσης κρατουμένου. Ο αθροιστής παράγει («γεννά») ένα κρατούμενο όταν το C_{out} έχει τιμή true (1) ανεξάρτητα από την τιμή του C_{in}, οπότε $G = A \cdot B$. Ο αθροιστής σταματά να διαδίδει («θανατώνει») ένα κρατούμενο όταν το C_{out} είναι false (0) ανεξάρτητα από την τιμή του C_{in}, οπότε $K = \overline{A} \cdot \overline{B} = \overline{A + B}$. Ο αθροιστής διαδίδει ένα κρατούμενο –δηλαδή, παράγει ένα κρατούμενο εξόδου–, εάν λαμβάνει ένα κρατούμενο εισόδου όταν μόνο η μία είσοδος είναι true: $P = A \oplus B$.

ΠΙΝΑΚΑΣ 11.1 Πίνακας αληθείας για τον ημιαθροιστή

A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

ΠΙΝΑΚΑΣ 11.2 Πίνακας αληθείας για τον πλήρη αθροιστή

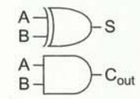
A	B	C	G	P	K	C_{out}	S
0	0	0	0	0	1	0	0
		1				0	1
0	1	0	0	1	0	0	1
		1				1	0
1	0	0	0	1	0	0	1
		1				1	0
1	1	0	1	0	0	1	0
		1				1	1

Από τον πίνακα αληθείας διαπιστώνουμε ότι η λογική του ημιαθροιστή είναι

$$\begin{aligned} S &= A \oplus B \\ C_{out} &= A \cdot B \end{aligned} \qquad \textbf{(11.1)}$$

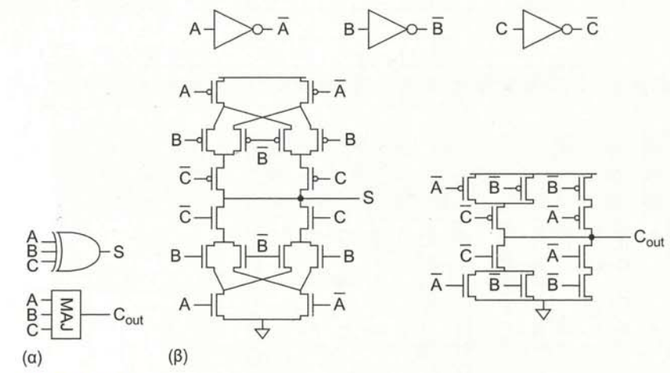
ενώ η λογική του πλήρους αθροιστή είναι

$$\begin{aligned} S &= A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C + ABC \\ &= (A \oplus B) \oplus C = P \oplus C \\ C_{out} &= AB + AC + BC \\ &= AB + C(A + B) \\ &= \overline{A\overline{B} + \overline{C}(\overline{A} + \overline{B})} \\ &= \text{MAJ}(A,\ B,\ C) \end{aligned} \qquad \textbf{(11.2)}$$



ΣΧΗΜΑ 11.2 Σχεδίαση ημιαθροιστή.

Η απλούστερη μέθοδος σχεδίασης ενός αθροιστή χρησιμοποιεί λογικές πύλες. Το Σχήμα 11.2 παρουσιάζει έναν ημιαθροιστή. Το Σχήμα 11.3 παρουσιάζει έναν πλήρη αθροιστή (α) σε επίπεδο λογικών πυλών και (β) σε επίπεδο τρανζίστορ. Η λογική πύλη που παράγει το σήμα του κρατουμένου ονομάζεται επίσης *πύλη πλειοψηφίας* (majority gate) επειδή παράγει 1 εάν τουλάχιστον δύο από τις τρεις εισόδους είναι 1. Δεδομένου ότι οι πλήρεις αθροιστές χρησιμοποιούνται ευρύτερα, θα ασχοληθούμε μ' αυτούς στο υπόλοιπο της παρούσας ενότητας.



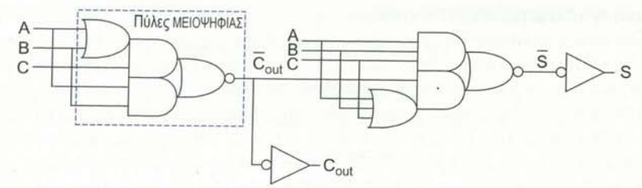
ΣΧΗΜΑ 11.3 Σχεδίαση πλήρους αθροιστή.

Ο πλήρης αθροιστής του Σχήματος 11.3(β) περιλαμβάνει 32 τρανζίστορ (6 για τους αντιστροφείς, 10 για την πύλη πλειοψηφίας και 16 για την πύλη XOR τριών εισόδων). Μια πιο «συμπαγής» σχεδίαση βασίζεται στην παρατήρηση ότι η λογική συνάρτηση του S μπορεί να αναδιατυπωθεί ώστε να επαναχρησιμοποιεί τον όρο C_{out}, ως εξής:

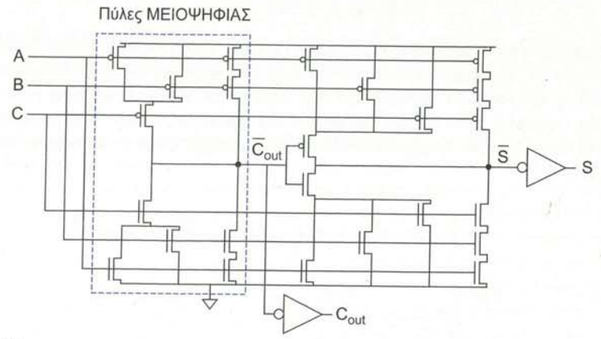
$$S = ABC + (A + B + C)\overline{C}_{out} \qquad \textbf{(11.3)}$$

Μια τέτοια σχεδίαση χρησιμοποιεί μόνο 28 τρανζίστορ και απεικονίζεται (α) σε επίπεδο λογικών πυλών και (β) σε επίπεδο τρανζίστορ στο Σχήμα 11.4. Σημειώστε ότι το δίκτυο των pMOS τρανζίστορ είναι πανομοιότυπο με το δίκτυο των nMOS και όχι *με το* συμπληρωματικό του. Μ' αυτή την απλοποίηση μειώνεται ο αριθμός των εν σειρά τρανζίστορ και το φυσικό σχέδιο γίνεται πιο ομοιόμορφο. Αυτό είναι δυνατό επειδή η πράξη της πρόσθεσης είναι συμμετρική – δηλαδή, η συνάρτηση των συμπληρωματικών εισόδων ισούται με το συμπλήρωμα της συνάρτησης.

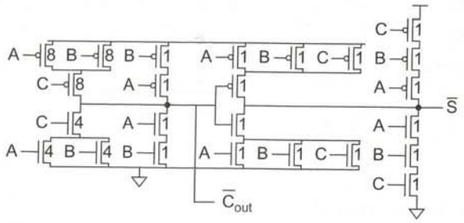
Ο «κατοπτρικός» αθροιστής έχει μεγαλύτερη καθυστέρηση για τον υπολογισμό του S απ' ότι για τον υπολογισμό του C_{out}. Στους αθροιστές διάδοσης κρατουμένου (Ενότητα 11.2.2.1), το κρίσιμο μονοπάτι διανύει τη διαδρομή από το C (C_{in}) έως το C_{out} περνώντας από πολλούς πλήρεις αθροιστές, οπότε η επιπλέον καθυστέρηση για τον υπολογισμό του S δεν είναι σημαντική. Το Σχήμα 11.4(γ) παρουσιάζει το συγκεκρι-



(α)



(β)



(γ)

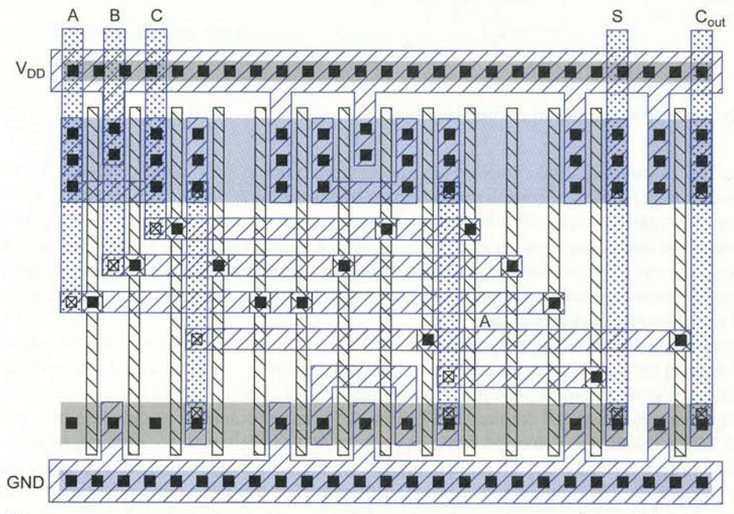
ΣΧΗΜΑ 11.4 Πλήρης αθροιστής κύματος (διάδοσης) κρατουμένου.

μένο αθροιστή, στον οποίο έχουν χρησιμοποιηθεί τρανζίστορ με διαστάσεις βελτιστοποιημένες ώστε να ευνοούν το κρίσιμο μονοπάτι χρησιμοποιώντας τις ακόλουθες τεχνικές:

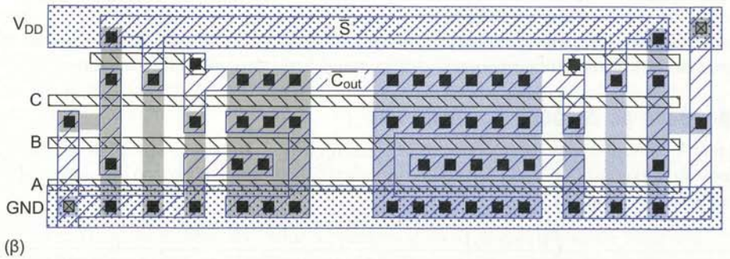
- Τροφοδότηση του σήματος κρατουμένου εισόδου (C) στις εσωτερικές εισόδους, οπότε η εσωτερική χωρητικότητα είναι ήδη εκφορτισμένη.
- Χρήση του ελάχιστου μεγέθους (1 μονάδα, ή 4λ) για όλα τα τρανζίστορ του λογικού κυκλώματος του αθροίσματος, των οποίων τα σήματα πύλης συνδέονται με το κρατουμένου εισόδου και τη λογική του κρατουμένου. Αυτό ελαχιστοποιεί το φόρτο διακλάδωσης στο κρίσιμο μονοπάτι. Η δρομολόγηση αυτού του σήματος πρέπει να έχει κατά το δυνατόν μικρότερο μήκος, για τη μείωση της χωρητικότητας διασύνδεσης.
- Καθορισμός του πλάτους των εν σειρά τρανζίστορ με τη μέθοδο του Λογικού Φόρτου και χρήση προσομοίωσης. Κατασκευή μιας ασύμμετρης πύλης που μειώνει το λογικό φόρτο για τη μετάβαση από το C στο C_{out} εις βάρος του φόρτου μετάβασης στο S.

- Χρήση σχετικά μεγάλων τρανζίστορ στο κρίσιμο μονοπάτι, έτσι ώστε η παρασιτική χωρητικότητα των αγωγών να αποτελεί μικρό μέρος της συνολικής χωρητικότητας.
- Αφαίρεση των αντιστροφέων εξόδου και εναλλαγή θετικής και αρνητικής λογικής για τη μείωση της καθυστέρησης και του πλήθους των τρανζίστορ σε 24 (δείτε την Ενότητα 11.2.2.1).

Το Σχήμα 11.5 παρουσιάζει δύο φυσικά σχέδια αθροιστή. Η επιλογή του λόγου διαστάσεων εξαρτάται από την εφαρμογή. Σε μια μεθοδολογία σχεδίασης με τυποποιημένα κύτταρα, η σχεδίαση του Σχήματος 11.5(α) μπορεί να είναι κατάλληλη όταν χρησιμοποιείται μόνο μία σειρά από nMOS και pMOS τρανζίστορ. Η δρομολόγηση των εισόδων A, B και C παρουσιάζεται μέσα στο κύτταρο, παρότι θα μπορούσε να γίνει εκτός του κυττάρου, επειδή θα πρέπει ούτως ή άλλως να ανατεθούν εξωτερικές γραμμές δρομολόγησης σ' αυτά τα σήματα. Το Σχήμα 11.5(β) παρουσιάζει μια σχεδίαση η οποία θα ήταν κατάλληλη για έναν υψηλής πυκνότητας χειριστή δεδομένων (εφόσον μπορεί να χρησιμοποιηθεί οριζόντιο πολυπυρίτιο). Εδώ τα τρανζίστορ εναλλάσσουν θέσεις κυκλικά και όλη η διασύνδεση γίνεται σε πολυπυρίτιο και μέταλλο1. Αυτό επιτρέπει στις γραμμές διαύλου μετάλλου2 να διατρέχουν το κύτταρο οριζοντίως, χωρίς όμως να έρχονται σε επαφή μ' αυτό. Επιπλέον, το πλάτος των τρανζίστορ μπορεί να αυξηθεί χωρίς αυτό να επηρεάσει το «ύψος ανά bit» του χειριστή δεδομένων. Σ' αυτή την περίπτωση, τα πλάτη επιλέγονται έτσι ώστε να μειώνουν την καθυστέρηση από το C_{in} έως το C_{out} στο κρίσιμο μονοπάτι ενός αθροιστή κύματος [διάδοσης] κρατουμένου.



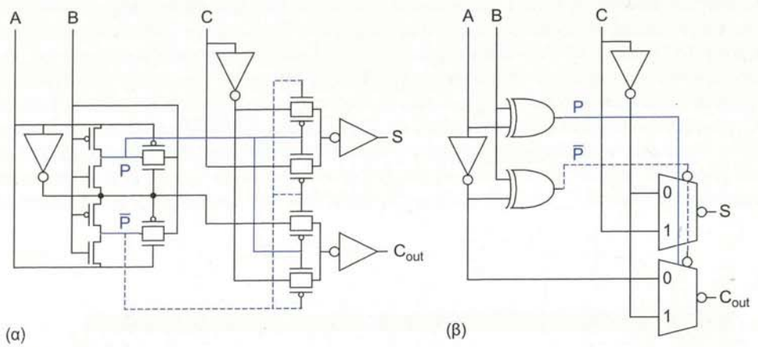
(α)



(β)

ΣΧΗΜΑ 11.5 Φυσικά σχέδια για πλήρεις αθροιστές

Μια διαφορετική σχεδίαση πλήρη αθροιστή χρησιμοποιεί πύλες μετάδοσης για το σχηματισμό των πολυπλεκτών και των πυλών XOR. Το Σχήμα 11.6(α) παρουσιάζει τη σχεδίαση σε επίπεδο τρανζίστορ, η οποία χρησιμοποιεί 24 τρανζίστορ και προβλέπει απομονωμένες εξόδους σωστής πολικότητας και ίσης καθυστέρησης. Η σχεδίαση αυτή θα γίνει καλύτερα κατανοητή εάν αναλύσουμε τις δομές των πυλών μετάδοσης σε ισοδύναμους πολυπλέκτες και μια δομή XOR «ελεγχόμενου αντιστροφέα» (δείτε την Ενότητα 11.7.4), όπως απεικονίζεται στο Σχήμα 11.6(β)[1]. Σημειώστε ότι ο πολυπλέκτης που επιλέγει το S είναι διαμορφωμένος ώστε να υπολογίζει το $P \oplus C$, όπως δίνεται από την Εξ. (11.2).



ΣΧΗΜΑ 11.6 Σχεδίαση πλήρους αθροιστή με πύλες μετάδοσης.

Το Σχήμα 11.7 παρουσιάζει μια διαφορετική προσέγγιση, η οποία χρησιμοποιεί λογική συμπληρωματικών τρανζίστορ περάσματος (complementary pass-transistor logic, CPL). Σε σύγκριση μ' έναν ανεπαρκώς βελτιστοποιημένο στατικό CMOS πλήρη αθροιστή των 40 τρανζίστορ, ο [Yano90] θεωρεί ότι ο CPL αθροιστής είναι δύο φορές ταχύτερος, καταναλώνει 30% λιγότερη ισχύ και είναι ελαφρώς μικρότερος σε μέγεθος. Από την άλλη, σε σύγκριση με μια προσεκτική υλοποίηση του «κατοπτρικού» αθροιστή, ο [Zimmermann97] θεωρεί την καθυστέρηση του CPL ελαφρώς καλύτερη, την κατανάλωση ισχύος συγκρίσιμη και την επιφάνεια πολύ μεγαλύτερη.

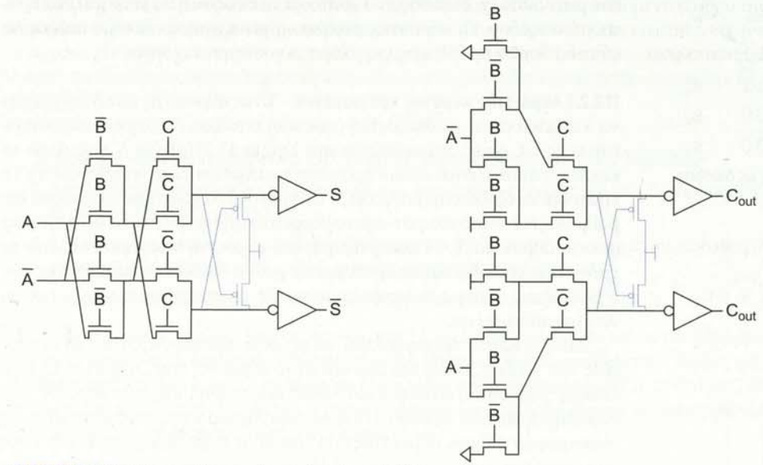
Οι δυναμικοί πλήρεις αθροιστές χρησιμοποιούνται ευρέως σε πολλαπλασιαστές υψηλής ταχύτητας, όταν η κατανάλωση ισχύος δεν αποτελεί πρόβλημα. Επειδή η λογική του αθροίσματος απαιτεί τόσο τις κανονικές εισόδους όσο και τις συμπληρωματικές τους, θα πρέπει να υλοποιηθεί με λογική διαδοχικής επίδρασης, διπλής γραμμής (dual-rail domino). Το Σχήμα 11.8 παρουσιάζει έναν τέτοιο αθροιστή, ο οποίος χρησιμοποιεί XOR/XNOR πύλες domino διπλής γραμμής χωρίς πόδι και πύλες πλειοψηφίας/μειοψηφίας (majority/minority) [Heikes94]. Οι καθυστερήσεις των δύο εξόδων είναι επαρκώς εξισορροπημένες μεταξύ τους, γεγονός ιδιαίτερα σημαντικό για πολλαπλασιαστές όπου και τα δύο μονοπάτια είναι κρίσιμα. Ο αθροιστής αυτός χρησιμοποιεί κοινόχρηστα τρανζίστορ στην πύλη αθροίσματος, με στόχο τη μείωση του αριθμού τους, και εκμεταλλεύεται την ιδιότητα της συμμετρίας ώστε να παρέχει πανομοιότυπες φυσικές διατάξεις για τις δύο πύλες κρατουμένου.

Τυπικά, οι στατικοί CMOS πλήρεις αθροιστές έχουν καθυστέρηση της τάξης των 2-3 αντιστροφέων FO4, ενώ οι αθροιστές domino έχουν καθυστέρηση της τάξης των 1.5 αντιστροφέων FO4 περίπου.

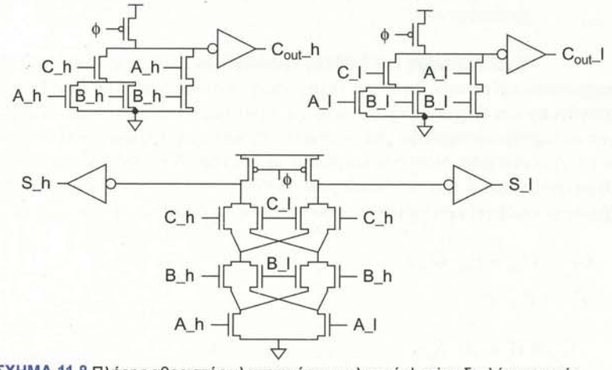
11.2.2 Πρόσθεση με Διάδοση Κρατουμένου

Οι αθροιστές των N bit λαμβάνουν ως εισόδους τα $\{A_N, \ldots, A_1\}$, $\{B_N, \ldots, B_1\}$ και το κρατούμενο εισόδου, C_{in}, και υπολογίζουν το άθροισμα $\{S_N, \ldots, S_1\}$ και το κρατούμενο εξόδου του περισσότερο σημαντικού bit, C_{out}, όπως παρουσιάζεται στο Σχήμα 11.9.

[1] Ορισμένοι προσομοιωτές επιπέδου διακοπτών, όπως ο IRSIM, αντιμετωπίζουν δυσκολίες μ' αυτήν τη δομή XOR και μπορεί να μην την προσομοιώνουν σωστά.

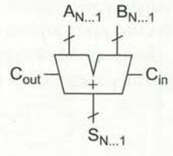


ΣΧΗΜΑ 11.7 Σχεδίαση πλήρους αθροιστή με λογική CPL.

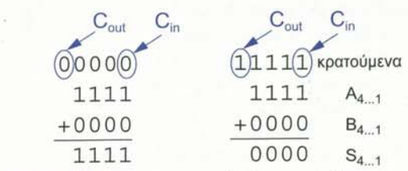


ΣΧΗΜΑ 11.8 Πλήρης αθροιστής υλοποιημένος με λογική domino διπλής γραμμής.

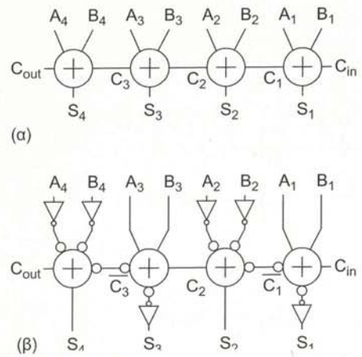
(Κανονικά, σ' αυτό το βιβλίο το λιγότερο σημαντικό bit συμβολίζεται ως A_0 και όχι ως A_1. Ωστόσο, ειδικά για τους αθροιστές, ο συμβολισμός που θα χρησιμοποιήσουμε στις ακόλουθες σελίδες δείχνει πιο «κομψός» εάν η στήλη 0 δεσμεύεται για το χειρισμό του κρατουμένου). Αυτά τα κυκλώματα αποκαλούνται *αθροιστές διάδοσης κρατουμένου* (carry-propagate adders, CPA), επειδή το κρατούμενο εισόδου κάθε βαθμίδας μπορεί να επηρεάσει το κρατούμενο εισόδου όλων των επόμενων βαθμίδων. Για παράδειγμα, το Σχήμα 11.10 παρουσιάζει την πρόσθεση $1111_2 + 0000_2 + 0/1$, στην οποία τα bit του αθροίσματος και του κρατουμένου επηρεάζονται από την τιμή του C_{in}. Η απλούστερη σχεδίαση είναι ο αθροιστής κύματος [διάδοσης] κρατουμένου (carry-ripple adder), στον οποίο το κρατούμενο εξόδου μιας βαθμίδας συνδέεται απλώς με το κρατούμενο εισόδου της επόμενης βαθμίδας. Οι ταχύτεροι αθροιστές διαθέτουν μηχανισμό για την πρόβλεψη του κρατουμένου εξόδου μιας ομάδας πολλαπλών βαθμίδων. Αυτό γίνεται συνήθως με τον υπολογισμό των σημάτων PG ομάδας, τα οποία υποδεικνύουν



ΣΧΗΜΑ 11.9 Αθροιστής διάδοσης κρατουμένου.



ΣΧΗΜΑ 11.10 Παράδειγμα πρόσθεσης με διάδοση κρατουμένου.



ΣΧΗΜΑ 11.11 Αθροιστής κύματος κρατουμένου των 4 bit.

εάν μια ομάδα θα διαδώσει ένα κρατούμενο εισόδου ή θα γεννήσει ένα κρατούμενο εξόδου. Οι αθροιστές μεγάλου μήκους χρησιμοποιούν πολλαπλά επίπεδα δομών πρόβλεψης για ακόμα μεγαλύτερη ταχύτητα.

11.2.2.1 Αθροιστής κύματος κρατουμένου Ένας αθροιστής των N bit μπορεί να κατασκευαστεί με διαδοχική (cascade) σύνδεση N πλήρων αθροιστών του ενός bit, όπως απεικονίζεται στο Σχήμα 11.11(α) για $N = 4$. Αυτό το κύκλωμα αποκαλείται *αθροιστής κύματος κρατουμένου* (carry-ripple adder). Το κρατούμενο εξόδου της βαθμίδας i, C_i, αποτελεί το κρατούμενο εισόδου της βαθμίδας i+1. Αυτό το κρατούμενο θεωρείται ότι έχει διπλάσια αξία (βάρος) από το άθροισμα S_i. Η καθυστέρηση του αθροιστή υπαγορεύεται από το χρόνο που χρειάζονται τα κρατούμενα για να διαδοθούν κατά μήκος των N βαθμίδων, πράγμα το οποίο σημαίνει ότι η καθυστέρηση $t_{C\rightarrow Cout}$ πρέπει να ελαχιστοποιείται.

Αυτή η καθυστέρηση μπορεί να μειωθεί αφαιρώντας τους αντιστροφείς των εξόδων, όπως κάναμε και στον αθροιστή του Σχήματος 11.4(γ). Επειδή η πρόσθεση είναι μια *αυτο-δυϊκή* πράξη (δηλαδή, η συνάρτηση των συμπληρωματικών εισόδων είναι το συμπλήρωμα της συνάρτησης), ένας αναστρέφων πλήρης αθροιστής που λαμβάνει συμπληρωματικές εισόδους δίνει true (μη-αντεστραμμένες) εξόδους. Το Σχήμα 11.11(β) παρουσιάζει έναν αθροιστή κύματος κρατουμένου, δημιουργημένο με αναστρέφοντες πλήρεις αθροιστές. Κάθε δεύτερη βαθμίδα χρησιμοποιεί συμπληρωματικά δεδομένα. Η καθυστέρηση για την αντιστροφή των εισόδων ή των εξόδων του αθροιστή είναι έξω από το κρίσιμο μονοπάτι του αθροιστή κύματος κρατουμένου.

11.2.2.2 Γέννηση και διάδοση κρατουμένου Σ' αυτή την ενότητα θα εισάγουμε μια σημειογραφία, η οποία που χρησιμοποιείται συνήθως για την περιγραφή ταχύτερων αθροιστών. Τα σήματα *διάδοσης* και *γέννησης* κρατουμένου, P (*propagate*) και G (*generate*), αντίστοιχα, ορίστηκαν στην Ενότητα 11.2.1. Μπορούμε να γενικεύσουμε αυτά τα σήματα για να περιγράψουμε εάν μια ομάδα συνεχόμενων βαθμίδων που αντιστοιχούν στα bit $i..j$ (συμπεριλαμβανομένων των i & j) γεννά ή διαδίδει ένα κρατούμενο. Μια ομάδα γεννά ένα κρατούμενο εάν το κρατουμένου εξόδου της έχει τιμή true ανεξάρτητα από το κρατούμενο εισόδου, ενώ διαδίδει ένα κρατούμενο εάν το κρατούμενο εξόδου της έχει τιμή true όταν υπάρχει κρατούμενο εισόδου. Αυτά τα σήματα μπορούν να οριστούν αναδρομικά για $i \geq k > j$

$$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j}$$
$$P_{i:j} = P_{i:k} \cdot P_{k-1:j} \qquad \textbf{(11.4)}$$

με βάση την ακόλουθη υπόθεση

$$G_{i:i} \equiv G_i = A_i \cdot B_i$$
$$P_{i:i} \equiv P_i = A_i \oplus B_i \qquad \textbf{(11.5)}$$

Με άλλα λόγια, μια ομάδα γεννά ένα κρατούμενο εάν το ανώτερο (υψηλότερης σημασίας) τμήμα ή το κατώτερο (χαμηλότερης σημασίας) τμήμα γεννά ένα κρατούμενο και το υψηλότερης σημασίας τμήμα το διαδίδει. Η ομάδα διαδίδει ένα κρατούμενο εάν αμφότερα το υψηλότερης και το χαμηλότερης σημασίας τμήμα διαδίδουν το κρατούμενο[2].

Το κρατούμενο εισόδου απαιτεί ειδικό χειρισμό. Έστω ότι $C_0 = C_{in}$ και $C_N = C_{out}$. Μπορούμε τότε να ορίσουμε τα σήματα γέννησης και διάδοσης κρατουμένου για τη βαθμίδα (bit) 0 ως εξής:

$$G_{0:0} = C_{in}$$
$$P_{0:0} = 0 \qquad \textbf{(11.6)}$$

[2] Εναλλακτικά, ορισμένοι αθροιστές χρησιμοποιούν τον όρο $\overline{K}_i = A_i + B_i$ στη θέση του P_i, επειδή η πύλη OR είναι ταχύτερη από την XOR. Η λογική της ομάδας χρησιμοποιεί τις ίδιες πύλες: $G_{i:j} = G_{i:k} + \overline{K}_{i:k} \cdot G_{k-1:j}$ και $\overline{K}_{i:j} = \overline{K}_{i:k} \cdot \overline{K}_{k-1:j}$. Ωστόσο, απαιτείται και πάλι το $P_i = A_i \oplus B_i$ στην Εξ. (11.7) για τον υπολογισμό του τελικού αθροίσματος. Σε ορισμένες περιπτώσεις μετονομάζεται σε X_i ή T_i για την αποφυγή ασαφειών.

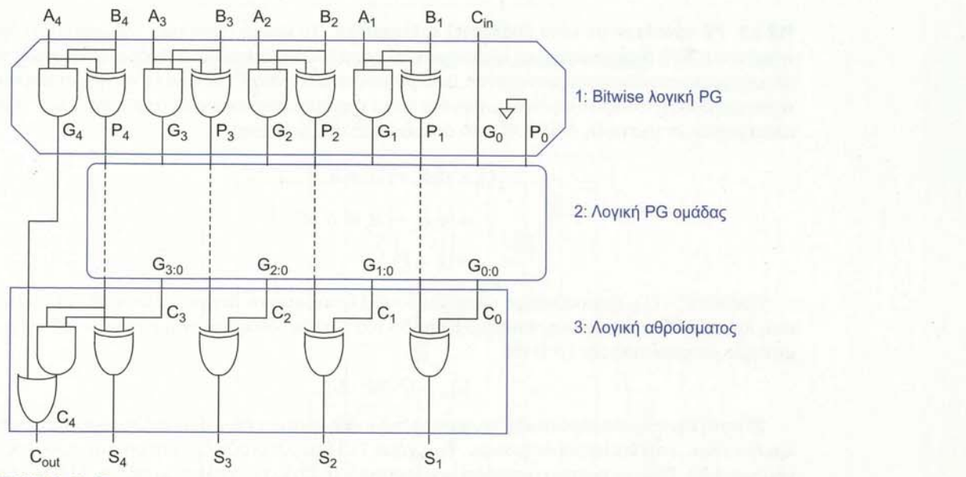
Παρατηρήστε ότι το κρατούμενο εισόδου της βαθμίδας i είναι το κρατούμενο εξόδου της βαθμίδας i-1 και ισχύει $C_{i-1} = G_{i-1:0}$. Αυτή είναι μια πολύ σημαντική σχέση· στις ακόλουθες ενότητες, θα αντιμετωπίζουμε τα σήματα γέννησης κρατουμένου ομάδας (group generate) και τα κρατούμενα ως ταυτόσημα. Μπορούμε, λοιπόν, να υπολογίσουμε το σήμα αθροίσματος για τη βαθμίδα i χρησιμοποιώντας την Εξ. (11.2):

$$S_i = P_i \oplus G_{i-1:0} \qquad \textbf{(11.7)}$$

Συνεπώς, η πρόσθεση μπορεί να απλοποιηθεί σε μια διαδικασία τριών βημάτων:

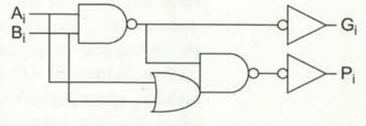
1. Υπολογισμός, σε επίπεδο bit (bitwise), των σημάτων γέννησης (G) και διάδοσης (P) κρατουμένου μέσω των Εξ. (11.5) και (11.6).
2. Συνδυασμός αυτών των σημάτων (PG) για τον καθορισμό των σημάτων γέννησης κρατουμένου ομάδας $G_{i-1:0}$ για όλα τα $N \geq i \geq 1$, χρησιμοποιώντας την Εξ. (11.4).
3. Υπολογισμός των σημάτων αθροίσματος χρησιμοποιώντας την Εξ. (11.7).

Αυτά τα βήματα απεικονίζονται στο Σχήμα 11.12. Επειδή το πρώτο και τρίτο βήμα είναι απλά, στο υπόλοιπο αυτής της ενότητας θα εστιάσουμε κατά κύριο λόγο στις εναλλακτικές λύσεις για τον υπολογισμό των σημάτων λογικής PG ομάδας, με διάφορους συμβιβασμούς μεταξύ ταχύτητας, επιφάνειας και πολυπλοκότητας. Μέρος του hardware μπορεί να είναι κοινόχρηστο στα κυκλώματα της bitwise (επιπέδου bit) λογικής PG, όπως απεικονίζεται στο Σχήμα 11.13.



ΣΧΗΜΑ 11.12 Πρόσθεση με λογική γέννησης και διάδοσης κρατουμένου (PG).

Στη βιβλιογραφία χρησιμοποιούνται πολλοί συμβολισμοί για την περιγραφή της λογικής PG ομάδας (group PG logic). Γενικά, η λογική PG είναι ένα παράδειγμα *προθεματικού* (prefix) υπολογισμού [Leighton92]. Δέχεται ως εισόδους τα $\{P_{N:N}, \ldots, P_{0:0}\}$ και $\{G_{N:N}, \ldots, G_{0:0}\}$ και υπολογίζει τα *προθέματα* $\{G_{N:0}, \ldots, G_{0:0}\}$ χρησιμοποιώντας την Εξ. (11.4). Στη βιβλιογραφία υπάρχουν διάφορες ονομασίες γι' αυτή την εξίσωση, όπως *τελεστής δέλτα*, *θεμελιώδης τελεστής κρατουμένου* και *τελεστής προθέματος*. Πολλά άλλα προβλήματα, όπως η κωδικοποίηση με προτεραιότητα, μπορούν να διατυπωθούν ως προθεματικοί υπολογισμοί, και μπορούν να εφαρμοστούν όλες οι τεχνικές που χρησιμοποιούνται για την υλοποίηση γρήγορης λογικής PG ομάδας, όπως θα δούμε στην Ενότητα 11.10.



ΣΧΗΜΑ 11.13 Κοινή χρήση bitwise λογικής PG.

Η Εξ. (11.4) ορίζει λογική PG ομάδας *2ης τάξης*, επειδή συνδυάζει ζεύγη μικρότερων ομάδων. Είναι επίσης δυνατό να οριστεί λογική ομάδας υψηλότερης τάξης, με στόχο τη χρησιμοποίηση λιγότερων σταδίων με πολυπλοκότερες πύλες [Beaumont-Smith99], όπως υποδεικνύει η Εξ. (11.8) και το Σχήμα 11.16(γ). Για παράδειγμα, στη λογική ομάδας 4ης τάξης, μια ομάδα διαδίδει το κρατούμενο εάν διαδίδουν κρατούμενο και τα τέσσερα τμήματα. Μια ομάδα γεννά ένα κρατούμενο εάν το ανώτερο (υψηλότερης σημασίας) τμήμα γεννά κρατούμενο, ή εάν το δεύτερο τμήμα γεννά και το υψηλότερης σημασίας διαδίδει κρατούμενο, ή εάν το τρίτο τμήμα γεννά και τα δύο υψηλότερης σημασίας διαδίδουν κρατούμενο, ή εάν το χαμηλότερης σημασίας τμήμα γεννά και τα τρία υψηλότερης σημασίας διαδίδουν κρατούμενο.

$$\left.\begin{aligned} G_{i:j} &= G_{i:k} + P_{i:k}\cdot G_{k-1:l} + P_{i:k}\cdot P_{k-1:l}\cdot G_{l-1:m} + P_{i:k}\cdot P_{k-1:l}\cdot P_{l-1:m}\cdot G_{m-1:j} \\ &= G_{i:k} + P_{i:k}\left(G_{k-1:l} + P_{k-1:l}\left(G_{l-1:m} + P_{l-1:m}G_{m-1:j}\right)\right) \\ P_{i:j} &= P_{i:k}\cdot P_{k-1:l}\cdot P_{l-1:m}\cdot P_{m-1:j} \end{aligned}\right\} \quad (i \geq k > l > m > j) \qquad \textbf{(11.8)}$$

Η μέθοδος του Λογικού Φόρτου υποδεικνύει ότι ο βέλτιστος φόρτος σταδίου είναι περίπου 4. Συνεπώς, δεν είναι κατ' ανάγκην καλύτερη η κατασκευή λιγότερων σταδίων με πύλες μεγαλύτερης τάξης. Θα πρέπει να εκτελούνται κατάλληλες προσομοιώσεις και υπολογισμοί, για τη σωστή σύγκριση και επιλογή μεταξύ των εναλλακτικών λύσεων για μια δεδομένη τεχνολογία και οικογένεια κυκλωμάτων.

11.2.2.3 PG πρόσθεση με κύμα (διάδοσης) κρατουμένου Το κρίσιμο μονοπάτι του αθροιστή κύματος κρατουμένου (CRA) διέρχεται από το κρατούμενο εισόδου έως το κρατούμενο εξόδου, κατά μήκος των πυλών πλειοψηφίας της αλυσίδας κρατουμένου. Δεδομένου ότι τα σήματα P και G θα έχουν ήδη σταθεροποιηθεί έως τη στιγμή άφιξης του κρατουμένου, μπορούμε να τα χρησιμοποιήσουμε για να μετατρέψουμε τη συνάρτηση πλειοψηφίας σε μια πύλη AND-OR[3], για απλοποίηση της σχεδίασης:

$$\begin{aligned} C_i &= A_iB_i + (A_i + B_i)C_{i-1} \\ &= A_iB_i + (A_i \oplus B_i)C_{i-1} \\ &= G_i + P_iC_{i-1} \end{aligned} \qquad \textbf{(11.9)}$$

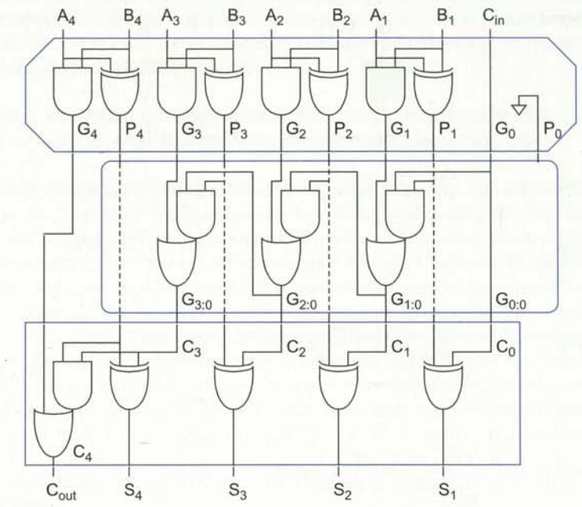
Εφόσον $C_i = G_{i:0}$, η πρόσθεση με κύμα (διάδοσης) κρατουμένου μπορεί να θεωρηθεί ακραία περίπτωση μιας λογικής PG ομάδας, στην οποία μια ομάδα του 1 bit συνδυάζεται με μια ομάδα των i bit για το σχηματισμό μιας ομάδας των $(i+1)$ bit:

$$G_{i:0} = G_i + P_i \cdot G_{i-1:0} \qquad \textbf{(11.10)}$$

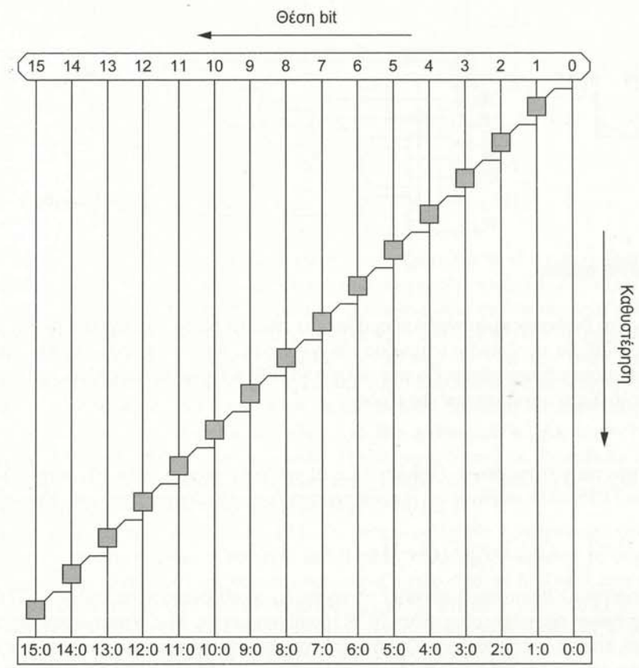
Σ' αυτή την ακραία περίπτωση, τα σήματα διάδοσης κρατουμένου ομάδας δεν χρησιμοποιούνται ποτέ και δεν είναι ανάγκη να υπολογιστούν. Το Σχήμα 11.14 παρουσιάζει έναν αθροιστή διάδοσης κρατουμένου των 4 bit. Εδώ, το κρίσιμο μονοπάτι κρατουμένου διασχίζει μια αλυσίδα πυλών AND-OR και όχι μια αλυσίδα πυλών πλειοψηφίας. Το Σχήμα 11.15 παρουσιάζει τη λογική PG ομάδας για ένα αθροιστή κύματος κρατουμένου των 16 bit, όπου οι πύλες AND-OR στο δίκτυο PG επισημαίνονται με γκρι κύτταρα.

Ανάλογα διαγράμματα θα χρησιμοποιηθούν για τη σύγκριση διαφόρων αρχιτεκτονικών αθροιστών σε επόμενες ενότητες. Αυτά τα διαγράμματα χρησιμοποιούν κύτταρα μαύρου και γκρι χρώματος και λευκούς απομονωτές, όπως ορίζονται στο Σχήμα 11.16(α), για κύτταρα 2ης τάξης. Τα μαύρα κύτταρα εμπεριέχουν τη λογική γέννησης και διάδοσης κρατουμένου ομάδας (μία πύλη AND-OR και μια πύλη AND), όπως ορίζονται στην Εξ. (11.4). Τα γκρι κύτταρα, τα οποία περιέχουν μόνο τη λογική γέννησης κρατουμένου ομάδας, χρησιμοποιούνται στη θέση του τελευταίου κυττάρου σε κάθε στήλη, επειδή απαιτείται μόνο το σήμα γέννησης κρατουμένου ομάδας για τον υπολογισμό των αθροισμάτων. Για την ελαχιστοποίηση του φόρτου στο κρίσιμο μονοπάτι, μπορούν να χρησιμοποιηθούν απομονωτές. Κάθε γραμμή αναπαριστά μια

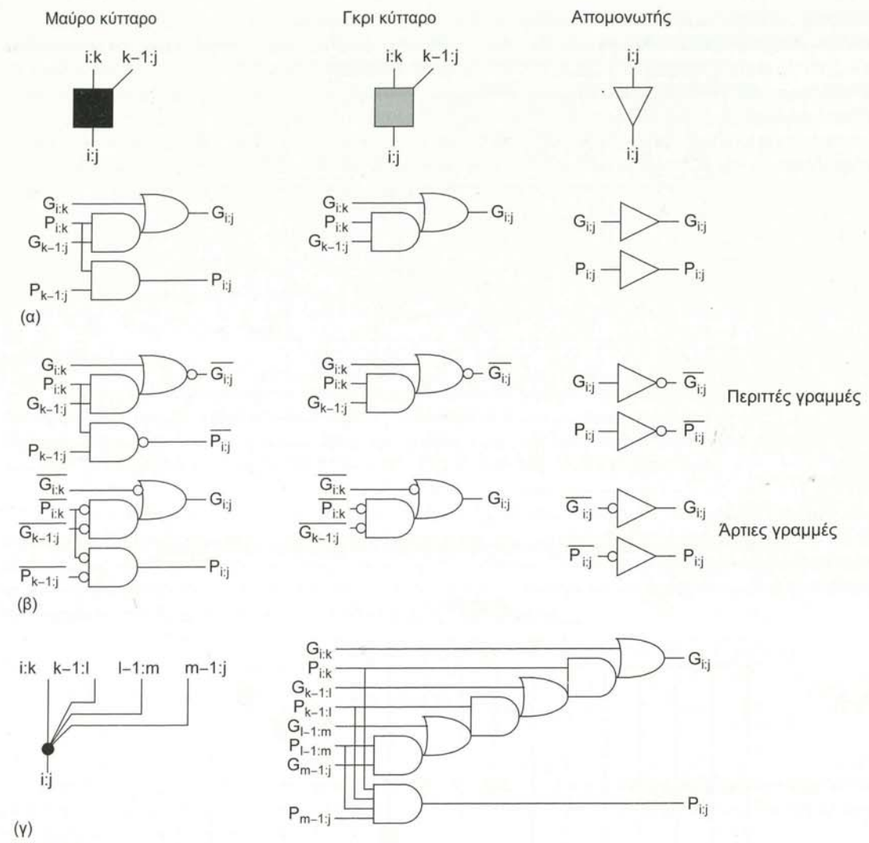
[3] Οποτεδήποτε περιγράφεται μια δομή θετικής λογικής, όπως η AND-OR, είναι επίσης δυνατό να χρησιμοποιηθεί μια πύλη AOI και βαθμίδες εναλλασσόμενης πολικότητας, όπως στο Σχήμα 11.11(β), για εξοικονόμηση επιφάνειας και καθυστέρησης.



ΣΧΗΜΑ 11.14 4-bit αθροιστής κύματος κρατουμένου που χρησιμοποιεί PG λογική.



ΣΧΗΜΑ 11.15 Δίκτυο PG ομάδας ενός αθροιστή κύματος κρατουμένου.



ΣΧΗΜΑ 11.16 Κύτταρα PG ομάδας.

δέσμη σημάτων γέννησης και διάδοσης κρατουμένου ομάδας (τα σήματα διάδοσης παραλείπονται μετά από τα γκρι κύτταρα). Οι πύλες XOR για την bitwise (επιπέδου bit) λογική PG και το άθροισμα έχουν αφαιρεθεί από το πρώτο και τελευταίο πλαίσιο και υποτίθεται ότι μια πύλη AND-OR λειτουργεί παράλληλα με τις πύλες XOR αθροίσματος για τον υπολογισμό του κρατουμένου εξόδου:

$$C_{\text{out}} = G_{N:0} = G_N + P_N G_{N-1:0} \tag{11.11}$$

Τα κύτταρα διευθετούνται κατακόρυφα, σύμφωνα με τη χρονική στιγμή κατά την οποία λειτουργούν [Guyot97]. Από το Σχήμα 11.15, είναι εμφανές ότι η καθυστέρηση κρίσιμου μονοπατιού του αθροιστή διάδοσης κρατουμένου είναι

$$t_{\text{ripple}} = t_{pg} + (N-1)t_{AO} + t_{\text{xor}} \tag{11.12}$$

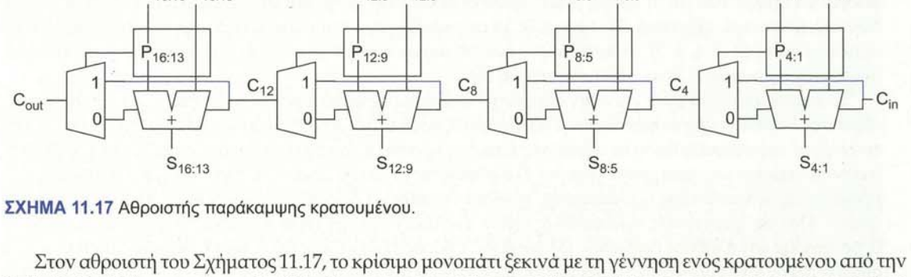
όπου t_{pg} είναι η καθυστέρηση των 1-bit πυλών διάδοσης/γέννησης, t_{AO} η καθυστέρηση της πύλης AND-OR στο γκρι κύτταρο και t_{XOR} η καθυστέρηση της τελευταίας πύλης XOR του αθροίσματος. Μια τέτοια εκτίμηση της καθυστέρησης είναι μόνο ποιοτική, επειδή δεν συνυπολογίζει την οδηγητική ικανότητα εξόδου (fanout) ή τις διαστάσεις.

Συχνά, η χρήση πυλών χωρίς αντιστροφή οδηγεί σε περισσότερα στάδια λογικής απ' όσα χρειάζονται. Το Σχήμα 11.16(β) παρουσιάζει την εναλλάξ χρήση δύο τύπων σταδίων με αντιστροφή σε εναλλασσόμενες σειρές

του δικτύου PG ομάδας, για την εξάλειψη των εξωτερικών αντιστροφέων. Για καλύτερη απόδοση, το $G_{k-1:j}$ θα πρέπει να οδηγεί το εσωτερικό τρανζίστορ του σωρού των εν σειρά τρανζίστορ. Είναι, επίσης, δυνατή η μείωση του πλήθους των σταδίων με τη χρήση κυττάρων μεγαλύτερης τάξης, όπως υποδεικνύει το Σχήμα 11.16(γ) για ένα μαύρο κύτταρο 4ης τάξης.

11.2.2.4 Αθροιστής Manchester αλυσίδας κρατουμένου *Αυτή η ενότητα περιλαμβάνεται στην ύλη που είναι διαθέσιμη online, μέσω του συνδέσμου «Web Enhanced», στον ιστότοπο* `www.cmosvlsi.com`.

11.2.2.5 Αθροιστής παράκαμψης κρατουμένου Το κρίσιμο μονοπάτι των αθροιστών διάδοσης κρατουμένου που εξετάσαμε έως τώρα περιλαμβάνει μια πύλη ή ένα τρανζίστορ για κάθε βαθμίδα (bit) του αθροιστή, πράγμα το οποίο μπορεί να είναι αργό, κυρίως σε μεγάλους αθροιστές. Ο αθροιστής *παράκαμψης κρατουμένου (carry-skip, carry-bypass)*, ο οποίος προτάθηκε αρχικά από τον Charles Babbage τον 19ο αιώνα και χρησιμοποιήθηκε για πολλά χρόνια σε μηχανικά συστήματα υπολογισμών, συντομεύει το κρίσιμο μονοπάτι με την εξής τεχνική: υπολογίζονται τα σήματα διάδοσης κρατουμένου ομάδας για κάθε αλυσίδα κρατουμένου και χρησιμοποιούνται για την παράκαμψη των μεγάλου μήκους διαδρομών διάδοσης κρατουμένου [Morgan59, Lehman61]. Το Σχήμα 11.17 παρουσιάζει έναν αθροιστή παράκαμψης κρατουμένου κατασκευασμένο με τέσσερις ομάδες. Οι μονάδες που αντιπροσωπεύουν τα ορθογώνια εκτελούν τους λογικούς υπολογισμούς που απαιτούνται για τα σήματα διάδοσης και γέννησης κρατουμένου (όμοια με το κύκλωμα του Σχήματος 11.15), ενώ περιλαμβάνεται επίσης μια πύλη AND 4 εισόδων για το σήμα διάδοσης κρατουμένου της ομάδας 4 bit. Ο πολυπλέκτης παράκαμψης επιλέγει το κρατούμενο εισόδου της ομάδας εάν το σήμα διάδοσης κρατουμένου της ομάδας αυτής έχει τιμή true· διαφορετικά, επιλέγει το κρατούμενο εξόδου του αθροιστή διάδοσης κρατουμένου.

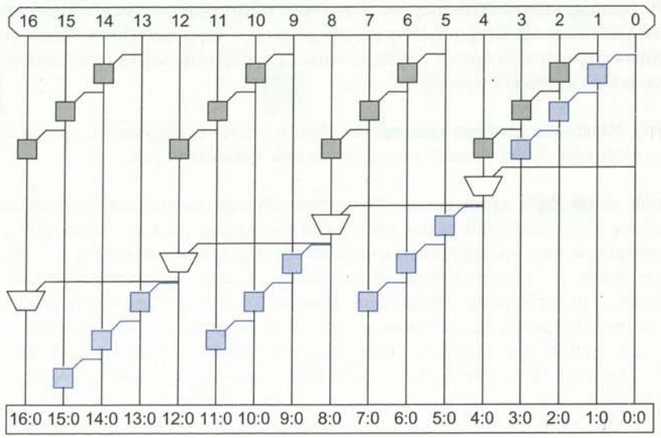


ΣΧΗΜΑ 11.17 Αθροιστής παράκαμψης κρατουμένου.

Στον αθροιστή του Σχήματος 11.17, το κρίσιμο μονοπάτι ξεκινά με τη γέννηση ενός κρατουμένου από την 1η βαθμίδα (bit), το οποίο και διαδίδει διαμέσου του υπόλοιπου αθροιστή. Το κρατούμενο πρέπει να περάσει από τα τρία επόμενα bit, αλλά στη συνέχεια μπορεί να παρακάμψει τα δύο επόμενα μπλοκ των 4 bit (συνολικά οκτώ bit). Τέλος, θα πρέπει να περάσει από το τελευταίο μπλοκ 4 bit, για τη δημιουργία των αθροισμάτων. Αυτό απεικονίζεται στο Σχήμα 11.18. Οι αλυσίδες διάδοσης των 4 bit στην επάνω πλευρά του διαγράμματος καθορίζουν εάν κάθε ομάδα γεννά ένα κρατούμενο. Η αλυσίδα παράκαμψης κρατουμένου στο μέσον του διαγράμματος παρακάμπτει τα 4-bit μπλοκ. Τέλος, οι 4-bit αλυσίδες διάδοσης που επισημαίνονται με τις μπλε γραμμές αναπαριστούν *τους ίδιους αθροιστές*, οι οποίοι μπορούν να παράγουν ένα κρατούμενο εξόδου όταν ένα κρατούμενο εισόδου στέλνεται μέσω παράκαμψης σ' αυτούς. Σημειώστε ότι η τελευταία πύλη AND-OR και η στήλη 16 δεν είναι απολύτως αναγκαία, επειδή το C_{out} μπορεί να υπολογίζεται παράλληλα με τις πύλες XOR του αθροίσματος, χρησιμοποιώντας την Εξ. (11.11).

Στον αθροιστή των Σχημάτων 11.17 & 11.18, το κρίσιμο μονοπάτι χρησιμοποιεί την αρχική λογική PG που παράγει ένα κρατούμενο από τη βαθμίδα (bit) 1, τρεις πύλες AND-OR που το διαδίδουν έως το bit 4, τρεις πολυπλέκτες που το περνούν, με παράκαμψη, στο C_{12}, τρεις πύλες AND-OR που το διαδίδουν έως το bit 15 και μια τελευταία πύλη XOR για την παραγωγή του S_{16}. Επειδή ο πολυπλέκτης αντιπροσωπεύει μια συνάρτηση AND22-OR, είναι ελαφρώς αργότερος απ' ό,τι η συνάρτηση AND-OR. Γενικά, ένας αθροιστής των N bit που χρησιμοποιεί k ομάδες των n bit ($N = n \times k$) έχει καθυστέρηση

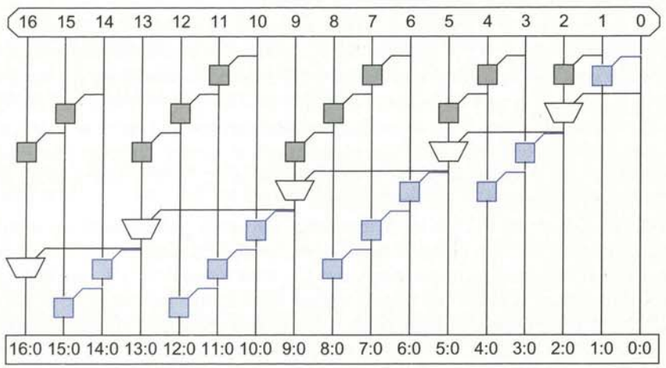
$$t_{\text{skip}} = t_{pg} + 2(n-1)t_{AO} + (k-1)t_{\text{mux}} + t_{\text{xor}} \tag{11.13}$$



ΣΧΗΜΑ 11.18 Δίκτυο PG ομάδων του αθροιστή παράκαμψης κρατουμένου.

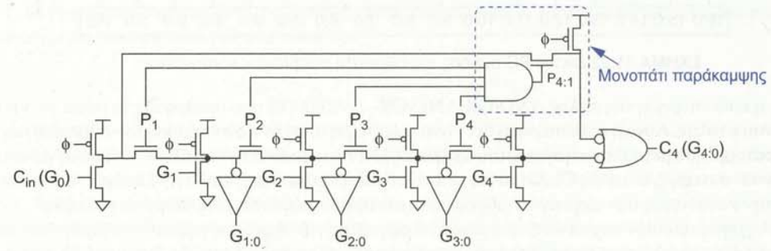
Αυτό το κρίσιμο μονοπάτι εξαρτάται από το μήκος της πρώτης και της τελευταίας ομάδας αλλά και από των αριθμό των ομάδων. Στις βαθμίδες του δικτύου που αντιπροσωπεύουν τα πιο σημαντικά bit, τα αποτελέσματα της διάδοσης κρατουμένου είναι διαθέσιμα αρκετά νωρίς. Συνεπώς, το κρίσιμο μονοπάτι μπορεί να σμικρυνθεί με τη χρήση μικρότερων ομάδων στην αρχή και στο τέλος και μεγαλύτερων ομάδων στο μέσον του αθροιστή. Το Σχήμα 11.19 απεικονίζει ένα τέτοιο δίκτυο PG, το οποίο χρησιμοποιεί ομάδες μήκους [2, 3, 4, 4, 3], σε αντίθεση μ' ένα δίκτυο με ομάδες μήκους [4, 4, 4, 4] το οποίο αποθηκεύει δύο επίπεδα λογικής σ' έναν αθροιστή των 16 bit.

Το κόστος του hardware για έναν αθροιστή παράκαμψης κρατουμένου είναι ίδιο μ' αυτό ενός απλού αθροιστή διάδοσης κρατουμένου συν k πολυπλέκτες και k πύλες AND των n εισόδων. Αποτελεί ελκυστική επιλογή σε περιπτώσεις όπου οι αθροιστές κύματος κρατουμένου είναι πολύ αργοί, αλλά το κόστος του hardware πρέπει να παραμείνει χαμηλό. Για αθροιστές μεγάλου μήκους, μπορεί να χρησιμοποιηθεί μια προσέγγιση πολυεπίπεδης παράκαμψης, η οποία θα παρέχει δυνατότητα «παράκαμψης των παρακάμψεων». Πολλές ερευνητικές προσπάθειες έχουν αναλωθεί για την εύρεση του καλύτερου δυνατού μεγέθους ομάδας και πλήθους βαθμίδων [Majerski67, Oklobdžija85, Guyot87, Chan90, Kantabutra91], αν και σήμερα πλέον χρησιμοποιούνται αθροιστές παράλληλου προθέματος αντί των αθροιστών μεγάλου μήκους.



ΣΧΗΜΑ 11.19 Δίκτυο PG ομάδων μεταβλητού μεγέθους του αθροιστή παράκαμψης κρατουμένου.

Στα κυκλώματα των Σχημάτων 11.17 και 11.18, θα ήταν δελεαστικό να αντικαταστήσει κανείς κάθε πολυπλέκτη παράκαμψης με μια πύλη AND-OR που θα συνδυάζει το κρατούμενο εξόδου του αθροιστή των n bit, ή το κρατούμενο εισόδου των ομάδων, με το σήμα διάδοσης των ομάδων. Αυτό δουλεύει όντως για τους domino αθροιστές παράκαμψης κρατουμένου, στους οποίους το κρατούμενο εξόδου προφορτίζεται σε κάθε κύκλο· δουλεύει επίσης για τους αθροιστές πρόβλεψης κρατουμένου και επιλογής κρατουμένου που θα εξετάσουμε στην επόμενη ενότητα. Ωστόσο, αυτή η τεχνική έχει ως αποτέλεσμα υπερβολικά μεγάλο κρίσιμο μονοπάτι για ένα συνηθισμένο αθροιστή παράκαμψης κρατουμένου. Υποθέστε ότι δίνεται η πρόσθεση 111…111 + 000…000 + C_{in}. Όλα τα σήματα διάδοσης κρατουμένου ομάδας έχουν τιμή true. Εάν C_{in} = 1, τότε κάθε μπλοκ 4 βαθμίδων θα γεννήσει ένα κρατούμενο εξόδου. Όταν το C_{in} πέσει στο μηδέν, το μηδενικό σήμα κρατουμένου πρέπει να διαδοθεί διαμέσου όλων των N βαθμίδων λόγω του ότι το μονοπάτι διέρχεται μέσω του κρατουμένου εξόδου κάθε αθροιστή των n bit. Οι domino αθροιστές παράκαμψης κρατουμένου αποφεύγουν αυτό το μονοπάτι, επειδή σε όλα τα κρατούμενα επιβάλλεται μηδενική τιμή κατά τη διάρκεια της προφόρτισης, οπότε μπορούν να χρησιμοποιούν πύλες AND-OR.



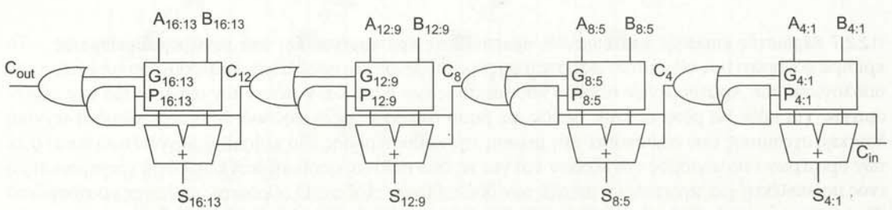
ΣΧΗΜΑ 11.20 Manchester αλυσίδα κρατουμένου στον αθροιστή παράκαμψης κρατουμένου.

Το Σχήμα 11.20 δείχνει πώς μπορεί να τροποποιηθεί η αλυσίδα κρατουμένου Manchester ενός αθροιστή, έτσι ώστε να εκτελεί παράκαμψη κρατουμένου [Chan90]. Χρησιμοποιείται αλυσίδα 5ης τάξης για την παράκαμψη μιας ομάδας των 4 bit τη φορά.

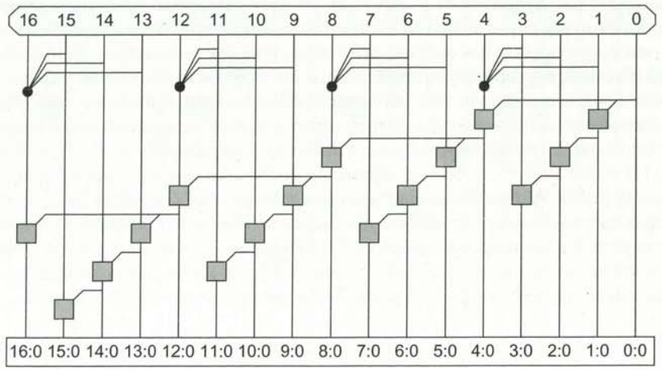
11.2.2.6 Αθροιστής πρόβλεψης κρατουμένου Ο *αθροιστής πρόβλεψης κρατουμένου* (carry-lookahead adder, CLA) [Weinberger58] είναι παρόμοιος με τον αθροιστή παράκαμψης κρατουμένου, με τη διαφορά ότι υπολογίζει τα σήματα γέννησης και διάδοσης κρατουμένου ομάδας ώστε να αποφεύγει την καθυστέρηση διάδοσης που χρειάζεται για να εξακριβώσει εάν η πρώτη ομάδα γέννησε ή όχι κρατούμενο. Ένας τέτοιος αθροιστής απεικονίζεται στο Σχήμα 11.21, ενώ το PG δίκτυό του, το οποίο χρησιμοποιεί μαύρα κύτταρα 4ης τάξης για τον υπολογισμό των σημάτων PG ομάδας 4 bit, παρουσιάζεται στο Σχήμα 11.22.

Γενικά, ένας CLA που χρησιμοποιεί k ομάδες των n bit έχει καθυστέρηση

$$t_{cla} = t_{pg} + t_{pg(n)} + \left[\left(n-1\right)+\left(k-1\right)\right] t_{AO} + t_{xor} \qquad \textbf{(11.14)}$$



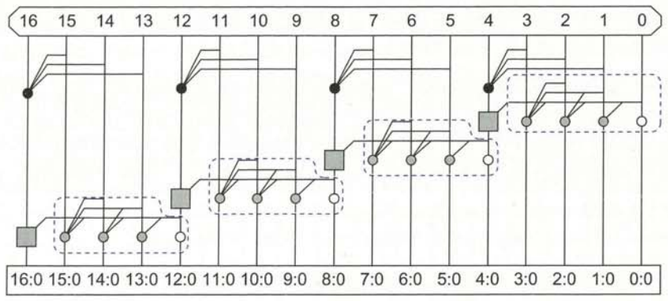
ΣΧΗΜΑ 11.21 Αθροιστής πρόβλεψης κρατουμένου.



ΣΧΗΜΑ 11.22 Δίκτυο PG ομάδας του αθροιστή πρόβλεψης κρατουμένου.

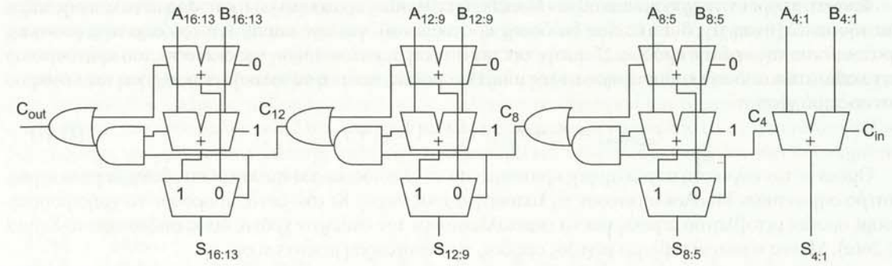
όπου $t_{pg(n)}$ η καθυστέρηση της πύλης AND-OR-AND-OR-…-AND-OR που υπολογίζει το σήμα γέννησης κρατουμένου n τάξης. Αυτή η καθυστέρηση δεν είναι μικρότερη από την καθυστέρηση του αθροιστή παράκαμψης κρατουμένου μεταβλητού μήκους του Σχήματος 11.19 και απαιτεί την επιπλέον n bit πύλη γέννησης κρατουμένου· συνεπώς, ο απλός CLA σπάνια αποτελεί καλή σχεδιαστική επιλογή. Ωστόσο, αποτελεί τη βάση για την κατανόηση των ταχύτερων αθροιστών που παρουσιάζονται στις επόμενες ενότητες.

Οι CLA χρησιμοποιούν συχνά κύτταρα μεγαλύτερης τάξης για τη μείωση της καθυστέρησης των προσθέσεων των n bit, εκτελώντας παράλληλα τον υπολογισμό των κρατουμένων. Το Σχήμα 11.23 παρουσιάζει έναν τέτοιο CLA, στον οποίο οι αθροιστές των τεσσάρων bit κατασκευάζονται με χρήση Manchester αλυσίδων κρατουμένου ή πολλαπλών στατικών πυλών που λειτουργούν παράλληλα.



ΣΧΗΜΑ 11.23 Δίκτυο PG ομάδας του βελτιωμένου αθροιστή πρόβλεψης κρατουμένου.

11.2.2.7 Αθροιστές επιλογής κρατουμένου, προσαύξησης κρατουμένου και υπό συνθήκη αθροίσματος Το κρίσιμο μονοπάτι των αθροιστών παράκαμψης κρατουμένου και πρόβλεψης κρατουμένου εμπεριέχει τον υπολογισμό των κρατουμένων εισόδου κάθε ομάδας των n bit, και κατόπιν τον υπολογισμό των αθροισμάτων για κάθε bit μέσα σε κάθε ομάδα, με βάση το κρατούμενο εισόδου. Μια καθιερωμένη τεχνική λογικής σχεδίασης που αποσκοπεί στη μείωση της καθυστέρησης του κρίσιμου μονοπατιού είναι ο εκ των προτέρων υπολογισμός των εξόδων και για τις δύο πιθανές εισόδους, και κατόπιν η χρησιμοποίηση ενός πολυπλέκτη για την επιλογή μεταξύ των δύο πιθανών εξόδων. Ο αθροιστής επιλογής κρατουμένου [Bedrij62] που παρουσιάζεται στο Σχήμα 11.24 κάνει ακριβώς αυτό, με τη χρήση ενός ζεύγους αθροιστών των n bit σε κάθε ομάδα. Ο ένας αθροιστής υπολογίζει τα αθροίσματα υποθέτοντας ότι το κρατούμενο εισόδου ισούται με 0, ενώ ο άλλος υπολογίζει τα αθροίσματα υποθέτοντας ότι το κρατούμενο εισόδου ισούται



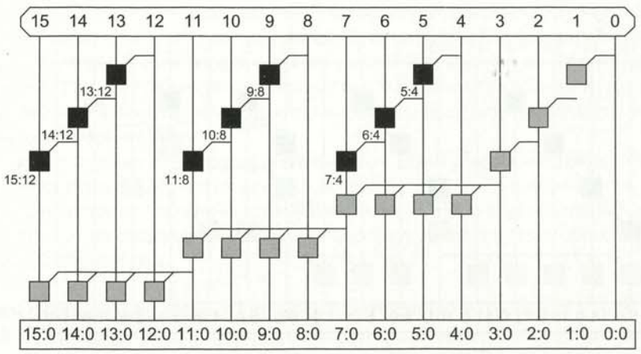
ΣΧΗΜΑ 11.24 Αθροιστής επιλογής κρατουμένου.

με 1. Το πραγματικό κρατούμενο ενεργοποιεί έναν πολυπλέκτη, ο οποίος επιλέγει το σωστό άθροισμα. Η καθυστέρηση του κρίσιμου μονοπατιού είναι

$$t_{\text{select}} = t_{pg} + \left[n + (k-2)\right] t_{AO} + t_{\text{mux}} \tag{11.15}$$

Η χρήση δύο αθροιστών των n bit συνιστά πλεονασμό, υπό την έννοια ότι αμφότεροι περιέχουν την ίδια αρχική PG λογική και την τελική πύλη XOR αθροίσματος. Ο [Tyagi93] μειώνει το μέγεθος του αθροιστή δημιουργώντας ένα κοινό κύκλωμα για τις πράξεις του κοινού τμήματος της λογικής και απλοποιώντας τον πολυπλέκτη σ' ένα μεμονωμένο γκρι κύτταρο, όπως βλέπετε στο Σχήμα 11.25. Ο προκύπτων αθροιστής αποκαλείται επίσης *αθροιστής προσαύξησης κρατουμένου* (carry-incrementadder) [Zimmermann96]. Χρησιμοποιεί μια μικρή αλυσίδα διάδοσης αποτελούμενη από μαύρα κύτταρα για τον υπολογισμό των PG σημάτων για τα bit εντός της κάθε ομάδας. Τα bit που περιλαμβάνει κάθε ομάδα επισημαίνονται στο διάγραμμα. Όταν το κρατούμενο εξόδου της προηγούμενης ομάδας γίνεται διαθέσιμο, το τελευταίο γκρι κύτταρο κάθε στήλης καθορίζει την τιμή του κρατουμένου εξόδου, το οποίο αποκτά τιμή true εάν η ομάδα γεννήσει ένα κρατούμενο ή εάν η προηγούμενη ομάδα είχε γεννήσει ένα κρατούμενο και η εν λόγω ομάδα διαδίδει το κρατούμενο αυτό. Ο αθροιστής προσαύξησης κρατουμένου έχει περίπου τα διπλάσια κύτταρα στο PG δίκτυό του από τον αθροιστή διάδοσης κρατουμένου. Η καθυστέρηση του κρίσιμου μονοπατιού ισούται περίπου με την καθυστέρηση ενός αθροιστή επιλογής κρατουμένου, επειδή ένας πολυπλέκτης και μια πύλη XOR έχουν συγκρίσιμη καθυστέρηση, αλλά η κατανάλωση επιφάνειας είναι μικρότερη στον αθροιστή προσαύξησης.

$$t_{\text{increment}} = t_{pg} + \left[(n-1) + (k-1)\right] t_{AO} + t_{\text{xor}} \tag{11.16}$$



ΣΧΗΜΑ 11.25 Δίκτυο PG ομάδας του αθροιστή επιλογής κρατουμένου.

Φυσικά, μπορούν να χρησιμοποιηθούν Manchester αλυσίδες κρατουμένου ή κύτταρα μεγαλύτερης τάξης για την επιτάχυνση της διαδικασίας διάδοσης κρατουμένου, για την παραγωγή του σήματος γέννησης κρατουμένου της πρώτης ομάδας. Σ' αυτήν την περίπτωση, η καθυστέρηση της διάδοσης του κρατουμένου αντικαθίσταται από την καθυστέρηση πύλης μίας PG ομάδας, οπότε η συνολική καθυστέρηση του κρίσιμου μονοπατιού γίνεται

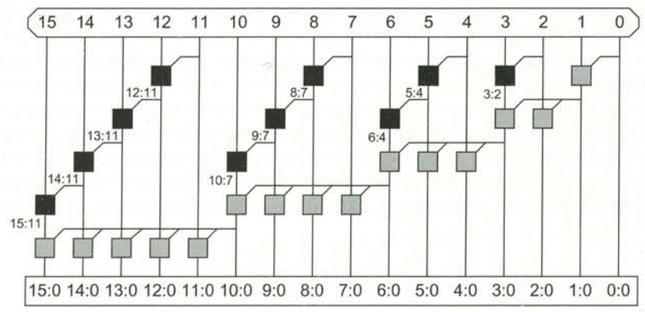
$$t_{\text{increment}} = t_{pg} + t_{pg(n)} + \left[k-1\right] t_{AO} + t_{\text{xor}} \qquad \textbf{(11.17)}$$

Όμοια με τον αθροιστή παράκαμψης κρατουμένου, οι αλυσίδες κρατουμένου για τις βαθμίδες των περισσότερο σημαντικών bit ολοκληρώνουν τη λειτουργία τους νωρίς. Κι εδώ ξανά, μπορούμε να χρησιμοποιήσουμε ομάδες μεταβλητού μήκους για να εκμεταλλευτούμε τον επιπλέον χρόνο, όπως επιδεικνύει το Σχήμα 11.26(α). Μ' ένα τέτοιο μεταβλητό μέγεθος ομάδας, η καθυστέρηση μειώνεται σε:

$$t_{\text{increment}} \approx t_{pg} + \sqrt{2N}\, t_{AO} + t_{\text{xor}} \qquad \textbf{(11.18)}$$

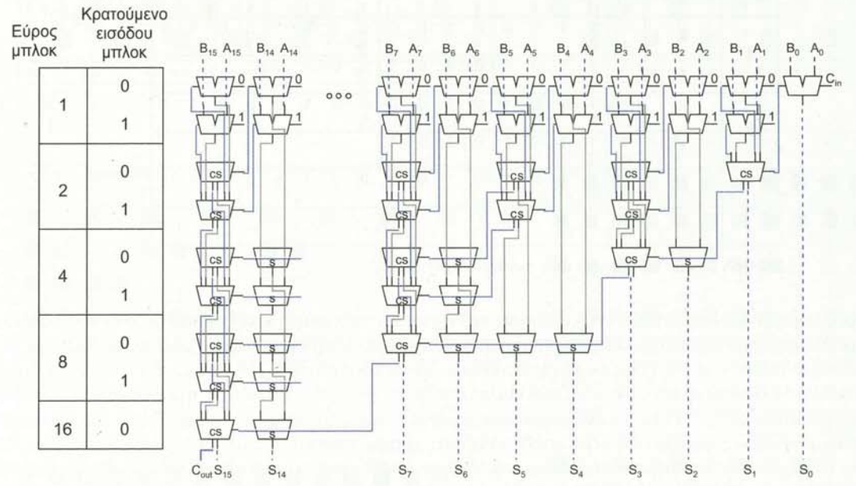
Οι εξισώσεις καθυστέρησης δεν συνυπολογίζουν τον αριθμό των εξόδων (fanout) που πρέπει να οδηγεί κάθε στάδιο. Σε μία ομάδα μεταβλητού μήκους, το fanout μπορεί να γίνει τόσο μεγάλο, ώστε να απαιτείται απομόνωση μεταξύ των βαθμίδων. Το Σχήμα 11.26(β) δείχνει πώς μπορούν να χρησιμοποιηθούν απομονωτές για τη μείωση του φόρτου διακλάδωσης, χωρίς επιβάρυνση του κρίσιμου μονοπατιού πρόβλεψης· πρόκειται για μια τεχνική που αποδεικνύεται χρήσιμη σε πολλές εφαρμογές.

Σε αθροιστές μεγάλου εύρους, μπορούμε να εφαρμόζουμε με αναδρομικό τρόπο πολλαπλά επίπεδα επιλογής ή προσαύξησης κρατουμένου. Για παράδειγμα, ένας αθροιστής επιλογής κρατουμένου των 64 bit μπορεί να κατασκευαστεί από τέσσερις αθροιστές επιλογής κρατουμένου των 16 bit, καθένας εκ των οποίων επιλέγει το κρατούμενο εισόδου της επόμενης ομάδας 16 bit. Επεκτείνοντας αυτό το σκεπτικό στο όριο, παίρνουμε τον



ΣΧΗΜΑ 10.26 Αθροιστής προσαύξησης κρατουμένου με PG ομάδες μεταβλητού μεγέθους.

αθροιστή υπό συνθήκη αθροίσματος (conditional-sum adder) [Sklansky60], ο οποίος εκτελεί επιλογή κρατουμένου ξεκινώντας με ομάδες του 1 bit και διπλασιάζοντας αναδρομικά έως τα $N/2$ bit. Το Σχήμα 11.27 παρουσιάζει έναν αθροιστή υπό συνθήκη αθροίσματος. Στις πρώτες δύο γραμμές, οι πλήρεις αθροιστές υπολογίζουν το άθροισμα και το κρατούμενο εξόδου για κάθε bit, υποθέτοντας κρατούμενα εισόδου 0 και 1, αντίστοιχα. Στις επόμενες δύο γραμμές, τα ζεύγη πολυπλεκτών επιλέγουν το άθροισμα και το κρατούμενο εξόδου του ανώτερου bit του κάθε μπλοκ των 2 bit, υποθέτοντας και πάλι κρατούμενα εισόδου 0 και 1, αντίστοιχα. Στις δύο επόμενες γραμμές, οι πολυπλέκτες επιλέγουν το άθροισμα και το κρατούμενο εξόδου των δύο σημαντικότερων bit του κάθε μπλοκ των 4 bit, κ.ο.κ.



ΣΧΗΜΑ 11.27 Αθροιστής υπό συνθήκη αθροίσματος.

Το Σχήμα 11.28 παρουσιάζει τη λειτουργία ενός αθροιστή υπό συνθήκη αθροίσματος για $N = 16$, με $C_{\text{in}} = 0$. Στη γραμμή του πίνακα που αντιστοιχεί σε εύρος μπλοκ ίσο με 1, ένα ζεύγος από πλήρεις αθροιστές υπολογίζει το άθροισμα και το κρατούμενο εξόδου για κάθε στήλη. Ο ένας αθροιστής λειτουργεί υποθέτοντας ότι το κρατούμενο εισόδου της στήλης ισούται με 0, ενώ ο άλλος αθροιστής υποθέτει ότι το κρατούμενο εισόδου ισούται με 1. Στη γραμμή του πίνακα που αντιστοιχεί σε εύρος μπλοκ ίσο με 2, ο αθροιστής επιλέγει το άθροισμα για το ανώτερο μισό του κάθε μπλοκ (τις μονές στήλες) βασισμένος στο κρατούμενο εξόδου του κατώτερου μισού. Ο αθροιστής υπολογίζει επίσης το κρατούμενο εξόδου του ζεύγους βαθμίδων. Κι εδώ ξανά, αυτό γίνεται δύο φορές – για τις δύο πιθανές τιμές του κρατουμένου εισόδου του μπλοκ. Στη γραμμή του πίνακα που αντιστοιχεί σε εύρος μπλοκ ίσο με 4, ο αθροιστής επιλέγει και πάλι το άθροισμα για το ανώτερο μισό του κάθε μπλοκ βασισμένος στο κρατούμενο εξόδου του κατώτερου μισού και βρίσκει το κρατούμενο εξόδου ολόκληρου του μπλοκ. Αυτή η διαδικασία επαναλαμβάνεται στις επόμενες γραμμές μέχρι να επιλεχτεί το άθροισμα των 16 bit και το τελικό κρατούμενο εξόδου.

Ο αθροιστής υπό συνθήκη αθροίσματος απαιτεί σχεδόν $2N$ πλήρεις αθροιστές και $2N\log_2 N$ πολυπλέκτες. Όμοια με τον αθροιστή επιλογής κρατουμένου, ο αθροιστής υπό συνθήκη αθροίσματος μπορεί να βελτιωθεί με τη δημιουργία ενός κοινού κυκλώματος για τις πύλες XOR του αθροίσματος και με τη χρήση πυλών AND-OR στη θέση των πολυπλεκτών. Αυτές οι προσαρμογές μας οδηγούν στον αθροιστή δένδρου Sklansky που θα εξετάσουμε στη συνέχεια.

11.2.2.8 Αθροιστές δένδρου (tree adders) Για τους αθροιστές μεγάλου εύρους ($N > 16$ bits), η καθυστέρηση των αθροιστών πρόβλεψης (ή παράκαμψης, ή επιλογής) κρατουμένου κυριαρχείται από την καθυστέρηση που συνεπάγεται το πέρασμα του κρατουμένου από τα στάδια πρόβλεψης. Αυτή η καθυστέρηση μπορεί να μειωθεί εφαρμόζοντας ένα δεύτερο επίπεδο πρόβλεψης επί των μπλοκ που εκτελούν την κατ' αρχήν

Εύρος μπλοκ	Κρατούμενο εισόδου μπλοκ		Άθροισμα και κρατούμενο εξόδου μπλοκ																
		a	1 0	0 0	1 0	1 1	1 1	0 0	1 0	1 1	0 1	1 0	1 1	0 1	1 0	1 1	0 1	1 0	0
		b	1 0	0 0	1 0	0 1	0 1	0 0	1 0	0 1	1 0	1 0	0 1	1 0	0 1	0 1	1 0	0 1	C_{in}
1	0	s	1 0	0 0	1 0	0 1	0 1	0 0	1 0	0 1	1 0	1 0	0 1	1 0	0 1	1 0	0 1	1 0	
	1	s	0 1	1 0	0 1	1 1	1 1	1 0	0 1	1 1	0 1	0 1	1 1	0 1	1 1	0 1	1 1	0 1	
2	0	s	1 0	1 0	0 1	0 0	0 1	0 0	0 1	0 0	1 0	1 0	0 1	0 1	0 0	0 1	1 0	1 1	
	1	s	1 0	1 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 0	1 1	0 1	0 1	0 1			
4	0	s	1 0	1 1	0 0	0 1	0 1	0 0	0 1	0 0	0 1	0 0	0 1	0 1	0 0	0 1	1 1		
	1	s	1 0	1 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 0	1 0	0 1					
8	0	s	1 0	1 1	0 1	0 1	0 1	0 0	0 1	0 0	1 0	0 0	0 1	1 1					
	1	s	1 0	1 1	0 1	0 1	0 1	0 1	0 1	0 1									
16	0	s	1 0	1 1	0 1	0 1	0 1	0 1	0 1	0 0	0 1	0 0	0 1	1 1					Άθροισμα
	1	s																	C_{out}

ΣΧΗΜΑ 11.28 Παράδειγμα υπό συνθήκη πρόσθεσης.

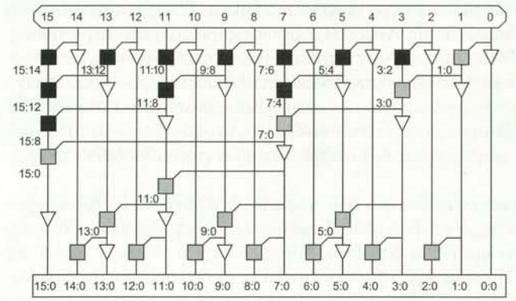
πρόβλεψη [Weinberger58]. Γενικά, μπορεί κανείς να κατασκευάσει ένα πολυεπίπεδο δένδρο δομών πρόβλεψης για να επιτύχει καθυστέρηση που αυξάνεται συναρτήσει του $\log N$. Οι αθροιστές αυτού του είδους αναφέρονται συνήθως ως *αθροιστές δένδρου, λογαριθμικοί αθροιστές, αθροιστές πολυεπίπεδης πρόβλεψης (multilevel-lookahead adders), αθροιστές παράλληλου προθέματος (parallel-prefix adders),* ή απλώς *αθροιστές πρόβλεψης (lookahead adders)*. Η τελευταία ονομασία εμφανίζεται περιστασιακά στη βιβλιογραφία, αλλά δεν είναι απολύτως ακριβής επειδή δεν υποδεικνύει εάν χρησιμοποιούνται πολλαπλά επίπεδα πρόβλεψης.

Υπάρχουν πολλοί τρόποι κατασκευής του δένδρου πρόβλεψης, οι οποίοι βασίζονται σε διαφορετικούς συμβιβασμούς μεταξύ του πλήθους επιπέδων λογικής, του πλήθους λογικών πυλών, του μέγιστου βαθμού οδήγησης εξόδου κάθε πύλης και της ποσότητας αγωγών διασύνδεσης μεταξύ των σταδίων. Τρεις βασικές αρχιτεκτονικές δένδρων είναι οι Brent-Kung, Sklansky και Kogge-Stone. Θα ξεκινήσουμε την εξέταση της κάθε αρχιτεκτονικής με την περίπτωση της 2ης τάξης, η οποία συνδυάζει ζεύγη ομάδων σε κάθε στάδιο.

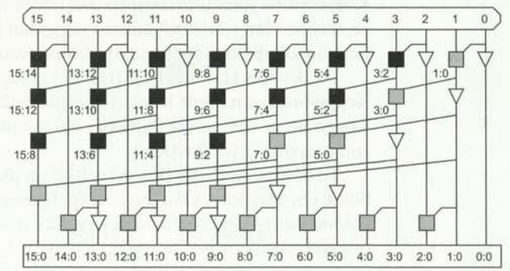
Το δένδρο Brent-Kung [Brent82] (Σχήμα 11.29(α)) υπολογίζει προθέματα για ομάδες 2 βαθμίδων (bit). Αυτά χρησιμοποιούνται για τον υπολογισμό προθεμάτων ομάδας 4 bit, τα οποία με τη σειρά τους χρησιμοποιούνται για τον υπολογισμό προθεμάτων ομάδας 8 bit, κ.ο.κ. Στη συνέχεια τα προθέματα αυτά «εξαπλώνονται» προς τα κάτω, για τον υπολογισμό των κρατουμένων εισόδου κάθε βαθμίδας. Το δένδρο απαιτεί την ύπαρξη $2(\log_2 N) - 1$ σταδίων. Ο βαθμός οδήγησης εξόδου περιορίζεται σε 2 για κάθε στάδιο. Το διάγραμμα παρουσιάζει απομονωτές, οι οποίοι χρησιμοποιούνται για την ελαχιστοποίηση του βαθμού οδήγησης εξόδου και του φόρτου των πυλών, αλλά στην πράξη οι απομονωτές αυτοί συνήθως αποφεύγονται.

Το δένδρο Sklansky, ή αλλιώς δένδρο λογικής «διαίρει και βασίλευε» [Sklansky60] (Σχήμα 11.29(β)), μειώνει την καθυστέρηση σε $\log_2 N$ στάδια, με τον υπολογισμό ενδιάμεσων προθεμάτων μαζί με τα προθέματα των μεγάλων ομάδων. Αυτό επιτυγχάνεται με αντίτιμο την αύξηση του βαθμού οδήγησης εξόδου, ο οποίος διπλασιάζεται σε κάθε στάδιο: οι πύλες οδηγούν [8, 4, 2, 1] άλλες στήλες. Αυτοί οι μεγάλοι βαθμοί οδήγησης έχουν ως αποτέλεσμα τη μείωση της απόδοσης των αθροιστών μεγάλου εύρους, εκτός εάν καθοριστούν τα κατάλληλα μεγέθη για τις πύλες, ή εάν τα σήματα του κρίσιμου μονοπατιού απομονώνονται πριν χρησιμοποιηθούν για τον υπολογισμό των ενδιάμεσων προθεμάτων. Η επιλογή συγκεκριμένων μεγεθών για τα τρανζίστορ μπορεί να υποβαθμίσει την κανονικότητα του φυσικού σχεδίου, επειδή απαιτούνται πολλές τιμές μεγέθους για κάθε κύτταρο, παρότι οι μεγαλύτερες πύλες μπορούν να εξαπλώνονται σε γειτονικές στήλες. Σημειώστε ότι ο περιοδικός διπλασιασμός στο δένδρο Sklansky είναι ανάλογος μ' αυτόν του αθροιστή υπό συνθήκη αθροίσματος του Σχήματος 11.27. Με κατάλληλη ενδιάμεση απομόνωση, ο βαθμός οδήγησης εξόδου μπορεί να μειωθεί σε [8, 1, 1, 1], όπως εξετάζεται στην Άσκηση 11.7.

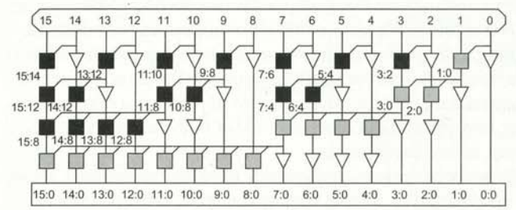
Το δένδρο Kogge-Stone [Kogge73] (Σχήμα 11.29(γ)) επιτυγχάνει καθυστέρηση $\log_2 N$ σταδίων και ταυ-



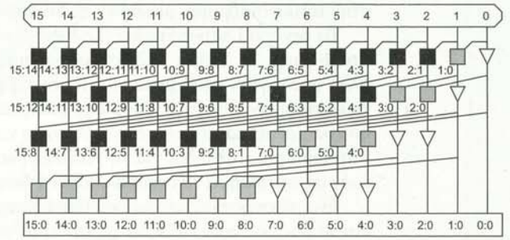
(α) Brent-Kung



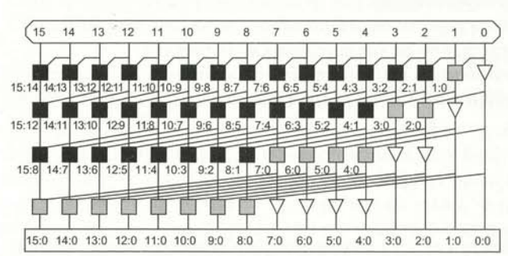
(δ) Han-Carlson



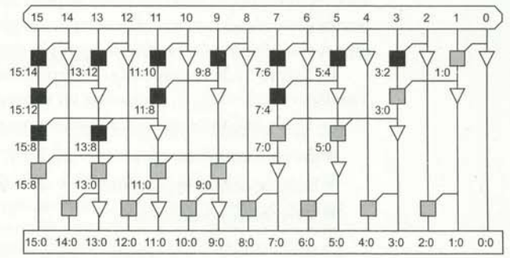
(β) Sklansky



(ε) Knowles [2,1,1,1]



(γ) Kogge-Stone



(στ) Ladner-Fischer

ΣΧΗΜΑ 11.29 Δίκτυα PG ομάδων για διάφορους αθροιστές δένδρου.

τόχρονα βαθμό οδήγησης εξόδου ίσο με 2 για κάθε στάδιο. Αυτό επιτυγχάνεται με αντίτιμο την ύπαρξη πολλών, μεγάλου μήκους αγωγών που πρέπει να δρομολογούνται ανάμεσα στα στάδια. Επίσης, το δένδρο αυτό περιλαμβάνει περισσότερα PG κύτταρα· παρόλο που αυτό μπορεί να μην επηρεάσει την επιφάνεια του αθροιστή εάν το φυσικό σχέδιο βασίζεται σ' ένα κανονικό πλέγμα, θα προκαλέσει αύξηση της κατανάλωσης ισχύος. Παρά τα μειονεκτήματά του, το δένδρο Kogge-Stone χρησιμοποιείται ευρέως σε υψηλής απόδοσης αθροιστές των 32 και 64 bit.

Συνοπτικά, ένας αθροιστής δένδρου Sklansky ή Kogge-Stone μειώνει το κρίσιμο μονοπάτι σε:

$$t_{tree} \approx t_{pg} + \lceil \log_2 N \rceil t_{AO} + t_{xor} \qquad \textbf{(11.19)}$$

Ένας ιδανικός αθροιστής δένδρου θα είχε $\log_2 N$ επίπεδα λογικής, βαθμό οδήγησης εξόδου που ποτέ δε θα ξεπερνούσε το 2, και όχι πάνω από ένα μονοπάτι αγωγών διασύνδεσης (δέσμες G_{ij} και P_{ij}) ανάμεσα στις γραμμές. Οι βασικές αρχιτεκτονικές δένδρου αναπαριστούν περιπτώσεις οι οποίες προσεγγίζουν την ιδανική, αλλά η καθεμία διαφέρει από αυτήν σε μια συγκεκριμένη παράμετρο. Η αρχιτεκτονική Brent-Kung

έχει πάρα πολλά λογικά επίπεδα. Η αρχιτεκτονική Sklansky έχει υπερβολικό fanout. Τέλος, η αρχιτεκτονική Kogge-Stone χρειάζεται υπερβολική ποσότητα διασύνδεσης. Ανάμεσα σ' αυτές τις τρεις ακραίες περιπτώσεις, τα δένδρα Han-Carlson, Ladner-Fischer και Knowles προσφέρουν επιπλέον σχεδιαστικές επιλογές, με διαφορετικούς συμβιβασμούς μεταξύ πλήθους σταδίων, βαθμού οδήγησης εξόδου και πλήθους αγωγών διασύνδεσης.

Τα δένδρα Han-Carlson [Han87] είναι μια οικογένεια δικτύων που βρίσκεται ανάμεσα στα δένδρα Kogge-Stone και Brent-Kung. Το Σχήμα 11.29(δ) παρουσιάζει ένα τέτοιο δένδρο, το οποίο υλοποιεί τη λογική Kogge-Stone στις μονές βαθμίδες και κατόπιν χρησιμοποιεί ένα επιπλέον στάδιο για να διαδώσει το κρατούμενο στις ζυγές βαθμίδες.

Τα δένδρα Knowles [Knowles01] είναι μια οικογένεια δικτύων που βρίσκεται ανάμεσα στα δένδρα Kogge-Stone και Sklansky. Όλα αυτά τα δένδρα έχουν $\log_2 N$ στάδια, αλλά διαφέρουν ως προς το βαθμό οδήγησης εξόδου και το πλήθος των αγωγών. Εάν θεωρήσουμε ότι οι αθροιστές Kogge-Stone και Sklansky των 16 bit οδηγούν άλλες στήλες με fanout ίσο με [1, 1, 1, 1] και [8, 4, 2, 1] αντίστοιχα, τότε τα δίκτυα Knowles βρίσκονται κάπου ανάμεσα σ' αυτές τις ακραίες περιπτώσεις. Για παράδειγμα, το Σχήμα 11.29(ε) παρουσιάζει ένα δένδρο Knowles με βαθμούς οδήγησης εξόδων [2, 1, 1, 1], το οποίο υποδιπλασιάζει το πλήθος των αγωγών στην τελική διαδρομή, με κόστος το διπλασιασμό του φόρτου σ' αυτούς τους αγωγούς.

Τα δένδρα Ladner-Fischer [Ladner80] είναι μια οικογένεια δικτύων μεταξύ των δένδρων Sklansky και Brent-Kung. Το Σχήμα 11.29(στ) παρουσιάζει ένα τέτοιο δένδρο, το οποίο είναι παρόμοιο με το Sklansky, αλλά υπολογίζει προθέματα για τις μονές βαθμίδες, ενώ χρησιμοποιεί ένα επιπλέον στάδιο για να διαδώσει το κρατούμενο στις ζυγές βαθμίδες. Τα κύτταρα που βρίσκονται σε κόμβους με υψηλό βαθμό οδήγησης εξόδου θα πρέπει και σ' αυτή την περίπτωση να κατασκευάζονται με συγκεκριμένο μέγεθος ή να ομαδοποιούνται (gang) κατάλληλα, για την επίτευξη καλής ταχύτητας. Σημειώστε ότι πολλοί συγγραφείς αντιμετωπίζουν ως ταυτόσημα τα δένδρα Ladner-Fischer και τα δένδρα Sklansky.

Ένα πλεονέκτημα του δικτύου Brent-Kung και των δικτύων που σχετίζονται μ' αυτό (το δίκτυο Han-Carlson και το δίκτυο Ladner-Fischer με την επιπλέον γραμμή) είναι ότι για κάθε γραμμή δεν υπάρχει ποτέ πάνω από ένα κύτταρο σε κάθε ζεύγος στηλών. Αυτά τα δίκτυα έχουν μικρό πλήθος πυλών. Επιπλέον, το φυσικό τους σχέδιο μπορεί να είναι μόνο το μισό σε πλάτος, μειώνοντας έτσι το μήκος των οριζόντιων αγωγών που διατρέχουν ολόκληρο τον αθροιστή. Αυτό μειώνει τη χωρητικότητα διασύνδεσης, η οποία μπορεί να αποτελεί σημαντικό παράγοντα καθυστέρησης σε αθροιστές των 64 bit και μεγαλύτερους [Huang00].

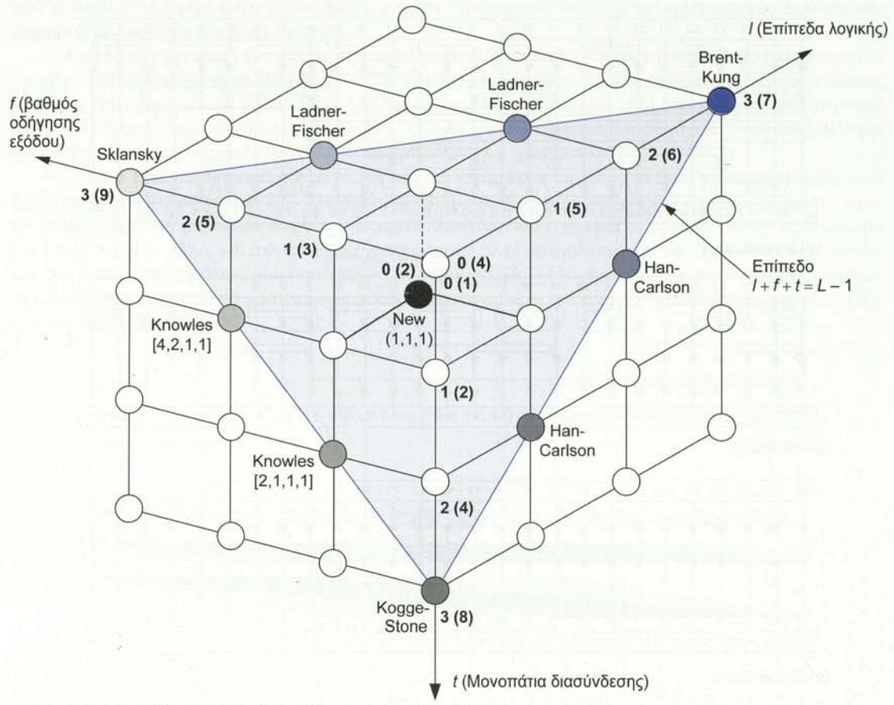
Το Σχήμα 11.30 παρουσιάζει ένα τρισδιάστατο σύστημα ταξινόμησης των αθροιστών δένδρου [Harris03]. Υποθέτοντας $L = \log_2 N$, μπορούμε να περιγράψουμε κάθε δένδρο με τρεις ακεραίους (l, f, t), στο πεδίο τιμών $[0, L - 1]$. Οι ακέραιοι αυτοί καθορίζουν τα εξής:

- Λογικά επίπεδα: $L + l$
- Βαθμός οδήγησης εξόδου: $2^f + 1$
- Μονοπάτια διασύνδεσης: 2^t

Οι αθροιστές δένδρου βρίσκονται στο επίπεδο $l + f + t = L - 1$.Οι αθροιστές δένδρου Brent-Kung, Sklansky και Kogge-Stone των 16 bit αντιστοιχούν στις κορυφές του κύβου (3, 0, 0), (0, 3, 0) και (0, 0, 3), αντίστοιχα. Οι αθροιστές δένδρου Han-Carlson, Ladner-Fischer και Knowles βρίσκονται κατά μήκος των διαγωνίων.

11.2.2.9 Αθροιστές δένδρου μεγαλύτερης τάξης Όλες οι δομές δένδρων που περιγράψαμε παραπάνω μπορούν να συνδυάζουν περισσότερες από δύο ομάδες σε κάθε στάδιο [Beaumont-Smith01]. Το πλήθος των ομάδων που συνδυάζονται σε κάθε πύλη ονομάζεται *τάξη* ή *βάση* (valency, radix) του κυττάρου. Για παράδειγμα, το Σχήμα 11.31 παρουσιάζει 3ης τάξης δένδρα Brent-Kung, Sklansky, Kogge-Stone και Han-Carlson των 27 bit. Τα στρογγυλεμένα πλαίσια υποδεικνύουν αλυσίδες κρατουμένου 3ης τάξης (οι οποίες μπορούν να κατασκευάζονται με μια Manchester αλυσίδα κρατουμένου, μια πύλη domino πολλαπλών εξόδων, ή πολλαπλές διακριτές πύλες). Τα τραπεζοειδή υποδεικνύουν λειτουργίες προσαύξησης κρατουμένου. Οι σχεδιάσεις υψηλότερης τάξης χρησιμοποιούν λιγότερα στάδια λογικής, αλλά το κάθε στάδιο έχει μεγαλύτερη καθυστέρηση. Αυτή δεν είναι αποτελεσματική επιλογή για τα στατικά κυκλώματα CMOS, επειδή ο φόρτος σταδίου γίνεται πολύ μεγαλύτερος από 4, αλλά δίνει καλά αποτελέσματα για τη λογική διαδοχικής επίδρασης (domino), επειδή ο λογικός φόρτος σ' αυτήν είναι πολύ μικρότερος και άρα απαιτούνται λιγότερα στάδια.

Οι κόμβοι με μεγάλο βαθμό οδήγησης εξόδου ή αγωγούς διασύνδεσης μεγάλου μήκους μπορούν να χρησιμοποιούν απομονωτές. Τα δένδρα υπολογισμού προθέματος μπορούν επίσης να υλοποιούνται εσωτερικά με διαδοχική διοχέτευση (pipeline), για λειτουργία υψηλής ρυθμαπόδοσης (throughput). Ορισμένες σχεδιάσεις υψηλότερης τάξης συνδυάζουν το αρχικό στάδιο PG με το πρώτο επίπεδο της συγχώνευσης PG. Για παράδειγμα, ο αθροιστής Ling, που περιγράφεται στην Ενότητα 11.2.2.11, υπολογίζει τα σήματα γέννησης και διάδοσης κρατουμένου για ομάδες 4 bit από τις αρχικές εισόδους, σ' ένα μόνο στάδιο.

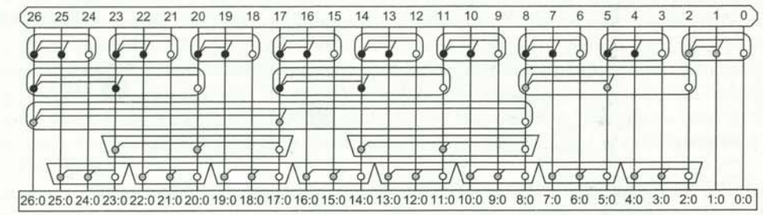


ΣΧΗΜΑ 11.30 Ταξινόμηση των αθροιστών με δίκτυο προθέματος.

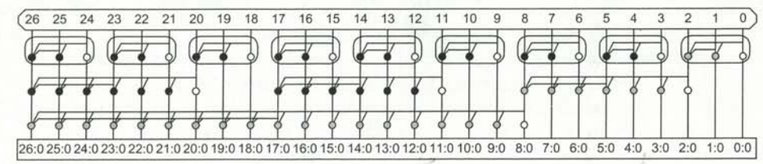
Οι αθροιστές υψηλότερης τάξης (v) μπορούν επίσης να περιγράφονται μ' ένα τρισδιάστατο σύστημα ταξινόμησης με $L = \log_v N$ και $l + f + t = L - 1$. Στη χείριστη περίπτωση, υπάρχουν $L + l$ επίπεδα λογικής, μέγιστος βαθμός οδήγησης εξόδου $(v - 1)v^f + 1$ και $(v - 1)v^t$ μονοπάτια διασύνδεσης.

11.2.2.10 Αθροιστές «αραιού» δένδρου Η δημιουργία ενός δένδρου προθέματος για τον υπολογισμό κρατουμένων σε κάθε βαθμίδα είναι δαπανηρή από την άποψη της κατανάλωσης ισχύος. Μια εναλλακτική επιλογή είναι ο υπολογισμός κρατουμένων μόνο σε μικρές ομάδες (π.χ., $s = 2, 4, 8,$ ή 16 bit). Εν τω μεταξύ, ζεύγη s-bit αθροιστών υπολογίζουν εκ των προτέρων τα αθροίσματα και για τις δύο πιθανές τιμές του κρατουμένου εισόδου (0 και 1) για κάθε ομάδα. Ένας πολυπλέκτης επιλέγει το σωστό άθροισμα για κάθε ομάδα με βάση τα κρατούμενα από το δένδρο προθεμάτων. Το μήκος της ομάδας μπορεί να εξισορροπηθεί με τρόπο ώστε το κρατούμενο εισόδου και τα προ-υπολογισμένα αθροίσματα να γίνονται διαθέσιμα σχεδόν την ίδια χρονική στιγμή. Μια τέτοια υβριδική σχεδίαση, μεταξύ αθροιστή προθέματος και αθροιστή επιλογής κρατουμένου, αποκαλείται *αθροιστής αραιού δένδρου* (sparse tree adder), με το s να αντιπροσωπεύει το συντελεστή αραιότητας του δένδρου.

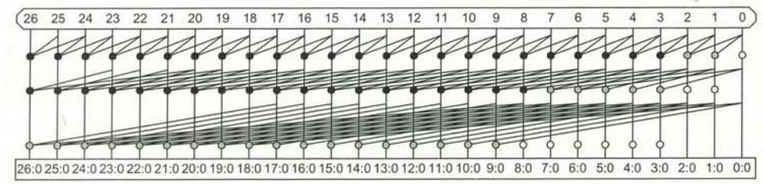
Ο *αθροιστής επεκτεινόμενου δένδρου* (spanning-tree adder) [Lynch92] είναι ένας υβριδικός αθροιστής βασιζόμενος σ' ένα υψηλής τάξης δένδρο Brent-Kung, όπως αυτό του Σχήματος 11.31(α). Το Σχήμα 11.32 παρουσιάζει μια απλή εκδοχή 3ης τάξης, η οποία προ-υπολογίζει αθροίσματα για ομάδες 3 βαθμίδων $(s = 3)$ και εξοικονομεί ένα επίπεδο λογικής λόγω του ότι επιλέγει την έξοδο με βάση τα κρατούμενα εισόδου της κάθε ομάδας. Το κρατούμενο εξόδου (C_{out}) υποδεικνύεται ρητά. Σημειώστε ότι η λιγότερο σημαντική



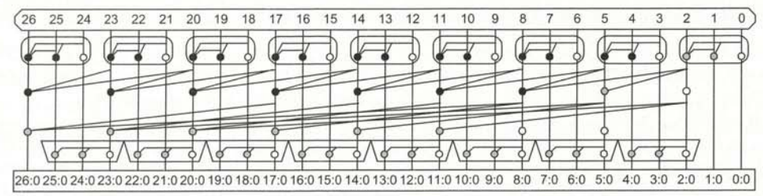
(α) Brent-Kung



(β) Sklansky

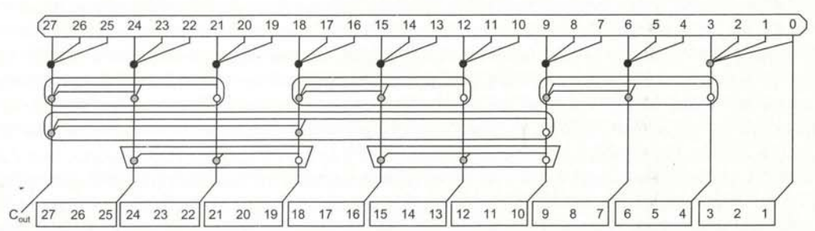


(γ) Kogge-Stone



(δ) Han-Carlson

ΣΧΗΜΑ 11.31 Αθροιστές δένδρου υψηλότερης τάξης.

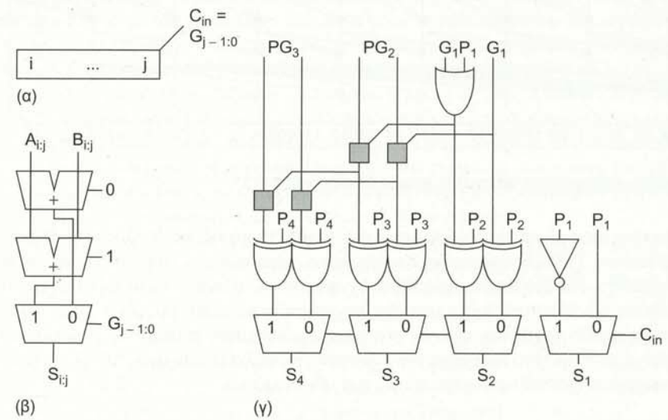


ΣΧΗΜΑ 11.32 3ης τάξης, Brent-Kung αθροιστής αραιού δένδρου με $s = 3$.

ομάδα απαιτεί ένα γκρι κύτταρο 4ης τάξης για τον υπολογισμό του $G_{3:0}$, του κρατουμένου εισόδου του δεύτερου μπλοκ επιλογής κρατουμένου.

Ο [Lynch92] περιγράφει μια σχεδίαση επεκτεινόμενου δένδρου των 56 bit, το οποίο υπάρχει στη μονάδα κινητής υποδιαστολής του επεξεργαστή AM29050 της AMD και χρησιμοποιεί στάδια 4ης τάξης και ομάδες επιλογής κρατουμένου των 8 βαθμίδων. Οι [Kantabutra93] και [Blackburn96] περιγράφουν βελτιστοποιήσεις του αθροιστή επεκτεινόμενου δένδρου με τη χρησιμοποίηση μεταβλητού μήκους σταδίων επιλογής κρατουμένου και την κατάλληλη επιλογή των μεγεθών των τρανζίστορ.

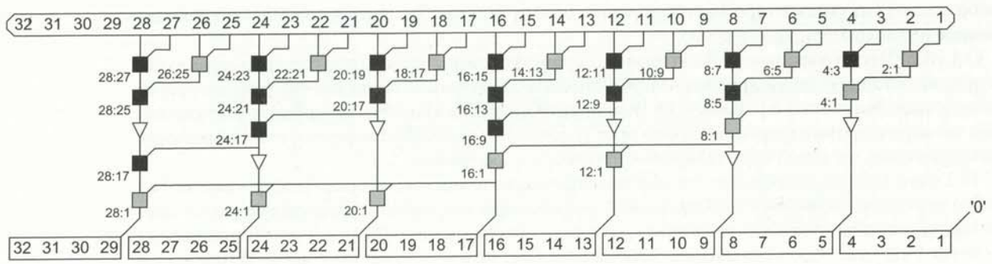
Το Σχήμα 11.33(α) παρουσιάζει ένα πλαίσιο επιλογής κρατουμένου, το οποίο αντιπροσωπεύει ένα δένδρο που εκτείνεται στα bits $i \ldots j$. Χρησιμοποιεί μικρούς αθροιστές διάδοσης κρατουμένου για τον προυπολογισμό των αθροισμάτων, υποθέτοντας κρατούμενο εισόδου της ομάδας ίσο με 0 για το ένα άθροισμα και ίσο με 1 για το άλλο, και στη συνέχεια χρησιμοποιεί έναν πολυπλέκτη για την επιλογή μεταξύ αυτών των δύο τιμών αθροισμάτων, όπως επιδεικνύει το Σχήμα 11.33(β). Οι αθροιστές μπορούν να απλοποιηθούν σε κάποιο βαθμό, επειδή τα κρατούμενα εισόδου είναι σταθερά, όπως υποδεικνύεται στο Σχήμα 11.33(γ) για μια ομάδα 4 bit.



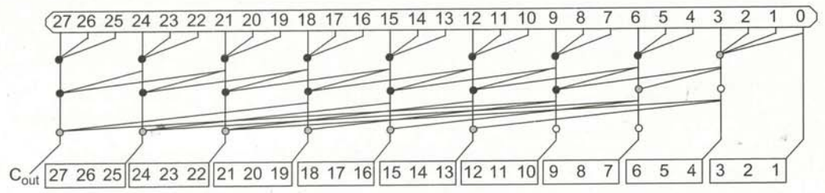
ΣΧΗΜΑ 11.33 Υλοποίηση της επιλογής κρατουμένου.

Ο [Mathew03] περιγράφει έναν αθροιστή αραιού δένδρου των 32 bit, ο οποίος χρησιμοποιεί ένα δένδρο 2ης τάξης παρόμοιο με το δένδρο Sklansky, για τον υπολογισμό μόνο των κρατουμένων εισόδου κάθε ομάδας 4 bit, όπως παρουσιάζεται στο Σχήμα 11.34. Αυτό μειώνει το πλήθος των πυλών και την κατανάλωση ισχύος του δένδρου. Το δένδρο μπορεί να θεωρηθεί ως ένα (2, 2, 0) Ladner-Fischer δένδρο, με την αντικατάσταση των δύο τελευταίων επιπέδων του δένδρου και της πύλης XOR μ' έναν πολυπλέκτη επιλογής. Ο αθροιστής υποθέτει κρατούμενο εισόδου ίσο με 0 και δεν παράγει κρατούμενο εξόδου, αποθηκεύοντας μια είσοδο στο λιγότερο σημαντικό γκρι πλαίσιο και εξαλείφοντας τη λογική του προθέματος στις τέσσερις περισσότερο σημαντικές στήλες.

Αυτές οι υβριδικές προσεγγίσεις αραιών δένδρων χρησιμοποιούνται ευρέως σε υψηλών επιδόσεων, 32-/64-bit αθροιστές μεγαλύτερης τάξης, επειδή προσφέρουν το μικρό πλήθος λογικών επιπέδων ενός δένδρου υψηλής τάξης με την παράλληλη μείωση του πλήθους των πυλών και της κατανάλωσης ισχύος. Το Σχήμα 11.35 παρουσιάζει μια 3ης τάξης σχεδίαση δένδρου Kogge-Stone των 27 bit με επιλογή κρατουμένου σε ομάδες των 3 bit. Παρατηρήστε ότι το πλήθος των πυλών στο δένδρο υποτριπλασιάζεται. Επιπλέον, επειδή μειώνεται επίσης το πλήθος των αγωγών, η επιφάνεια που περισσεύει μπορεί να χρησιμοποιηθεί ως προστασία, για τη μείωση της καθυστέρησης μονοπατιού. Αυτή η σχεδίαση μπορεί να θεωρηθεί ως μια εκδοχή του αθροιστή Han-Carlson του Σχήματος 11.31(δ), με αντικατάσταση του τελευταίου επιπέδου λογικής από έναν πολυπλέκτη επιλογής κρατουμένου.



ΣΧΗΜΑ 11.34 2ης τάξης, Sklansky αθροιστής αραιού δένδρου με s = 4 (χρησιμοποιείται από την Intel).



ΣΧΗΜΑ 11.35 3ης τάξης, Kogge-Stone αθροιστής αραιού δένδρου με s = 3.

Οι υβριδικές προσεγγίσεις με αραιά δένδρα μειώνουν εκείνο το μέρος του δένδρου προθεμάτων που έχει το μεγαλύτερο κόστος. Για αρχιτεκτονικές Kogge-Stone, μειώνουν τον αριθμό των απαιτούμενων αγωγών κατά συντελεστή s. Για αρχιτεκτονικές Sklansky, μειώνουν το fanout κατά s. Για αρχιτεκτονικές Brent-Kung, εξαλείφουν τα τελευταία $\log_2 s$ επίπεδα λογικής. Ουσιαστικά, μπορούν να μετακινήσουν έναν αθροιστή προς το σημείο αρχής των αξόνων στο χώρο σχεδιαστικών επιλογών (l, f, t). Ωστόσο, αυτά τα οφέλη παρέχονται με αντίτιμο ένα fanout s στον τελευταίο πολυπλέκτη επιλογής, τη χρήση επιφάνειας και την ισχύ που απαιτείται για τον προ-υπολογισμό των αθροισμάτων.

OPTIONAL

11.2.2.11 Αθροιστές Ling Ο Ling ανακάλυψε μια τεχνική για την αφαίρεση ενός εν σειρά τρανζίστορ από το κρίσιμο μονοπάτι των σημάτων γέννησης κρατουμένου ομάδας με αντίτιμο τη συμπερίληψη μιας επιπλέον πύλης XOR στο δίκτυο προ-υπολογισμού του αθροίσματος [Ling81, Doran88, Bewick94]. Η τεχνική βασίζεται στη χρήση του σήματος $\overline{K}$ αντί του P στο δίκτυο προθέματος και στην παρατήρηση ότι $G_i\overline{K}_i = (A_iB_i)(A_i + B_i) = G_i$.

Ορίζεται ένα σήμα ψευδο-γέννησης κρατουμένου (αποκαλείται επίσης σήμα ψευδο-κρατουμένου) $H_{i:j} = G_i + G_{i-1:j}$. Αυτό είναι απλούστερο από το $G_{i:j} = G_i + P_iG_{i-1:j}$. Το $G_{i:j}$ μπορεί να ληφθεί (μέσω μιας AND) από το $H_{i:j}$ αργότερα, όταν θα καταστεί αναγκαίο:

$$\overline{K}_iH_{i:j} = \overline{K}_iG_i + \overline{K}_iG_{i-1:j} = G_i + \overline{K}_iG_{i-1:j} = G_{i:j} \tag{11.20}$$

Το πλεονέκτημα των σημάτων ψευδο-γέννησης κρατουμένου έναντι του κανονικού σήματος γέννησης κρατουμένου έγκειται στο ότι είναι ευκολότερο να υπολογιστεί η πρώτη γραμμή στο δίκτυο προθέματος.

Ορίζεται επίσης ένα σήμα ψευδο-διάδοσης κρατουμένου I, το οποίο είναι απλώς μια μετατοπισμένη εκδοχή του σήματος διάδοσης: $I_{i:j} = \overline{K}_{i-1:j-1}$. Τα σήματα ψευδο-γέννησης και ψευδο-διάδοσης κρατουμένου ομάδας συνδυάζονται, χρησιμοποιώντας τα ίδια μαύρα ή γκρι κύτταρα με τα κανονικά σήματα γέννησης και διάδοσης, όπως θα σας ζητηθεί να δείξετε στην Άσκηση 11.11.

$$\begin{aligned} H_{i:j} &= H_{i:k} + I_{i:k}H_{k-1:j} \\ I_{i:j} &= I_{i:k}I_{k-1:j} \end{aligned} \tag{11.21}$$

Τα πραγματικά σήματα γέννησης ομάδας σχηματίζονται από τα σήματα ψευδο-γέννησης, χρησιμοποιώντας την Εξ. (11.20). Αυτά τα σήματα μπορούν να χρησιμοποιηθούν για τον υπολογισμό των αθροισμάτων με τη συνήθη XOR: $S_i = P_i \oplus G_{i-1:0} = P_i \oplus (\overline{K}_{i-1}H_{i-1:0})$. Για να αποφύγουμε την εισαγωγή μιας πύλης AND στο κρίσιμο μονοπάτι, αναπτύσσουμε το S_i σε όρους του $H_{i-1:0}$

$$S_i = H_{i-1:0}\left[P_i \oplus \overline{K}_{i-1}\right] + \overline{H}_{i-1:0}\left[P_i\right] \tag{11.22}$$

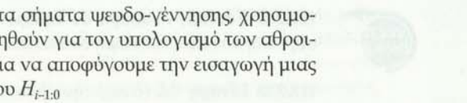
Συνεπώς, η επιλογή αθροίσματος μπορεί να υλοποιηθεί μ' έναν πολυπλέκτη, ο οποίος επιλέγει είτε το $P_i \oplus \overline{K}_{i-1}$ είτε το P_i, βάσει του $H_{i-1:0}$.

Η τεχνική που χρησιμοποιεί ο αθροιστής Ling μπορεί επίσης να χρησιμοποιηθεί με οποιαδήποτε μορφή αθροιστή, ο οποίος χρησιμοποιεί μαύρα και γκρι κύτταρα σ' ένα δίκτυο προθέματος. Δουλεύει για σχεδιάσεις οποιασδήποτε τάξης, τόσο στατικές όσο και διαδοχικής επίδρασης (domino). Το αρχικό στάδιο PG και τα πρώτα επίπεδα του δικτύου προθέματος αντικαθίστανται από ένα κύτταρο το οποίο υπολογίζει απευθείας τα σήματα H και I της ομάδας. Το μεσαίο τμήμα του δικτύου προθέματος είναι πανομοιότυπο μ' ένα συνηθισμένο αθροιστή προθέματος, αλλά λειτουργεί με τα σήματα H και I αντί των G και P. Η λογική επιλογής αθροίσματος χρησιμοποιεί τον πολυπλέκτη από την Εξ. (11.22) αντί μιας XOR. Σε αραιά δένδρα, το άθροισμα από τα μπλοκ s βαθμίδων (bit) επιλέγεται απευθείας, βάσει των σημάτων H.

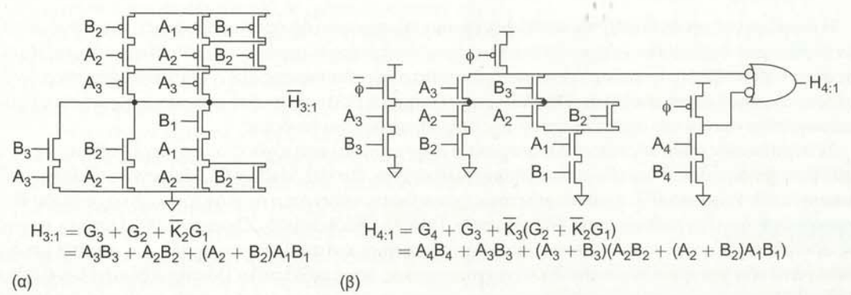
Για έναν αθροιστή τάξης v, η τεχνική Ling μετατρέπει μια πύλη σήματος γέννησης με v εν σειρά nMOS τρανζίστορ και v εν σειρά pMOS τρανζίστορ σε μια πύλη ψευδο-γέννησης με v–1 εν σειρά nMOS τρανζίστορ, αλλά v εν σειρά pMOS τρανζίστορ. Για παράδειγμα, σε 2η τάξη, η πύλη AOI γίνεται NOR2. Αυτό δεν είναι ιδιαίτερα χρήσιμο για κυκλώματα στατικής λογικής, αλλά είναι ευεργετικό για υλοποιήσεις domino, επειδή εξαλείφονται τα εν σειρά pMOS και ταυτόχρονα μικραίνουν οι σωροί των nMOS τρανζίστορ.

Ένα άλλο πλεονέκτημα της τεχνικής Ling είναι ότι επιτρέπει στο πρώτο επίπεδο σημάτων ψευδο-γέννησης και ψευδο-διάδοσης να υπολογίζονται απευθείας από τις εισόδους A_i και B_i αντί να βασίζονται στις πύλες G_i και K_i. Για παράδειγμα, το Σχήμα 11.36 συγκρίνει στατικές πύλες που υπολογίζουν τα $G_{2:1}$ και $H_{2:1}$ απευθείας από τα $A_{2:1}$ και $B_{2:1}$. Η πύλη H έχει ένα λιγότερο εν σειρά τρανζίστορ και πολύ λιγότερη παρασιτική χωρητικότητα. Το $H_{3:1}$ μπορεί επίσης να υπολογιστεί απευθείας από τα $A_{3:1}$ και $B_{3:1}$ χρησιμοποιώντας την πολύπλοκη στατική CMOS πύλη που παρουσιάζεται στο Σχήμα 11.37(α) [Quach92]. Παρόμοια, το Σχήμα 11.37(β) παρουσιάζει μια σύνθετη πύλη domino, η οποία υπολογίζει απευθείας το $H_{4:1}$ από τα A και B χρησιμοποιώντας μόνο τέσσερα εν σειρά τρανζίστορ αντί για τα πέντε που απαιτούνται για το $G_{4:1}$ [Naffziger96, Naffziger98].

Ο [Jackson04] πρότεινε την εφαρμογή της μεθόδου Ling με αναδρομικό τρόπο, για τη μεταφορά του σήματος K σε άλλο σημείο στο δένδρο του αθροιστή. Ο [Burgess09] έδειξε ότι αυτή η αναδρομική τεχνική Ling ανοίγει ένα νέο χώρο σχεδιαστικών επιλογών, για ταχύτερους και μικρότερους αθροιστές.



ΣΧΗΜΑ 11.36 2-bit πύλες για τα σήματα γέννησης και ψευδο-γέννησης, οι οποίες χρησιμοποιούν τις αρχικές εισόδους.



ΣΧΗΜΑ 11.37 3-bit & 4-bit πύλες για τα σήματα ψευδο-γέννησης, οι οποίες χρησιμοποιούν τις αρχικές εισόδους.

11.2.2.12 Σύντομη Αναφορά σε Ζητήματα Σχετικά με την Υλοποίηση Domino *Αυτή η ενότητα περιλαμβάνεται στην όλη που είναι διαθέσιμη online, μέσω του συνδέσμου «Web Enhanced», στον ιστότοπο* `www.cmosvlsi.com`.

11.2.2.13 Σύνοψη Μετά από την εξέταση μιας μεγάλης ποικιλίας αθροιστών, αυτό που πιθανότατα σας ενδιαφέρει είναι να ξέρετε ποιος είναι ο κατάλληλος αθροιστής για κάθε εφαρμογή. Ο Πίνακας 11.3 επιχειρεί μια συγκριτική παρουσίαση των διάφορων αρχιτεκτονικών αθροιστών, για τις οποίες έχουμε υποθέσει δίκτυα με προθέματα 2ης τάξης. Η στήλη «Επίπεδα Λογικής» δίνει τον αριθμό των πυλών AND-OR του κρίσιμου μονοπατιού, χωρίς να συνυπολογίζεται η αρχική λογική PG και η τελική πύλη XOR. Φυσικά, η καθυστέρηση εξαρτάται από το βαθμό οδήγησης εξόδου και από τα φορτία των αγωγών, καθώς επίσης και από το πλήθος των επιπέδων λογικής. Η στήλη «Κύτταρα» αναφέρει (κατά προσέγγιση) το πλήθος των γκρι και μαύρων κυττάρων του δικτύου. Η πρόβλεψη κρατουμένου δεν αναφέρεται στον πίνακα, επειδή χρησιμοποιεί κύτταρα μεγαλύτερης τάξης. Η επιλογή κρατουμένου επίσης δε αναφέρεται, επειδή είναι μεγαλύτερη από την προσαύξηση κρατουμένου για την ίδια απόδοση.

Γενικά, οι αθροιστές κύματος κρατουμένου πρέπει να προτιμώνται όταν ικανοποιούν τους περιορισμούς χρονισμού, επειδή χρησιμοποιούν τη λιγότερη δυνατή ενέργεια και είναι εύκολοι στην κατασκευή. Όταν απαιτούνται ταχύτεροι αθροιστές, οι αρχιτεκτονικές προσαύξησης κρατουμένου και παράκαμψης κρατουμένου είναι καλές επιλογές, ιδιαίτερα για 8-16 bit. Οι υβριδικοί αθροιστές που συνδυάζουν αυτές τις τεχνικές είναι επίσης δημοφιλείς. Για μήκος λέξης 32 bit και ιδιαίτερα 64 bit, οι αθροιστές δένδρου είναι σαφώς ταχύτεροι.

ΠΙΝΑΚΑΣ 11.3 Συγκριτική παρουσίαση αρχιτεκτονικών αθροιστών

Αρχιτεκτονική	Ταξινόμηση	Επίπεδα Λογικής	Μέγιστο fanout	Μονοπάτια	Κύτταρα
Διάδοσης κρατουμένου		$N-1$	1	1	N
Παράκαμψης κρατουμένου ($n = 4$)		$N/4+5$	2	1	$1.25N$
Προσαύξησης κρατουμένου ($n = 4$)		$N/4+2$	4	1	$2N$
Προσαύξησης κρατουμένου (μεταβλητή ομάδα)		$\sqrt{2N}$	$\sqrt{2N}$	1	$2N$
Brent-Kung	$(L-1, 0, 0)$	$2\log_2 N-1$	2	1	$2N$
Sklansky	$(0, L-1, 0)$	$\log_2 N$	$N/2+1$	1	$0.5\,N\log_2 N$
Kogge-Stone	$(0, 0, L-1)$	$\log_2 N$	2	$N/2$	$N\log_2 N$
Han-Carlson	$(1, 0, L-2)$	$\log_2 N+1$	2	$N/4$	$0.5\,N\log_2 N$
Ladner Fischer ($l = 1$)	$(1, L-2, 0)$	$\log_2 N+1$	$N/4+1$	1	$0.25\,N\log_2 N$
Knowles [2,1,…,1]	$(0, 1, L-2)$	$\log_2 N$	3	$N/4$	$N\log_2 N$

Η διαμάχη για την ανάδειξη της καλύτερης σχεδίασης αθροιστή δένδρου καλά κρατεί· η επιλογή μεταξύ των διαθέσιμων σχεδιάσεων επηρεάζεται από παράγοντες όπως οι περιορισμοί στην κατανάλωση ισχύος και στην καθυστέρηση, η επιλογή domino έναντι στατικής υλοποίησης και η συγκεκριμένη τεχνολογία κατασκευής που χρησιμοποιείται. Επιπλέον, η μελετημένη βελτιστοποίηση μιας συγκεκριμένης αρχιτεκτονικής είναι ακόμα πιο σημαντική από την αρχιτεκτονική του δένδρου.

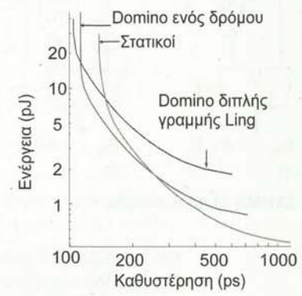
Σε περιπτώσεις όπου δεν τίθενται περιορισμοί στην κατανάλωση ισχύος, οι ταχύτεροι αθροιστές είναι αυτοί που χρησιμοποιούν κυκλώματα domino [Naffziger96, Park00, Mathew03, Mathew05, Oklobdzija05, Zlatanovici09, Wijeratne07]. Αρκετοί μελετητές έχουν διαπιστώσει ότι η αρχιτεκτονική Kogge-Stone δίνει τη μικρότερη δυνατή καθυστέρηση [Silberman98, Park00, Oklobdzija05, Zlatanovici09]. Ωστόσο, ο μεγάλος αριθμός αγωγών μεγάλου μήκους οδηγεί σε σημαντική κατανάλωση ενέργειας και απαιτεί μεγάλα κυκλώματα οδήγησης για ταχύτητα. Άλλες αρχιτεκτονικές, όπως οι Sklansky [Mathew03] και Han-Carlson [Vangal02], παρέχουν καλύτερη απόδοση όσον αφορά την ενέργεια, επειδή χρησιμοποιούν λιγότερους μεγάλου μήκους αγωγούς διασύνδεσης. Οι δυναμικές πύλες 4ης τάξης που ακολουθούνται από αντιστροφείς

δίνουν συνήθως ένα μικρό πλεονέκτημα ταχύτητας [Naffziger96, Park00, Zlatanovici09, Harris04, Oklobdzija05], αλλά υπάρχουν επίσης σύνθετες υλοποιήσεις domino που χρησιμοποιούν δυναμικές πύλες 2ης τάξης ακολουθούμενες από 2ης τάξης στατικές πύλες Y-απόκλισης [Mathew03]. Τα αραιά δένδρα εξοικονομούν ενέργεια σε αθροιστές υλοποίησης domino, έχοντας μικρή επίδραση στην απόδοση [Naffziger96, Mathew03, Zlatanovici09].

Η τεχνική βελτιστοποίησης του Ling δεν χρησιμοποιείται καθολικά, αλλά αρκετές μελέτες έχουν διαπιστώσει ότι είναι επωφελής [Quach92, Naffziger96, Zlatanovici09, Grad04]. Ο μικροεπεξεργαστής UltraSparc III χρησιμοποιεί έναν αθροιστή Kogge-Stone σε υλοποίηση domino διπλής γραμμής [Heald00]. Οι σειρές 64-bit μικροεπεξεργαστών Itanium 2 και PA-RISC της Hewlett Packard χρησιμοποιούν αθροιστή Ling σε υλοποίηση domino διπλής γραμμής [Naffziger96, Fetzer02]. Ο Pentium 4 στα 65 nm χρησιμοποιεί αρχιτεκτονική αραιού δένδρου Sklansky 2η τάξης σε υλοποίηση domino [Wijeratne07]. Μια καλή υλοποίηση domino για έναν 64-bit αθροιστή απαιτεί 7–9 καθυστερήσεις FO4 και έχει επιφάνεια 4–12 $M\lambda^2$ [Naffziger96, Zlatanovici09, Mathew05].

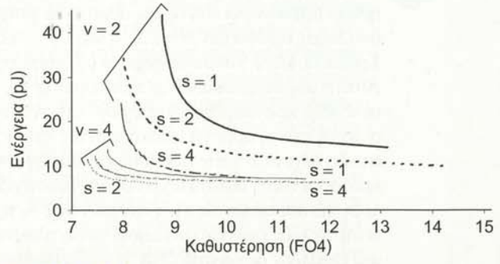
Οι σχεδιάσεις που πρέπει να συμμορφώνονται με περιορισμούς κατανάλωσης ισχύος χρησιμοποιούν στατικούς αθροιστές, οι οποίοι καταναλώνουν το ένα τρίτο έως το ένα δέκατο της ενέργειας που απαιτούν οι δυναμικοί αθροιστές και επιδεικνύουν καθυστέρηση περίπου 13 FO4 [Oklobdzija05, Harris03, Zlatanovici09]. Για παράδειγμα, η μονάδα κινητής υποδιαστολής του επεξεργαστή CELL χρησιμοποιεί έναν στατικό αθροιστή Kogge-Stone 2ης τάξης [Oh06].

Ο Patil [Patil07] παρουσιάζει μια εκτενή μελέτη του βασιζόμενου στο δίπολο ενέργεια-καθυστέρηση χώρου σχεδιαστικών επιλογών για αθροιστές. Η εργασία του ολοκληρώνεται με το συμπέρασμα ότι η αρχιτεκτονική Sklansky είναι η πλέον αποτελεσματική από άποψη ενέργειας για οποιαδήποτε δεδομένη απαίτηση καθυστέρησης, επειδή αποφεύγει το μεγάλο αριθμό αγωγών (που αυξάνουν την κατανάλωση ισχύος) της αρχιτεκτονικής Kogge-Stone και τον υπερβολικό αριθμό επιπέδων λογικής της αρχιτεκτονικής Brent-Kung. Οι πύλες με υψηλό fanout στο δένδρο Sklansky προσαρμόζονται προς τα επάνω, ώστε ο λογικός φόρτος να διατηρείται σε αποδεκτό επίπεδο. Οι στατικοί αθροιστές είναι πιο αποτελεσματικοί με κύτταρα 2ης τάξης, τα οποία παρέχουν φόρτο σταδίου ίσο περίπου με 4. Οι αθροιστές υλοποίησης domino είναι πιο αποτελεσματικοί όταν χρησιμοποιούνται εναλλάξ δυναμικές πύλες 4ης τάξης και στατικοί αντιστροφείς. Η λογική για τον προ-υπολογισμό του αθροίσματος σ' έναν στατικό αθροιστή αραιού δένδρου κοστίζει περισσότερο σε ενέργεια απ' όσο εξοικονομεί από το δίκτυο προθέματος. Σ' έναν αθροιστή υλοποίησης domino, ο συντελεστής αραιότητας 2 εξοικονομεί ενέργεια επειδή ο προ-υπολογισμός του αθροίσματος μπορεί να εκτελείται με στατικές πύλες. Το Σχήμα 11.38 παρουσιάζει ορισμένα αποτελέσματα, τα οποία υποδεικνύουν ότι οι στατικοί αθροιστές είναι πιο αποδοτικοί ενεργειακά όταν χρησιμοποιούνται για αργούς αθροιστές, ενώ οι υλοποιήσεις domino γίνονται προτιμότερες όταν υπάρχουν απαιτήσεις υψηλής ταχύτητας· επιπλέον, οι αθροιστές Ling σε υλοποιήσεις domino διπλής γραμμής είναι προτιμητέοι μόνο για εφαρμογές υπερυψηλής ταχύτητας και εξίσου υψηλής κατανάλωσης ενέργειας. Οι πολύ μικροί χρόνοι καθυστέρησης επιτυγχάνονται με τη χρήση μεγαλύτερης V_{DD} και μικρότερης V_t.



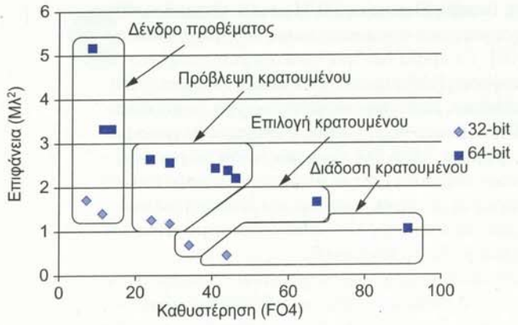
ΣΧΗΜΑ 11.38 Συμβιβασμοί μεταξύ ενέργειας και καθυστέρησης για 32-bit στατικούς αθροιστές Sklansky, domino και domino διπλής γραμμής (τεχνολογία κατασκευής 90 nm, με 31 ps καθυστέρηση αντιστροφέα FO4 στο 1.0 V και ονομαστική V_t [© IEEE 2007]).

Ο [Zlatanovici09] διερεύνησε το βασιζόμενο στο δίπολο ενέργεια-καθυστέρηση χώρο σχεδιαστικών επιλογών για domino υλοποιήσεις αθροιστών στα 64 bit και κατέληξε στο συμπέρασμα ότι η αρχιτεκτονική Kogge-Stone είναι ανώτερη. Και σ' αυτή την περίπτωση, η εναλλάξ χρήση δυναμικών πυλών 4ης τάξης και στατικών αντιστροφέων και η χρήση συντελεστή αραιότητας 2 έδωσε τα καλύτερα αποτελέσματα, όπως υποδεικνύει το Σχήμα 11.39. Υπάρχουν κι άλλες καλές σχεδιαστικές επιλογές για αθροιστές σ' αυτό το χώρο, πράγμα το οποίο σημαίνει ότι δεν υπάρχει κάποιος ισχυρός λόγος προτίμησης μιας ορισμένης τοπολογίας έναντι κάποιας άλλης και η διαμάχη για τον «καλύτερο» αθροιστή μάλλον θα συνεχιστεί και στο μέλλον.

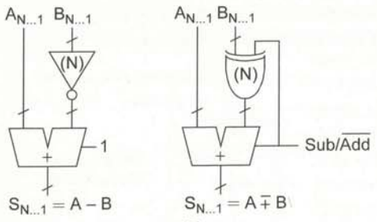


ΣΧΗΜΑ 11.39 Συμβιβασμοί μεταξύ ενέργειας και καθυστέρησης για 64-bit αθροιστές domino Kogge-Stone και Ling, βάσει τάξης (v) και αραιότητας (s). (© IEEE 2009).

Τα καλά εργαλεία σύνθεσης λογικής αντιστοιχίζουν αυτόματα τον τελεστή «+» σ' έναν κατάλληλο αθροιστή, για την ικανοποίηση των περιορισμών χρονισμού, ενώ ελαχιστοποιούν την απαιτούμενη επιφάνεια. Για παράδειγμα, οι βιβλιοθήκες DesignWare της Synopsys περιλαμβάνουν αθροιστές διάδοσης κρατουμένου, επιλογής κρατουμένου, πρόβλεψης κρατουμέ-



ΣΧΗΜΑ 11.40 Σχέση μεταξύ επιφάνειας και καθυστέρησης για αθροιστές δημιουργημένους με εργαλεία σύνθεσης



ΣΧΗΜΑ 11.41 Αφαιρέτες.

νου και μια ποικιλία αθροιστών προθέματος. Το Σχήμα 11.40 παρουσιάζει τα αποτελέσματα της σύνθεσης αθροιστών των 32 και των 64 bit, με διαφορετικούς περιορισμούς χρονισμού. Καθώς μειώνεται η καθυστέρηση, το εργαλείο σύνθεσης επιλέγει πολυπλοκότερους αθροιστές, με μεγαλύτερη επιφάνεια. Τα απεικονιζόμενα αποτελέσματα αφορούν μια εμπορική βιβλιοθήκη κυττάρων των 0.18 μm με 89 ps καθυστέρηση αντιστροφέα FO4 στη γωνία TTTT, ενώ οι εκτιμήσεις επιφάνειας συνυπολογίζουν τη διασύνδεση και τις πύλες. Οι ταχύτερες σχεδιάσεις χρησιμοποιούν αθροιστές δένδρου και επιτυγχάνουν απίθανα μικρές (προ φυσικού σχεδίου) καθυστερήσεις των 7.0 και 8.5 αντιστροφέων FO4 για αθροιστές των 32 και των 64 bit, αντίστοιχα, με τη δημιουργία μη-ομοιόμορφων σχεδιάσεων και τη σχολαστική απομόνωση των παράπλευρων φορτίων από το κρίσιμο μονοπάτι. Οι αθροιστές επιλογής κρατουμένου επιτυγχάνουν ενδιαφέροντα συμβιβασμό μεταξύ επιφάνειας και καθυστέρησης, με τη χρήση κύματος (διάδοσης) κρατουμένου για τα κατώτερα 3/4 του συνόλου των bit και επιλογής κρατουμένου για το υπόλοιπο 1/4 των bit υψηλότερης τάξης. Τα αποτελέσματα είναι σχετικά πιο αργά όταν συνυπολογίζονται οι παρασιτικές χωρητικότητες της διασύνδεσης.

11.2.3 Αφαίρεση

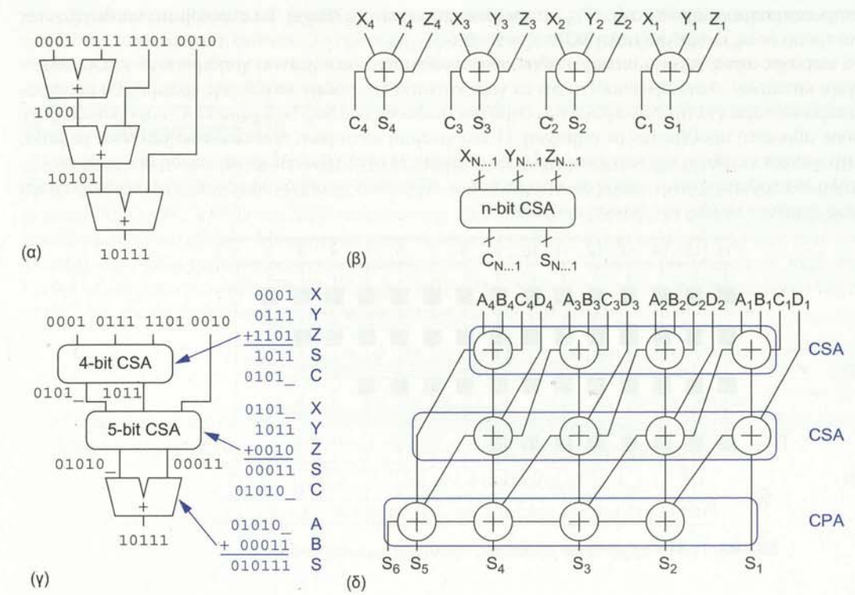
Ένας αφαιρέτης των N bit χρησιμοποιεί τη σχέση συμπληρώματος ως προς 2

$$A - B = A + \overline{B} + 1 \qquad \textbf{(11.23)}$$

Η σχέση αυτή απαιτεί την αντιστροφή του ενός τελεστέου σ' έναν αθροιστή διάδοσης κρατουμένου (CPA) και την πρόσθεση του 1 μέσω της εισόδου κρατουμένου, όπως απεικονίζεται στο Σχήμα 11.41(α). Ένας αθροιστής/αφαιρέτης χρησιμοποιεί πύλες XOR για την υπό συνθήκη αντιστροφή του B, όπως απεικονίζεται στο Σχήμα 11.41(β). Στους αθροιστές προσημασμένων αριθμών, οι πύλες XOR στις εισόδους B συγχωνεύονται ορισμένες φορές στο κύκλωμα PG.

11.2.4 Πρόσθεση Πολλαπλών Εισόδων

Η προφανέστερη μέθοδος πρόσθεσης k λέξεων των N bit είναι χρησιμοποιώντας k-1 αθροιστές διάδοσης κρατουμένου (CPA) εν σειρά, όπως απεικονίζεται στο Σχήμα 11.42(α) για την πράξη 0001+0111+1101+0010. Ωστόσο, η μέθοδος αυτή απαιτεί μεγάλη ποσότητα hardware και είναι αργή. Μια καλύτερη τεχνική βασίζεται στην παρατήρηση ότι ένας πλήρης αθροιστής μπορεί να προσθέτει τρεις εισόδους της ίδιας αξίας και να παράγει ένα αποτέλεσμα αθροίσματος της ίδιας αξίας κι ένα αποτέλεσμα κρατουμένου διπλάσιας αξίας. Εάν χρησιμοποιηθούν παράλληλα N πλήρεις αθροιστές, μπορούν να δεχθούν τρεις λέξεις εισόδου των N bit και να παράγουν δύο λέξεις εξόδου των N bit, $S_{N...1}$ και $C_{N...1}$, ικανοποιώντας τη σχέση $X + Y + Z = S + 2C$, όπως παρουσιάζεται στο Σχήμα 11.42(β). Τα αποτελέσματα αντιστοιχούν στα αθροίσματα και στα κρατούμενα εξόδου του κάθε αθροιστή. Αυτή η μορφή αποκαλείται *πλεονασματική μορφή αθροίσματος-κρατουμένου* (carry-save redundant format), επειδή οι έξοδοι κρατουμένου διατηρούνται αντί να διαδίδονται κατά μήκος του αθροιστή. Οι πλήρεις αθροιστές σ' αυτή την εφαρμογή αποκαλούνται επίσης *[3:2] αθροιστές αποθήκευσης κρατουμένου* (carry-save adder, CSA), επειδή δέχονται τρεις εισόδους και παράγουν δύο εξόδους σε μορφή αθροίσματος-κρατουμένου. Όταν η λέξη κρατουμένου C υφίσταται ολίσθηση αριστερά κατά μια θέση (επειδή είναι διπλάσιας αξίας) και προστίθεται στη λέξη του αθροίσματος S μ' έναν κοινό CPA, το αποτέλεσμα είναι X+Y+Z. Εναλλακτικά, μπορεί να προστεθεί μια τέταρτη λέξη εισόδου στο αποτέλεσμα πλεονασματικής μορφής αθροίσματος-κρατουμένου, χρησιμοποιώντας μια επιπλέον σειρά από CSA, η οποία θα δίνει επίσης ένα αποτέλεσμα σε πλεονασματική μορφή αθροίσματος-κρατουμένου. Μια τέτοια πρόσθεση τεσσάρων αριθμών σε μορφή αθροίσματος-κρατουμένου απεικονίζεται στο Σχήμα 11.42(γ), όπου οι υπογραμμίσεις στις εξόδους κρατουμένου υποδεικνύουν ότι τα κρατούμενα πρέπει να ολισθαίνουν αριστερά κατά μια στήλη, λόγω της μεγαλύτερης αξίας τους.



ΣΧΗΜΑ 11.42 Αθροιστές πολλαπλών εισόδων.

Το κρίσιμο μονοπάτι που διατρέχει έναν αθροιστή [3:2] χρησιμοποιείται για τον υπολογισμό του αθροίσματος, ο οποίος γίνεται με μια XOR 3 εισόδων, ή με δύο επίπεδα από XOR2. Η διάταξη αυτή είναι πολύ ταχύτερη από έναν CPA. Γενικά, μπορούν να προστίθενται k αριθμοί χρησιμοποιώντας k-2 αθροιστές CSA [3:2] κι έναν μόνο CPA. Θα εκμεταλλευτούμε αυτή την προσέγγιση στην Ενότητα 11.9 για τη γρήγορη πρόσθεση πολλών μερικών γινομένων σ' έναν πολλαπλασιαστή. Η τεχνική αυτή χρονολογείται από την εποχή του πρώτου υπολογιστή του von Neumann [Burks46].

Σε περιπτώσεις όπου μία από τις εισόδους ενός CSA είναι σταθερή, μπορεί να μειωθεί ακόμη περισσότερο το απαιτούμενο υλικό (hardware). Εάν ένα bit της εισόδου είναι 0, η στήλη CSA απλοποιείται σ' έναν ημιαθροιστή. Εάν το εν λόγω bit είναι 1, η στήλη CSA απλοποιείται σε $S = \overline{A \oplus B}$ και $C = A + B$.

11.2.5 Αθροιστές Προθέματος με Σήμανση

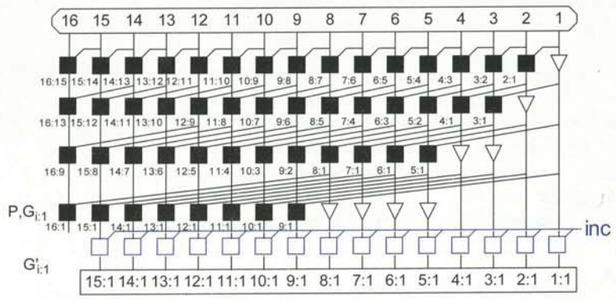
OPTIONAL

Ορισμένες φορές είναι αναγκαίο να υπολογιστεί είτε το $A + B$ και κατόπιν, ανάλογα μ' ένα σήμα ελέγχου που φτάνει αργά, να προστεθεί το 1. Αυτό είναι χρήσιμο για τον υπολογισμό του $A + B$ mod 2^n–1 σε εφαρμογές κρυπτογραφίας και κωδικοποίησης Reed-Solomon, για τον υπολογισμό της απόλυτης διαφοράς $|A - B|$, για την πρόσθεση/αφαίρεση αριθμών σε μορφή προσήμου-μέτρου και για την εκτέλεση στρογγυλοποίησης σε συγκεκριμένους αθροιστές κινητής υποδιαστολής [Beaumont-Smith99]. Μια απλή προσέγγιση συνίσταται στην κατασκευή δύο αθροιστών, την παροχή ενός κρατουμένου στον ένα και την επιλογή μεταξύ των αποτελεσμάτων. Ο [Burgess02] περιγράφει μια έξυπνη εναλλακτική λύση, τον αποκαλούμενο *αθροιστή προθέματος με σήμανση* (flagged prefix adder), η οποία χρησιμοποιεί πολύ λιγότερο hardware.

Ένας αθροιστής προθέματος με σήμανση λαμβάνει τα A, B κι ένα σήμα ελέγχου *inc*, και υπολογίζει το $A + B + inc$. Όπως θυμάστε, ένας συμβατικός αθροιστής υπολογίζει τα προθέματα $G_{i-1:0}$ σαν κρατούμενα σε κάθε στήλη i, και κατόπιν υπολογίζει το άθροισμα $S_i = P_i \oplus G_{i-1:0}$. Σ' αυτή την περίπτωση δεν υπάρχει κρατούμενο εισόδου, C_{in}, οπότε η στήλη 0 παραλείπεται· αντ' αυτής χρησιμοποιείται το $G_{i-1:1}$. Ο στόχος του αθροιστή προθέματος με σήμανση είναι να προσαρμόζει αυτά τα κρατούμενα όταν λαμβάνεται το σήμα *inc*. Ένας αθροιστής προθέματος με σήμανση χρησιμοποιεί το $G'_{i-1:1} = G_{i-1:1} + P_{i-1:1} \cdot inc$. Συνεπώς, εάν το *inc* είναι true, παράγει ένα κρατούμενο σε όλα τα bit χαμηλής τάξης των οποίων τα σήματα διάδοσης ομάδας είναι

TRUE. Τα τροποποιημένα προθέματα, $G'_{i:1:1}$, αποκαλούνται *σημάνσεις* (flags). Τα αθροίσματα υπολογίζονται με τον ίδιο τρόπο όπως και με μια πύλη XOR: $S_i = P_i \oplus G'_{i-1:1}$.

Για να παράγει αυτές τις σημάνσεις, ο αθροιστής προθέματος με σήμανση χρησιμοποιεί μία επιπλέον γραμμή γκρι κυττάρων. Αυτό προϋποθέτει ότι τα γκρι κύτταρα της πρώην κατώτερης γραμμής θα μετατραπούν σε μαύρα κύτταρα για την παραγωγή των σημάτων διάδοσης ομάδας. Το Σχήμα 11.43 παρουσιάζει έναν Kogge-Stone αθροιστή προθέματος με σήμανση. Η νέα γραμμή κυττάρων, η οποία επισημαίνεται με μπλε, προστίθεται για την εκτέλεση της «αργοπορημένης» αύξησης. Η στήλη 0 εξαλείφεται επειδή δεν υπάρχει C_{in}, αλλά η στήλη 16 περιλαμβάνεται επειδή οι εφαρμογές των αθροιστών με σήμανση χρειάζονται την παραγωγή και διάδοση σημάτων σε όλη την έκταση των n bit.



ΣΧΗΜΑ 11.43 Kogge-Stone αθροιστής προθέματος με σήμανση.

11.2.5.1 Πρόσθεση Modulo $2_n - 1$ Για τον υπολογισμό της πράξης $A + B \bmod 2^n - 1$ με μη-προσημασμένους τελεστέους, ένας αθροιστής θα πρέπει πρώτα να υπολογίσει το άθροισμα $A + B$. Εάν το άθροισμα είναι μεγαλύτερο από ή ίσο με $2^n - 1$, το αποτέλεσμα θα πρέπει να αυξηθεί και κατόπιν να περικοπεί σε n bits. Το $G_{n:1}$ είναι TRUE εάν ο αθροιστής υπερχειλίσει – δηλαδή, το αποτέλεσμα είναι μεγαλύτερο από $2^n - 1$. Το $P_{n:1}$ είναι TRUE εάν διαδίδουν όλες οι στήλες, πράγμα το οποίο συμβαίνει μόνο όταν το άθροισμα ισούται με $2^n - 1$. Συνεπώς, μπορούν να εκτελούνται προσθέσεις modulo μ' έναν αθροιστή προθέματος με σήμανση, χρησιμοποιώντας $inc = G_{n:1} + P_{n:1}$.

Σε σύγκριση με τη συνηθισμένη πρόσθεση, η πρόσθεση modulo απαιτεί μία επιπλέον γραμμή μαύρων κυττάρων, μια πύλη OR για τον υπολογισμό του inc κι έναν απομονωτή για την οδήγηση του inc σε όλα τα n bits.

11.2.5.2 Απόλυτη διαφορά Το αποτέλεσμα της πράξης $|A - B|$ αποκαλείται *απόλυτη διαφορά* και χρησιμοποιείται ευρέως σε εφαρμογές όπως η συμπίεση βίντεο. Η απλούστερη προσέγγιση συνίσταται στον υπολογισμό αμφότερων των $A - B$ και $B - A$ και κατόπιν την επιλογή του θετικού αποτελέσματος. Μια πιο αποτελεσματική τεχνική συνίσταται στον υπολογισμό του $A + \overline{B}$ και την εξέταση του προσήμου, το οποίο υποδεικνύεται από το $\overline{G}_{n:1}$. Εάν το αποτέλεσμα είναι αρνητικό, θα πρέπει να αντιστραφεί για να ληφθεί το $B - A$. Εάν το αποτέλεσμα είναι θετικό, θα πρέπει να αυξηθεί για να ληφθεί το $A - B$.

Όλες αυτές οι πράξεις μπορούν να εκτελούνται χρησιμοποιώντας μια «επαυξημένη» έκδοση του αθροιστή προθέματος με σήμανση, η οποία θα έχει δυνατότητα υπό συνθήκη αντιστροφής του αποτελέσματος. Η λογική του αθροίσματος τροποποιείται ώστε να υπολογίζεται το $S_i = (P_i \oplus inv) \oplus G'_{i-1:1}$. Επιλέγονται τα $inv = \overline{G}_{n:1}$ και $inc = G_{n:1}$.

Σε σύγκριση με τη συμβατική πρόσθεση, η απόλυτη διαφορά απαιτεί τη χρήση μιας συστοιχίας αντιστροφέων για τη λήψη του $\overline{B}$, μία επιπλέον γραμμή μαύρων κυττάρων, απομονωτές για την οδήγηση των inv και inc σε όλες τις n βαθμίδες και μια σειρά πυλών XOR για την υπό συνθήκη αντιστροφή του αποτελέσματος. Σημειώστε ότι το $(P_i \oplus inv)$ μπορεί να υπολογίζεται εκ των προτέρων, οπότε δεν επηρεάζει το κρίσιμο μονοπάτι.

11.2.5.3 Αριθμητική σε μορφή προσήμου-μέτρου Η πρόσθεση αριθμών σε μορφή προσήμου-μέτρου απαιτεί την εξέταση των προσήμων των τελεστέων. Εάν τα πρόσημα συμφωνούν, τα μέτρα των αριθμών προστίθενται και το πρόσημο παραμένει αμετάβλητο. Εάν τα πρόσημα διαφέρουν, πρέπει να υπολογιστεί η απόλυτη διαφορά

των μέτρων των αριθμών. Αυτό μπορεί να γίνει χρησιμοποιώντας τον αθροιστή με σήμανση που περιγράψαμε στην προηγούμενη ενότητα. Το πρόσημο του αποτελέσματος είναι $\text{sign}(A) \oplus G_{n:1}$.

Η αφαίρεση εκτελείται πανομοιότυπα, εκτός από το ότι αντιστρέφεται πρώτα το πρόσημο του B.

11.3 Ανιχνευτές 1/0

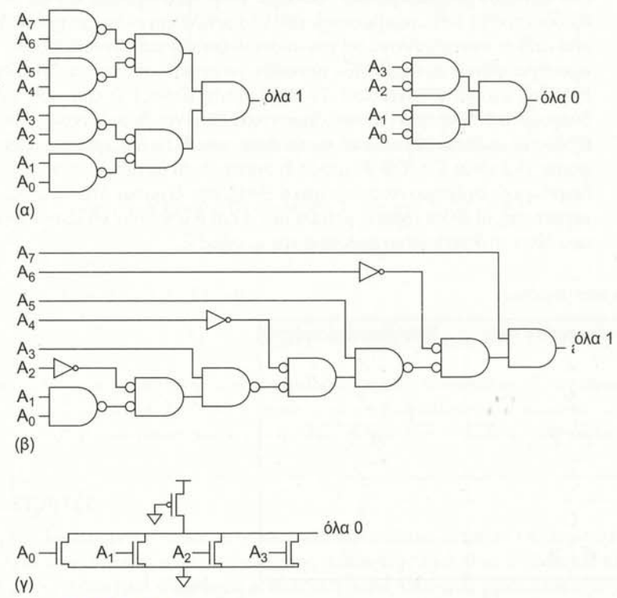
Η ανίχνευση όλων των ψηφίων 1 ή των 0 σε μεγάλες λέξεις των N bit απαιτεί πύλες AND ή NOR με μεγάλο βαθμό οδήγησης εισόδου (fan-in). Όπως γνωρίζετε από το νόμο DeMorgan, στο πλέον θεμελιώδες επίπεδο οι πύλες AND, OR, NAND και NOR εκτελούν την ίδια λειτουργία, εκτός από πιθανές αντιστροφές των εισόδων και/ή των εξόδων. Μπορείτε να κατασκευάσετε ένα δένδρο από πύλες AND, όπως αυτό του Σχ. 11.44(α), στο οποίο χρησιμοποιούνται εναλλάξ πύλες NAND και NOR. Το μονοπάτι έχει $\log N$ στάδια. Γενικά, ο ελάχιστος λογικός φόρτος επιτυγχάνεται μ' ένα δένδρο το οποίο χρησιμοποιεί εναλλάξ πύλες NAND και αντιστροφείς, ενώ ο λογικός φόρτος του μονοπατιού είναι:

$$G_{\text{and}}(N) = \left(\frac{4}{3}\right)^{\log_2 N} = N^{\log_2 \frac{4}{3}} = N^{0.415} \qquad \textbf{(11.24)}$$

Μια προσεγγιστική εκτίμηση της καθυστέρησης του μονοπατιού, κατά την οδήγηση ενός μονοπατιού με ηλεκτρικό φόρτο H και με χρήση στατικών πυλών CMOS, είναι η ακόλουθη

$$D \approx (\log_4 F)\, t_{FO4} = (\log_4 H + 0.415 \log_4 N)\, t_{FO4} \qquad \textbf{(11.25)}$$

όπου t_{FO4} είναι η καθυστέρηση ενός αντιστροφέα με βαθμό οδήγησης εξόδου 4.



ΣΧΗΜΑ 11.44 Ανιχνευτές 1/0.

Εάν η λέξη που ελέγχεται έχει φυσιολογική απόκλιση (skew) ως προς το χρόνο άφιξης των εξόδων (όπως αυτή στην έξοδο ενός αθροιστή κύματος κρατουμένου) ο σχεδιαστής θα μπορούσε να εξετάζει τη δυνατότητα χρήσης μιας ασύμμετρης σχεδίασης, η οποία ευνοεί τις πιο καθυστερημένες εισόδους, όπως

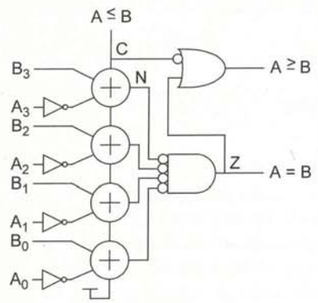
υποδεικνύει το Σχήμα 11.44(β). Εδώ, η καθυστέρηση από το τελευταία bit που αλλάζει, A_7, είναι μια καθυστέρηση μιας πύλης.

Ένας άλλος γρήγορος ανιχνευτής χρησιμοποιεί μια δομή ψευδο-nMOS ή δυναμική δομή NOR για την υλοποίηση της λογικής «καλωδιωμένης (wired) OR», όπως παρουσιάζεται στο Σχήμα 11.44(γ). Αυτό δουλεύει ικανοποιητικά για λέξεις μεγέθους έως 16 bit περίπου. Για λέξεις μεγαλύτερου μεγέθους, οι πύλες μπορούν να χωριστούν σε τμήματα των 8 έως 16 bit, για τη μείωση της καθυστέρησης των παρασιτικών στοιχείων και την αποφυγή προβλημάτων με τη διαρροή υποκατωφλίου.

11.4 Συγκριτές

11.4.1 Συγκριτής Μεγέθους

Ένας συγκριτής μεγέθους (μέτρου, magnitude) συγκρίνει δύο δυαδικούς αριθμούς και βρίσκει το μεγαλύτερο. Για να συγκρίνουμε δύο μη-προσημασμένους αριθμούς A και B, υπολογίζουμε την έκφραση $B-A = B+\overline{A}+1$. Εάν υπάρχει κρατούμενο εξόδου, τότε $A \le B$· διαφορετικά, $A > B$. Ένας ανιχνευτής μηδενός υποδεικνύει ότι οι αριθμοί είναι ίσοι. Το Σχήμα 11.45 παρουσιάζει ένα συγκριτή μη-προσημασμένων αριθμών των 4 bit, υλοποιημένο μ' έναν αθροιστή κύματος κρατουμένου κι ένα μετατροπέα σε συμπλήρωμα ως προς 2. Το σχετικό μέγεθος καθορίζεται από το σήμα κρατουμένου εξόδου (C) και το σήμα μηδέν (Z), σύμφωνα με τον Πίνακα 11.4. Για μεγαλύτερους αριθμούς, μπορεί να χρησιμοποιηθεί οποιαδήποτε από τις αρχιτεκτονικές ταχύτερων αθροιστών.



ΣΧΗΜΑ 11.45 Συγκριτής μεγέθους μη-προσημασμένων αριθμών.

Η σύγκριση προσημασμένων αριθμών σε συμπλήρωμα ως προς 2 είναι ελαφρώς πολυπλοκότερη, λόγω της πιθανότητας υπερχείλισης (overflow) κατά την αφαίρεση δύο αριθμών με διαφορετικά πρόσημα. Αντί να ελέγχουμε απλώς το κρατούμενο εξόδου, πρέπει να εξακριβώσουμε εάν το αποτέλεσμα είναι αρνητικό (N, υποδεικνύεται από το πιο σημαντικό bit του αποτελέσματος) και εάν υπερβαίνει το εύρος των προσημασμένων αριθμών που μπορούν να παρασταθούν για το δεδομένο πλήθος bits (δηλ. έχουμε υπερχείλιση). Το σήμα υπερχείλισης V είναι 1 εάν οι είσοδοι έχουν διαφορετικά πρόσημα (τα πιο σημαντικά bit είναι διαφορετικά) και το πρόσημο εξόδου είναι διαφορετικό από το πρόσημο του B. Το πραγματικό πρόσημο της διαφοράς B-A είναι $S = N \oplus V$, επειδή η υπερχείλιση αντιστρέφει το πρόσημο. Εάν το διορθωμένο πρόσημο είναι αρνητικό (S=1), τότε ξέρουμε ότι $A > B$. Και σ' αυτή την περίπτωση, οι άλλες σχέσεις μεταξύ των A και B μπορούν να εξαχθούν από το σήμα που δίνει το διορθωμένο πρόσημο και το σήμα Z.

ΠΙΝΑΚΑΣ 11.4 Σύγκριση μεγέθους

Σχέση	Μη προσημασμένες τιμές	Προσημασμένες τιμές
$A = B$	Z	Z
$A \neq B$	$\overline{Z}$	$\overline{Z}$
$A < B$	$C \cdot \overline{Z}$	$\overline{S} \cdot \overline{Z}$
$A > B$	$\overline{C}$	S
$A \le B$	C	$\overline{S}$
$A \ge B$	$\overline{C} + Z$	$S + Z$

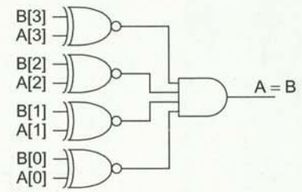
11.4.2 Συγκριτής Ισότητας

Ένας *συγκριτής ισότητας* καθορίζει εάν ($A = B$). Αυτό μπορεί να επιτευχθεί ταχύτερα και απλούστερα με πύλες XNOR κι έναν ανιχνευτή των 1, όπως παρουσιάζεται στο Σχήμα 11.46.

11.4.3 Συγκριτής K = A+B

OPTIONAL

Σε ορισμένες περιπτώσεις είναι αναγκαίο να καθοριστεί εάν ($A+B = K$). Για παράδειγμα, η διευθυνσιοδοτούμενη από άθροισμα μνήμη (βλ. Ενότητα 12.2.2.4) περιλαμβάνει έναν αποκωδικοποιητή, το αποτέλεσμα του οποίου πρέπει να ελέγχεται έναντι του αθροίσματος δύο αριθμών, όπως π.χ. μιας διεύθυνσης βάσης και μιας μετατόπισης (offset). Σημειώστε ότι αυτή η σύγκριση μπορεί να εκτελεστεί ταχύτερα από τον υπολογισμό του $A+B$, επειδή δεν απαιτείται διάδοση κρατουμένου. Το κλειδί είναι ότι εάν ξέρετε τα A και B, ξέρετε επίσης και ποιο θα πρέπει να είναι το κρατούμενο εισόδου για κάθε bit, εάν $K = A+B$ [Cortadella92]. Συνεπώς, χρειάζεται να ελεγχθούν μόνο γειτονικά ζεύγη bit για να επαληθευτεί ότι το προηγούμενο bit παράγει το κρατούμενο που απαιτείται από το τρέχον, και κατόπιν να χρησιμοποιηθεί ένας ανιχνευτής των 1 για να ελεγχθεί ότι η συνθήκη ισότητας είναι αληθής για όλα τα ζεύγη N. Συγκεκριμένα, εάν $K = A+B$, ο Πίνακας 11.5 υποδεικνύει ποιο θα πρέπει να είναι το κρατούμενο εισόδου c_{i-1} ώστε αυτή η συνθήκη να είναι αληθής για κάθε bit i και ποιο θα πρέπει να είναι το κρατούμενο εξόδου c_i για κάθε θέση bit, i.



ΣΧΗΜΑ 11.46 Συγκριτής ισότητας.

ΠΙΝΑΚΑΣ 11.5 Απαιτούμενα και παραγόμενα κρατούμενα εάν $K = A + B$

A_i	B_i	K_i	c_{i-1} (απαιτούμενο)	c_i (παραγόμενο)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Από αυτό τον πίνακα βλέπουμε ότι το απαιτούμενο c_{i-1} για το bit i είναι:

$$c_{i-1} = A_i \oplus B_i \oplus K_i \tag{11.26}$$

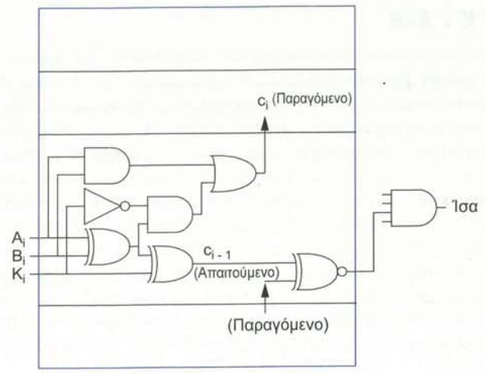
και το παραγόμενο c_{i-1} από το bit i-1 είναι:

$$c_{i-1} = \left(A_{i-1} \oplus B_{i-1}\right)\overline{K}_{i-1} + A_{i-1} \cdot B_{i-1} \tag{11.27}$$

Το Σχήμα 11.47 παρουσιάζει μέρος ενός κυκλώματος που υλοποιεί αυτή την πράξη. Η πύλη XNOR χρησιμοποιείται για να διασφαλίσει ότι το απαιτούμενο κρατούμενο ταιριάζει με το παραγόμενο κρατούμενο σε κάθε θέση bit· στη συνέχεια, η πύλη AND ελέγχει εάν η συνθήκη ικανοποιείται για όλα τα bit.

11.5 Μετρητές

Δύο κοινά χρησιμοποιούμενοι τύποι μετρητών είναι οι *δυαδικοί μετρητές* (binary counters) και οι καταχωρητές ολίσθησης γραμμικής ανατροφοδότησης (linear-feedback shift register). Ένας δυαδικός μετρητής των N bit μετρά διαδοχικά 2^N εξόδους σε δυαδική σειρά. Οι απλές σχεδιάσεις έχουν ελάχιστη περίοδο ρολογιού που αυξάνεται με το N, αλλά οι ταχύτερες σχεδιάσεις λειτουργούν σε σταθερό χρόνο. Ένας N-bit καταχωρητής ολίσθησης γραμμικής ανατροφοδότησης μετρά διαδοχικά έως 2^N-1 εξόδους σε ψευδοτυχαία σειρά. Έχει μικρό χρόνο κύκλου, ανεξάρτητο του N, οπότε είναι χρήσιμος τόσο για εξαιρετικά γρήγορους μετρητές, όσο και για ψευδοτυχαία παραγωγή αριθμών.



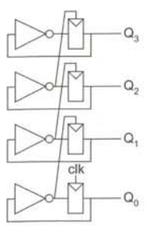
ΣΧΗΜΑ 11.47 Συγκριτής $A+B = K$.

Ορισμένα κοινά χαρακτηριστικά των μετρητών είναι τα ακόλουθα:

- *Επαναφορά* (*resettable*): η τιμή του μετρητή μηδενίζεται όταν επιβεβαιώνεται ένα σήμα επαναφοράς, *RESET* (σημαντικό για σκοπούς ελέγχου)
- *Φόρτωση* (*loadable*): ο μετρητής «φορτώνεται» με μια *N*-bit τιμή όταν επιβεβαιώνεται ένα σήμα φόρτωσης, *LOAD*
- *Επίτρεψη (enable)*: ο μετρητής μετρά μόνο κατά τους κύκλους του ρολογιού όπου επιβεβαιώνεται ένα σήμα επίτρεψης, *EN*
- *Αντιστροφή μέτρησης*: ο μετρητής μετρά προς τα επάνω ή προς τα κάτω, ανάλογα με το εάν λάβει σήμα εισόδου $\overline{UP}$ *ή DOWN*
- *Τερματισμός μέτρησης*: δίνεται σήμα εξόδου *TC* όταν συμβαίνει υπερχείλιση (κατά τη μέτρηση προς τα επάνω) ή υποχείλιση (κατά τη μέτρηση προς τα κάτω)

Γενικά, μετρητές *M*-άδων (όπου $M < 2^N$) μπορούν να κατασκευάζονται χρησιμοποιώντας ένα συμβατικό *N*-bit μετρητή κι ένα κύκλωμα για την επανεκκίνηση του μετρητή όταν φτάσει στην τιμή *M*. Το *M* μπορεί να είναι μια προγραμματιζόμενη είσοδος, εάν χρησιμοποιηθεί ένας συγκριτής ισότητας. Εναλλακτικά, μπορεί να χρησιμοποιηθεί ένας μετρητής με δυνατότητα φόρτωσης τιμής που θα επανεκκινεί από την τιμή $N - M$ οποτεδήποτε ένα σήμα *TC* υποδεικνύει υπερχείλιση στο μετρητή.

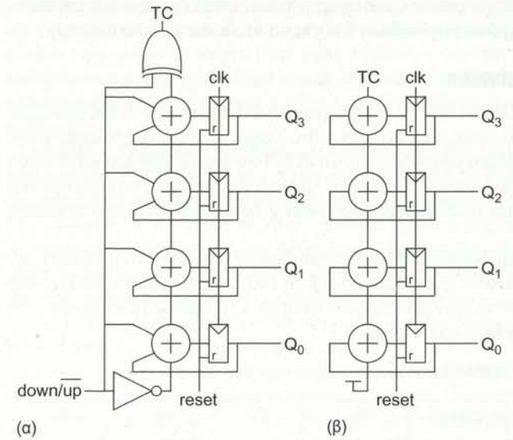
11.5.1 Δυαδικοί Μετρητές



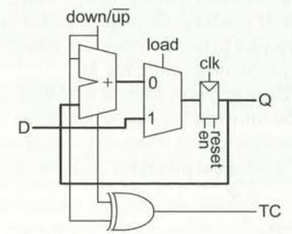
ΣΧΗΜΑ 11.48 Ασύγχρονος μετρητής κύματος κρατουμένου.

Ο απλούστερος δυαδικός μετρητής είναι ένας ασύγχρονος μετρητής κύματος κρατουμένου, όπως παρουσιάζεται στο Σχήμα 11.48. Αποτελείται από *N* καταχωρητές συνδεδεμένους με τρόπο ώστε η καθοδική μετάβαση της κατάστασης κάθε καταχωρητή να αποτελεί σήμα ρολογιού για τον αμέσως επόμενο. Αυτό σημαίνει ότι η καθυστέρηση μπορεί να γίνει αρκετά μεγάλη. Δεν έχει σήμα επαναφοράς (reset), γεγονός το οποίο καθιστά δύσκολο τον έλεγχό του. Γενικά, τα ασύγχρονα κυκλώματα παρουσιάζουν πολλά προβλήματα και γι' αυτό ο μετρητής κύματος κρατουμένου έχει κυρίως ιστορικό ενδιαφέρον και δεν είναι κατάλληλος για σχεδιάσεις εμπορικών συστημάτων.

Ένας σύγχρονος, γενικού σκοπού μετρητής επάνω/κάτω παρουσιάζεται στο Σχήμα 11.49(α). Χρησιμοποιεί έναν καταχωρητή με είσοδο reset κι έναν πλήρη αθροιστή για κάθε θέση bit. Η περίοδος του ρολογιού περιορίζεται από την καθυστέρηση κύματος (διάδοσης) κρατουμένου. Αν και θα μπορούσε να χρησιμοποιηθεί ένας ταχύτερος αθροιστής, στην επόμενη ενότητα θα περιγράψουμε έναν καλύτερο τρόπο κατασκευής γρήγορων μετρητών. Εάν απαιτείται μόνο ένας άνω μετρητής (αποκαλείται επίσης *προσαυξητής* [incrementer]), ο πλήρης αθροιστής μετασχηματίζεται σ' έναν ημιαθροιστή, όπως παρουσιάζεται στο Σχήμα 11.49(β). Η συμπερίληψη ενός πολυπλέκτη εισόδου επιτρέπει στο μετρητή να φορτώσει μια αρχική τιμή. Συχνά, παρέχεται επίσης ένα σήμα επίτρεψης ρολογιού (clock enable) σε κάθε καταχωρητή, για την εκτέλεση μέτρησης υπό συνθήκη. Το σήμα *TC* στην έξοδο υποδεικνύει ότι έχει συμβεί υπερχείλιση ή υποχείλιση. Το Σχήμα 11.50 παρουσιάζει έναν σύγχρονο μετρητή επάνω/κάτω με πλήρη γκάμα δυνατοτήτων (επανεκκίνηση, φόρτωση τιμής, επίτρεψη).



ΣΧΗΜΑ 11.49 Σύγχρονοι μετρητές.



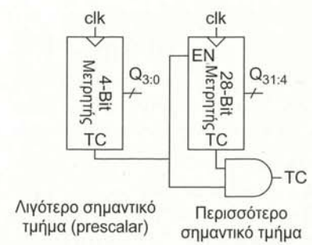
ΣΧΗΜΑ 11.50 Σύγχρονος μετρητής επάνω/κάτω με σήμα reset, load και enable.

11.5.2 Γρήγοροι Δυαδικοί Μετρητές

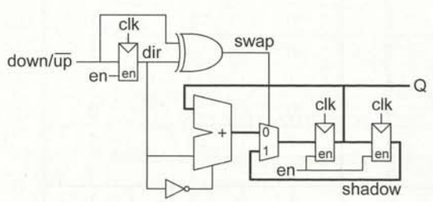
OPTIONAL

Η ταχύτητα του μετρητή του Σχήματος 11.49 περιορίζεται από την ταχύτητα του αθροιστή. Αυτό το πρόβλημα μπορεί να ξεπεραστεί διαχωρίζοντας το μετρητή σε δύο ή περισσότερα τμήματα [Ercegovac89]. Για παράδειγμα, ένας μετρητής των 32 bit θα μπορούσε να κατασκευαστεί από έναν *prescalar* μετρητή των 4 bit κι ένα μετρητή των 28 bit, όπως παρουσιάζεται στο Σχήμα 11.51. Η έξοδος *TC* του prescalar επιτρέπει την εκτέλεση μέτρησης στο περισσότερο σημαντικό τμήμα. Σ' αυτή την περίπτωση, ο χρόνος κύκλου περιορίζεται μόνο από την ταχύτητα του prescalar, επειδή ο αθροιστής των 28 bit έχει στη διάθεσή του 24 κύκλους για να παράγει ένα αποτέλεσμα. Χρησιμοποιώντας διαχωρισμό σε περισσότερα τμήματα, ένας μετρητής τυχαίου μήκους θα μπορούσε να λειτουργεί με την ταχύτητα ενός μετρητή των 1 ή 2 bit.

Η χρήση prescalar δεν επαρκεί για τους μετρητές επάνω/κάτω, επειδή το περισσότερο σημαντικό τμήμα μπορεί να έχει στη διάθεσή του μόνο έναν κύκλο για να ανταποκριθεί όταν ο μετρητής αλλάζει κατεύθυνση μέτρησης. Για την επίλυση αυτού του προβλήματος, μπορεί να χρησιμοποιηθεί ένας *σκιώδης καταχωρητής* (shadow register) στα περισσότερο σημαντικά τμήματα, για την κατακράτηση της προηγούμενης τιμής που θα πρέπει να χρησιμοποιηθεί όταν αλλάζει η κατεύθυνση μέτρησης [Stan98]. Το Σχήμα 11.52 παρουσιάζει το περισσότερο σημαντικό τμήμα για ένα γρήγορο μετρητή επάνω/κάτω. Όταν λαμβάνει σήμα reset (δεν παρουσιάζεται στο σχήμα), ο καταχωρητής *dir* τίθεται σε 0, η έξοδος *Q* σε 0 και ο καταχωρητής *shadow* σε –1. Όταν αλλάζει η κατεύθυνση μέτρησης ($\overline{UP}$/*DOWN*), δίνεται το σήμα εναλλαγής, *swap*, για



ΣΧΗΜΑ 11.51 Γρήγορος δυαδικός μετρητής.



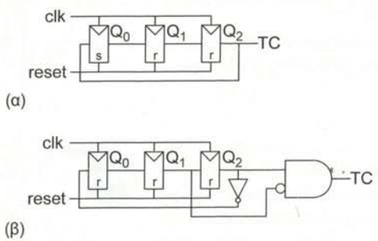
ΣΧΗΜΑ 11.52 Γρήγορος δυαδικός μετρητής επάνω/κάτω (το περισσότερο σημαντικό τμήμα).

έναν κύκλο, για τη φόρτωση της νέας τιμής *count* από τον καταχωρητή *shadow* και όχι από τον αθροιστή (ο οποίος ενδεχομένως να μην είχε επαρκή χρόνο στη διάθεσή του για να διαδώσει τα κρατούμενα).

11.5.3 Μετρητές Δακτυλίου και Johnson

Ένας *μετρητής δακτυλίου* (ring counter) αποτελείται από έναν καταχωρητή ολίσθησης των *M* bit, του οποίου η έξοδος τροφοδοτείται πίσω στην είσοδο, όπως παρουσιάζεται στο Σχήμα 11.53(α). Με το σήμα reset, το πρώτο bit αρχικοποιείται σε 1 και τα υπόλοιπα αρχικοποιούνται σε 0. Το σήμα *TC* αλλάζει κατάσταση κάθε *M* κύκλους. Οι μετρητές δακτυλίου αποτελούν έναν βολικό τρόπο για την κατασκευή εξαιρετικά γρήγορων μετρητών prescalar, επειδή δεν υπάρχει λογική μεταξύ των flip-flop, αλλά αυξάνουν το κόστος για μεγαλύτερες τιμές του *M*.

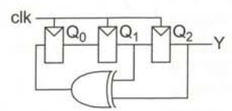
Ένας *μετρητής Johnson* ή *Mobius* είναι παρόμοιος μ' ένα μετρητή δακτυλίου, αλλά αντιστρέφει την έξοδο πριν αυτή τροφοδοτηθεί πίσω στην είσοδο, όπως παρουσιάζεται στο Σχήμα 11.53(β). Τα flip-flop μηδενίζονται και προσμετρούν 2*M* καταστάσεις πριν από την επανάληψη. Ο Πίνακας 11.6 παρουσιάζει την ακολουθία μέτρησης για έναν μετρητή Johnson των 3 bit.



ΣΧΗΜΑ 11.53 3-bit μετρητές δακτυλίου και Johnson.

ΠΙΝΑΚΑΣ 11.6 Ακολουθία μετρητή Johnson

Κύκλος	Q_0	Q_1	Q_2	TC
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	0	1	1	0
5	0	0	1	1
6	0	0	0	0
Επαναλαμβάνεται επ' άπειρον				



ΣΧΗΜΑ 11.54 3-bit καταχωρητής LFSR.

11.5.4 Καταχωρητές Ολίσθησης Γραμμικής Ανατροφοδότησης

Ένας καταχωρητής ολίσθησης γραμμικής ανατροφοδότησης (linear-feedback shift register, LFSR) αποτελείται από *N* καταχωρητές συνδεδεμένους μαζί, σαν ένας καταχωρητή ολίσθησης. Η είσοδος στον καταχωρητή ολίσθησης προέρχεται από την εφαρμογή μιας XOR σε συγκεκριμένα bit του καταχωρητή, όπως παρουσιάζεται στο Σχήμα 11.54 για έναν LFSR των 3 bit. Με το σήμα reset, οι καταχωρητές πρέπει να αρχικοποιηθούν σε μια μη-μηδενική τιμή (π.χ. όλοι σε 1). Τα μοτίβα εξόδων για τον LFSR παρουσιάζονται στον Πίνακα 11.7.

ΠΙΝΑΚΑΣ 11.7 Ακολουθία καταχωρητή LFSR

Κύκλος	Q_0	Q_1	Q_2 / Y
0	1	1	1
1	0	1	1
2	0	0	1
3	1	0	0
4	0	1	0
5	1	0	1
6	1	1	0
7	1	1	1
Επαναλαμβάνεται επ' άπειρον			

Ο συγκεκριμένος LFSR είναι ένα παράδειγμα *καταχωρητή ολίσθησης μέγιστου μήκους*, επειδή η έξοδός του διέρχεται διαδοχικά απ' όλους τους 2^n-1 συνδυασμούς (εκτός αυτού όπου όλες οι έξοδοι είναι 0). Οι είσοδοι που τροφοδοτούνται στην πύλη XOR ονομάζονται *ακολουθία λήψεων* (tap sequence) και συχνά καθορίζονται μ' ένα χαρακτηριστικό πολυώνυμο. Για παράδειγμα, ο συγκεκριμένος LFSR των 3 bit έχει το χαρακτηριστικό πολυώνυμο $1+x^2+x^3$, επειδή οι λήψεις (taps) βγαίνουν μετά το δεύτερο και τον τρίτο καταχωρητή.

Η έξοδος *Y* προκύπτει από την ακολουθία των 7 bit [1110010]. Αυτό είναι ένα παράδειγμα ψευδοτυχαίας ακολουθίας από bit (pseudo-random bit sequence, PRBS). Οι LFSR χρησιμοποιούνται ως μετρητές υψηλής ταχύτητας και γεννήτριες ψευδοτυχαίων αριθμών. Οι ψευδοτυχαίες ακολουθίες είναι βολικές για την υλοποίηση εγγενών λειτουργιών αυτο-ελέγχου και για τον έλεγχο του ρυθμού μετάδοσης σφαλμάτων σε τηλεπικοινωνιακές συνδέσεις. Χρησιμοποιούνται επίσης σε πολλά τηλεπικοινωνιακά συστήματα ευρέως φάσματος, όπως τα GPS και CDMA, όπου οι ιδιότητες συσχέτισης των σημάτων τους κάνουν τους άλλους χρήστες να παρουσιάζονται ως θόρυβος χωρίς συσχέτιση.

Ο Πίνακας 11.8 παραθέτει χαρακτηριστικά πολυώνυμα για ορισμένους ευρέως χρησιμοποιούμενους LFSR μέγιστου μήκους. Για ορισμένα μήκη *N*, είναι πιθανό να απαιτηθούν περισσότερες από δύο έξοδοι που επηρεάζουν τις εισόδους. Για πολλές τιμές του *N* υπάρχουν πολλαπλά πολυώνυμα που δίνουν ως αποτέλεσμα διαφορετικούς LFSR μέγιστου μήκους. Παρατηρήστε ότι η περίοδος του ρολογιού καθορίζεται από τον καταχωρητή κι ένα μικρό αριθμό καθυστερήσεων στην XOR. Ο [Golomb81] παρέχει μια ολοκληρωμένη ανάλυση των καταχωρητών ολίσθησης γραμμικής ανατροφοδότησης.

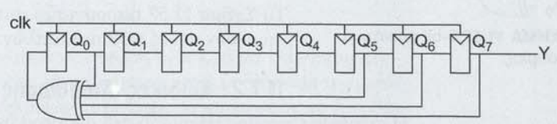
ΠΙΝΑΚΑΣ 11.8 Χαρακτηριστικά πολυώνυμα

N	Πολυώνυμο
3	$1 + x^2 + x^3$
4	$1 + x^3 + x^4$
5	$1 + x^3 + x^5$
6	$1 + x^5 + x^6$
7	$1 + x^6 + x^7$
8	$1 + x^1 + x^6 + x^7 + x^8$
9	$1 + x^5 + x^9$
15	$1 + x^{14} + x^{15}$
16	$1 + x^4 + x^{13} + x^{15} + x^{16}$
23	$1 + x^{18} + x^{23}$
24	$1 + x^{17} + x^{22} + x^{23} + x^{24}$
31	$1 + x^{28} + x^{31}$
32	$1 + x^{10} + x^{30} + x^{31} + x^{32}$

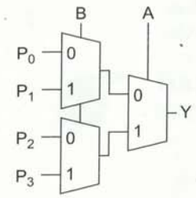
Παράδειγμα 11.1

Σχεδιάστε έναν καταχωρητή ολίσθησης γραμμικής ανατροφοδότησης, των 8 bit. Πόσο είναι το μήκος της ψευδοτυχαίας ακολουθίας bit που παράγει;

ΛΥΣΗ: Το Σχήμα 11.55 παρουσιάζει έναν LFSR των 8 bit που χρησιμοποιεί τις τέσσερις λήψεις μετά από το πρώτο, έκτο, έβδομο και όγδοο bit, όπως δίνεται στον Πίνακα 11.7. Το μήκος της ακολουθίας που παράγει (πριν αρχίσει να την επαναλαμβάνει) είναι 2^8-1 = 255 bits.



ΣΧΗΜΑ 11.55 8-bit καταχωρητής LFSR.



ΣΧΗΜΑ 11.56 Boolean λογική μονάδα.

11.6 Λειτουργίες Λογικής Boole

Οι λειτουργίες λογικής Boole πραγματοποιούνται εύκολα με τη χρήση βασιζόμενων σε πολυπλέκτες κυκλωμάτων, όπως παρουσιάζεται στο Σχήμα 11.56. Ο Πίνακας 11.9 δείχνει πώς ανατίθενται οι τιμές στις εισόδους για την υλοποίηση διαφορετικών λογικών λειτουργιών. Αναθέτοντας διαφορετικές τιμές στις εισόδους *P*, η μονάδα μπορεί να υλοποιεί άλλες πράξεις, όπως οι XNOR(*A*, *B*) και NOT(*A*). Μια *Αριθμητική Λογική Μονάδα* (Arithmetic Logic Unit, ALU) χρειάζεται και αριθμητικές λειτουργίες (πρόσθεση, αφαίρεση) αλλά και λειτουργίες λογικής Boole.

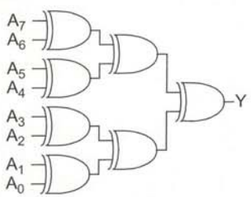
ΠΙΝΑΚΑΣ 11.9 Οι λειτουργίες (συναρτήσεις) που υλοποιεί μια μονάδα Boolean λογικής

Λειτουργία	P_0	P_1	P_2	P_3
AND(*A*, *B*)	0	0	0	1
OR(*A*, *B*)	0	1	1	1
XOR(*A*, *B*)	0	1	1	0
NAND(*A*, *B*)	1	1	1	0
NOR(*A*, *B*)	1	0	0	0

11.7 Κωδικοποίηση

Οι κώδικες ανίχνευσης και διόρθωσης σφαλμάτων χρησιμοποιούνται για την αύξηση της αξιοπιστίας ενός συστήματος. Οι διατάξεις μνημών είναι ιδιαίτερα ευάλωτες σε τυχαία/παροδικά σφάλματα (soft errors), τα οποία προκαλούνται όταν σωματίδια *a* ή κοσμική ακτινοβολία αλλάζουν την κατάσταση ενός bit. Τέτοια σφάλματα μπορούν να ανιχνευθούν, ή ακόμα και να διορθωθούν, προσθέτοντας λίγα επιπλέον bit ελέγχου σε κάθε λέξη της διάταξης. Οι κώδικες χρησιμοποιούνται επίσης για τη μείωση του ρυθμού σφαλμάτων μετάδοσης σε τηλεπικοινωνιακές συνδέσεις.

Η απλούστερη μορφή κώδικα ανίχνευσης σφαλμάτων είναι η *ισοτιμία*, η οποία ανιχνεύει σφάλματα σε επίπεδο μεμονωμένων bit. Πολυπλοκότεροι και πιο προηγμένοι κώδικες διόρθωσης σφαλμάτων (error-correcting codes, ECC) έχουν δυνατότητες μονής διόρθωσης σφαλμάτων και διπλής ανίχνευσης σφαλμάτων (single-error correcting/double-error detecting, SEC-DEC). Οι κώδικες Gray είναι μια άλλη χρήσιμη εναλλακτική επιλογή έναντι των συνηθισμένων δυαδικών κωδίκων. Δεδομένου ότι όλοι οι κώδικες βασίζονται σε μεγάλο βαθμό στη λειτουργία της αποκλειστικής διάζευξης, XOR, θα εξετάσουμε μια γκάμα σχεδιάσεων CMOS με πύλες XOR.



ΣΧΗΜΑ 11.57 8-bit γεννήτρια ισοτιμίας.

11.7.1 Ισοτιμία

Ένα bit ισοτιμίας (parity bit) μπορεί να προστεθεί σε μια λέξη των *N* bit για να υποδείξει εάν ο αριθμός των 1 στη λέξη είναι άρτιος ή περιττός. Στην *άρτια ισοτιμία*, το επιπλέον bit είναι το αποτέλεσμα της XOR επί των υπόλοιπων *N* bit, το οποίο διασφαλίζει ότι η (*N*+1)-bit κωδικοποιημένη λέξη έχει άρτιο αριθμό ψηφίων 1:

$$A_n = \text{PARITY} = A_0 \oplus A_1 \oplus A_2 \oplus \ldots \oplus A_{n-1} \qquad (11.28)$$

Το Σχήμα 11.57 παρουσιάζει μια συμβατική υλοποίηση. Μπορούν επίσης να χρησιμοποιηθούν πύλες XOR πολλαπλών εισόδων.

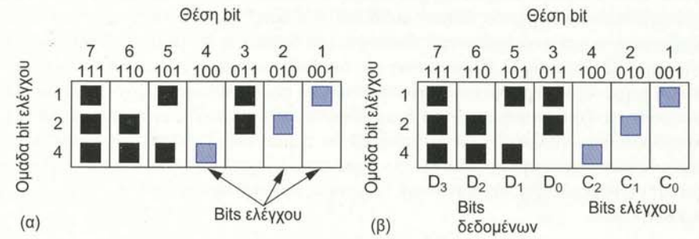
11.7.2 Κώδικες Διόρθωσης Σφαλμάτων

Η *απόσταση Hamming* [Hamming50] ανάμεσα σε δύο δυαδικούς αριθμούς είναι ο αριθμός των bit κατά τα οποία διαφέρουν μεταξύ τους οι δύο αριθμοί. Ένα σφάλμα σ' ένα μεμονωμένο bit μετατρέπει μια λέξη δεδομένων σε μια διαφορετική λέξη, η οποία απέχει από την αρχική λέξη απόσταση Hamming ίση με 1. Οι κώδικες διόρθωσης σφαλμάτων προσθέτουν bits ελέγχου στις λέξεις δεδομένων, έτσι ώστε η ελάχιστη απόσταση Hamming μεταξύ έγκυρων λέξεων να αυξάνεται. Η ισοτιμία είναι ένα παράδειγμα κώδικα μ'

ένα μεμονωμένο bit ελέγχου και απόσταση Hamming μεταξύ έγκυρων λέξεων ίση με 2, έτσι ώστε τα σφάλματα σε επίπεδο μεμονωμένων bit να οδηγούν σε μη αποδεκτές λέξεις και κατά συνέπεια να είναι ανιχνεύσιμα. Εάν προστεθούν περισσότερα bit ελέγχου, έτσι ώστε η ελάχιστη απόσταση μεταξύ έγκυρων λέξεων να είναι 3, ένα σφάλμα σε επίπεδο μεμονωμένου bit μπορεί να διορθωθεί επειδή θα υπάρχει μια έγκυρη λέξη σε απόσταση 1. Εάν η ελάχιστη απόσταση μεταξύ έγκυρων λέξεων είναι 4, ένα σφάλμα σε επίπεδο μεμονωμένου bit μπορεί να διορθωθεί, ενώ ένα σφάλμα το οποίο αλλοιώνει δύο bit μπορεί να ανιχνευθεί (αλλά όχι να διορθωθεί). Εάν η πιθανότητα ύπαρξης λανθασμένων bit είναι χαμηλή και χωρίς συσχέτιση μεταξύ των bit, τέτοιου είδους κώδικες διόρθωσης 1/ανίχνευσης 2 (SEC-DEC) μειώνουν σε μεγάλο βαθμό το συνολικό ποσοστό σφαλμάτων του συστήματος. Μεγαλύτερες αποστάσεις Hamming βελτιώνουν ακόμη περισσότερο το ποσοστό σφαλμάτων, με κόστος τη χρήση περισσότερων bit ελέγχου.

Γενικά, μπορείτε να κατασκευάσετε ένα κώδικα Hamming απόστασης 3 και μήκους 2^c-1 με *c* bit ελέγχου και $N = 2^c$-*c*-1 bit δεδομένων χρησιμοποιώντας μια απλή διαδικασία [Wakerly00]. Εάν τα bit αριθμούνται από το 1 ως το 2^c-1, κάθε bit σε μια θέση που είναι δύναμη του 2 χρησιμεύει ως bit ελέγχου. Η τιμή του bit ελέγχου επιλέγεται ώστε να έχουμε άρτια ισοτιμία για όλα τα bit που έχουν 1 στην ίδια θέση μ' αυτή του ψηφίου ελέγχου, όπως απεικονίζεται στο Σχήμα 11.58(α) για ένα κώδικα των 7 bit, με 4 bit δεδομένων και 3 bit ελέγχου. Κατά παράδοση, τα bit αναδιοργανώνονται ώστε να είναι συνεχή (γειτονικά) τα bit δεδομένων και ελέγχου, όπως παρουσιάζεται στο Σχήμα 11.58(β). Η δομή αυτή ονομάζεται *πίνακας ελέγχου ισοτιμίας* και κάθε bit ελέγχου μπορεί να υπολογιστεί ως αποτέλεσμα της XOR επί των επισημαινόμενων bit δεδομένων.

$$C_0 = D_3 \oplus D_1 \oplus D_0$$
$$C_1 = D_3 \oplus D_2 \oplus D_0 \qquad (11.29)$$
$$C_2 = D_3 \oplus D_2 \oplus D_1$$



ΣΧΗΜΑ 11.58 Πίνακας ελέγχου ισοτιμίας.

Ο αποκωδικοποιητής του μηχανισμού διόρθωσης σφαλμάτων εξετάζει τα bit ελέγχου. Εάν έχουν όλα άρτια ισοτιμία, η λέξη θεωρείται σωστή. Εάν μια ή περισσότερες ομάδες ψηφίων έχουν περιττή ισοτιμία, έχει προκύψει κάποιο σφάλμα. Το μοτίβο των bit ελέγχου που έχουν λανθασμένη ισοτιμία αποκαλείται *σύνδρομο* (syndrome) και αντιστοιχεί στη θέση του λανθασμένου bit. Ο αποκωδικοποιητής πρέπει να αναστρέψει αυτό το bit για να επαναφέρει το σωστό αποτέλεσμα.

Παράδειγμα 11.2

Υποθέστε ότι η τιμή δεδομένων 1001 πρόκειται να μεταδοθεί με χρήση ενός κώδικα Hamming απόστασης 3. Ποιά είναι τα bit ελέγχου; Εάν τα bit δεδομένων αλλοιώνονταν κατά τη διάρκεια της μετάδοσης, καταλήγοντας σε 1101, εξηγήστε ποιο θα ήταν το *σύνδρομο* και πώς θα μπορούσαν να διορθωθούν τα δεδομένα.

Λύση: Σύμφωνα με τις εξισώσεις (11.29), τα bit ελέγχου θα ήταν 100, σε αντιστοιχία με τη μεταδιδόμενη λέξη, 1001100. Η ληφθείσα λέξη είναι 1101100. Το *σύνδρομο* είναι 110 – δηλαδή, περιττή ισοτιμία στα bit ελέγχου C_2 και C_1, η οποία αποκαλύπτει ένα σφάλμα στη θέση bit 110 = 6. Το bit αυτής της θέσης ανα-

στρέφεται για την παραγωγή της διορθωμένης λέξης, 1001100, και τα ψηφία ελέγχου απομακρύνονται ώστε να μείνει τελικά η σωστή τιμή δεδομένων, 1001.

Ένας SEC-DED κώδικας Hamming απόστασης 4 μπορεί να κατασκευαστεί από έναν κώδικα απόστασης 3, με την προσθήκη ενός επιπλέον bit ισοτιμίας για ολόκληρη τη λέξη. Εάν υπάρξει σφάλμα σε επίπεδο μεμονωμένου bit, η ισοτιμία θα αποτύχει και τα bit ελέγχου θα υποδείξουν πώς πρέπει να διορθωθούν τα δεδομένα. Εάν υπάρξει σφάλμα σε 2 bits, τα bit ελέγχου θα υποδείξουν μεν σφάλμα, αλλά η ισοτιμία θα περάσει – δηλαδή, το σφάλμα είναι ανιχνεύσιμο αλλά μη-διορθώσιμο.

Ο πίνακας ελέγχου ισοτιμίας καθορίζει τον αριθμό των XOR που απαιτούνται για την υλοποίηση της λογικής της κωδικοποίησης και αποκωδικοποίησης. Ένας SEC-DED κώδικας Hamming για λέξεις δεδομένων των 64 bit έχει 8 bits ελέγχου και απαιτεί 296 πύλες XOR. Η λογική ελέγχου ισοτιμίας για ολόκληρη τη λέξη έχει 72 εισόδους. Ο κώδικας SEC-DED του Hsiao [Hsiao70] επιτυγχάνει την ίδια λειτουργία με τον ίδιο αριθμό bits δεδομένων και ελέγχου, αλλά είναι έξυπνα σχεδιασμένος ώστε να ελαχιστοποιεί το κόστος, χρησιμοποιώντας μόνο 216 πύλες XOR και λογική ελέγχου ισοτιμίας με 27 το πολύ εισόδους. Ο [Hsiao70] παρουσιάζει πίνακες ελέγχου ισοτιμίας για λέξεις δεδομένων των 16, 32 και 64 bit, με 6, 7 και 8 bit ελέγχου.

Καθώς αυξάνεται το μήκος των δεδομένων και η επιτρεπτή πολυπλοκότητα των αποκωδικοποιητών, αναδεικνύονται ως πιο αποτελεσματικοί άλλοι κώδικες. Σ' αυτούς περιλαμβάνονται οι Reed-Solomon, BCH και Turbo. Στα [Lin83, Sweeney02, Sklar01, Fujiwara06] αλλά και σε πολλά άλλα κείμενα παρέχονται εκτενείς πληροφορίες για μεγάλη ποικιλία κωδίκων διόρθωσης σφαλμάτων.

11.7.3 Κώδικες Gray

Οι κώδικες Gray, οι οποίοι οφείλουν το όνομά τους στον Frank Gray που κατοχύρωσε με ευρεσιτεχνία τη χρήση τους σε περιστροφικούς κωδικοποιητές [Gray53], έχουν την εξής χρήσιμη ιδιότητα: διαδοχικοί αριθμοί διαφέρουν κατά μία μόνο θέση bit. Αν και υπάρχουν πολλοί πιθανοί κώδικες Gray, ένας από τους απλούστερους είναι ο *δυαδικά κατοπτρικός* (binary-reflected) κώδικας Gray, ο οποίος παράγεται ξεκινώντας με όλα τα bit μηδενισμένα και αναστρέφοντας διαδοχικά το δεξιότερο bit με αποτέλεσμα να παράγεται μια νέα συμβολοσειρά. Ο Πίνακας 11.10 συγκρίνει ένα δυαδικό κώδικα κι ένα *δυαδικά-κατοπτρικό* κώδικα Gray, των 3 bit. Οι μηχανές πεπερασμένων καταστάσεων, οι οποίες τυπικά διέρχονται από διαδοχικές καταστάσεις, μπορούν να εξοικονομήσουν ενέργεια κωδικοποιώντας κατά Gray τις καταστάσεις, με στόχο τη μείωση του αριθμού των εναλλαγών κατάστασης. Όταν η τιμή ενός μετρητή πρέπει να συγχρονιστεί με ρολόγια διαφορετικών συχνοτήτων, μπορεί να κωδικοποιηθεί κατά Gray, έτσι ώστε το κύκλωμα συγχρονισμού να είναι σίγουρο ότι θα λάβει είτε την τρέχουσα, είτε την προηγούμενη τιμή, επειδή αλλάζει μόνο ένα bit σε κάθε κύκλο.

ΠΙΝΑΚΑΣ 11.10 Κώδικας Gray των 3 bit

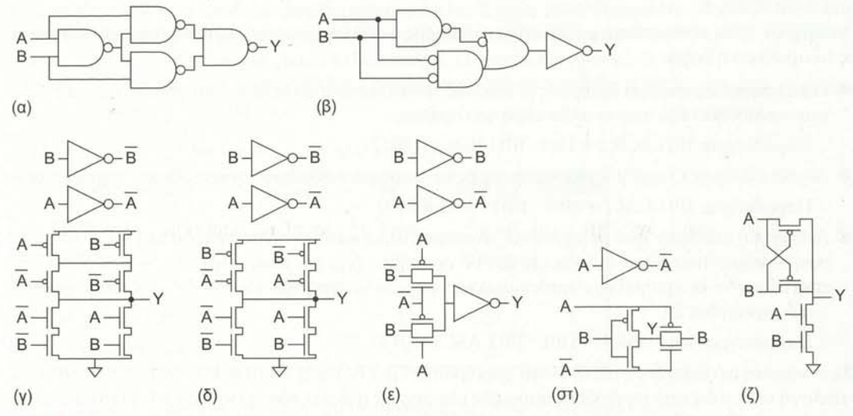
Αριθμός	Δυαδικό	Κώδικας Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Η μετατροπή αναπαραστάσεων από δυαδικό N-bit κώδικα B σε *δυαδικά-κατοπτρικό* Gray κώδικα G είναι εξαιρετικά απλή.

$$\begin{array}{ll} \text{Δυαδικό} \rightarrow \text{Gray} & \text{Gray} \rightarrow \text{Δυαδικό} \\ G_{N-1} = B_{N-1} & B_{N-1} = G_{N-1} \\ G_i = B_{i+1} \oplus B_i & B_i = B_{i+1} \oplus G_i \quad N-1 > i \geq 0 \end{array} \tag{11.30}$$

11.7.4 Κυκλωματικές Μορφές XOR/XNOR

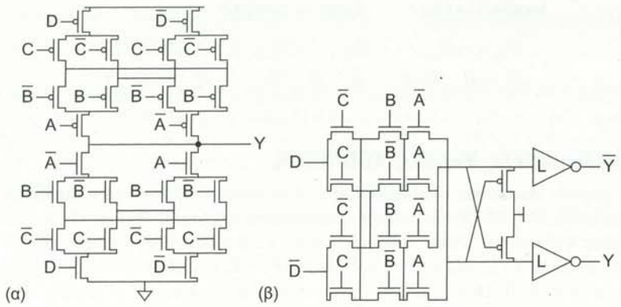
Μία από τις χρόνιες δυσκολίες στη σχεδίαση κυκλωμάτων CMOS είναι η κατασκευή μιας γρήγορης, συμπαγούς πύλης XOR ή XNOR με χαμηλή κατανάλωση ισχύος. Το Σχήμα 11.59 παρουσιάζει ορισμένες κοινές σχεδιάσεις για στατικές, μιας γραμμής (single rail) πύλες XOR δύο εισόδων· οι σχεδιάσεις πυλών XNOR είναι παρόμοιες. Τα Σχήματα 11. 59(α) και 11.59(β) παρουσιάζουν υλοποιήσεις σε επίπεδο πυλών. Η πρώτη είναι έξυπνη, αλλά η δεύτερη είναι λίγο πιο αποτελεσματική. Το Σχήμα 11.59(γ) παρουσιάζει μια συμπληρωματική πύλη CMOS. Το Σχήμα 11.59(δ) βελτιώνει ελαφρώς την πύλη, βελτιστοποιώντας δύο επαφές· αυτή είναι μια ευρέως χρησιμοποιούμενη σχεδίαση τυποποιημένου κυττάρου. Το Σχήμα 11.59(ε) παρουσιάζει μια σχεδίαση πύλης μετάδοσης. Το Σχήμα 11.59(στ) παρουσιάζει μια σχεδίαση «ελεγχόμενου αντιστροφέα» 6 εισόδων. Όταν η είσοδος A είναι «0», η πύλη μετάδοσης ενεργοποιείται και η είσοδος B περνιέται στην έξοδο. Όταν η A είναι «1», τροφοδοτεί ένα ζεύγος τρανζίστορ που αναστρέφουν την B. Η σχεδίαση αυτή είναι συμπαγής, αλλά μη-αποκαταστάσιμου τύπου. Ορισμένοι προσομοιωτές επιπέδου διακοπτών, όπως ο IRSIM, δε μπορούν να χειριστούν αυτή την αντισυμβατική σχεδίαση. Το Σχήμα 11.59(ζ) [Wang94] παρουσιάζει μια γρήγορη και συμπαγή πύλη διέλευσης με 4 τρανζίστορ, η οποία δεν καλύπτει όλο το εύρος μεταβολής του σήματος (rail to rail swing).



ΣΧΗΜΑ 11.59 Σχεδιάσεις στατικών πυλών XOR 2 εισόδων.

Οι πύλες XOR 3 ή 4 εισόδων μπορούν να είναι πιο συμπαγείς, αν και όχι κατ' ανάγκην πιο γρήγορες από μια σειρά διαδοχικά συνδεδεμένων πυλών 2 εισόδων. Το Σχήμα 11.60(α) παρουσιάζει μια στατική CMOS πύλη XOR 4 εισόδων [Griffin83] και το Σχήμα 11.60(β) μια CPL πύλη XOR/XNOR 4 εισόδων, ενώ στο Σχήμα 9.20(γ) παρουσιάσαμε μια CVSL πύλη XOR/XNOR 4 εισόδων. Παρατηρήστε ότι τα δένδρα των true εισόδων και των συμπληρωματικών τους μοιράζονται τα περισσότερα τρανζίστορ. Όπως αναφέραμε στο Κεφάλαιο 6, η λογική CPL δεν συμπεριφέρεται καλά στις χαμηλές τάσεις.

Οι δυναμικές πύλες XOR θέτουν ένα πρόβλημα, επειδή απαιτούνται τόσο οι true όσο και οι συμπληρωματικές είσοδοι, πράγμα το οποίο παραβιάζει τον κανόνα της μονοτονικότητας. Οι κοινές λύσεις που αναφέρθηκαν στην Ενότητα 11.2.2.11 είναι είτε η τοποθέτηση της πύλης XOR στο τέλος μιας αλυσίδας λογικής domino και η υλοποίησή της με στατική λογική CMOS, είτε η κατασκευή μιας δομής domino διπλής γραμμής. Μια υλοποίηση πύλης XOR 2 εισόδων με λογική domino διπλής γραμμής παρουσιάζεται στο Σχήμα 9.30(γ).



ΣΧΗΜΑ 11.60 Σχεδιάσεις πυλών XOR 4 εισόδων

11.8 Ολισθητές

Οι λειτουργίες ολίσθησης (shift) μπορούν να εκτελούνται είτε κατά σταθερό είτε κατά μεταβλητό ποσό. Οι σταθερές ολισθήσεις είναι απλές όσον αφορά τις απαιτήσεις τους σε hardware – χρειάζονται μόνο αγωγούς διασύνδεσης. Αποτελούν επίσης έναν αποτελεσματικό τρόπο εκτέλεσης πολλαπλασιασμών ή διαιρέσεων κατά δυνάμεις του δύο. Ένας μεταβλητός ολισθητής δέχεται μια N-bit είσοδο A, ένα ποσό ολίσθησης k και σήματα ελέγχου τα οποία υποδεικνύουν τον τύπο και την κατεύθυνση της ολίσθησης. Παράγει μια N-bit έξοδο, Y.

Υπάρχουν τρεις κοινοί τύποι μεταβλητής ολίσθησης, κάθε ένας εκ των οποίων μπορεί να εκτελείται προς τα αριστερά ή δεξιά:

- *Περιστροφή*: Περιστρέφει αριθμούς με κυκλικό τρόπο, ώστε οι κενές θέσεις να γεμίζουν με τα bit που έχουν ολισθήσει έξω από το άλλο άκρο του αριθμού.
  - Παράδειγμα: 1011 ROR 1 = 1101· 1011 ROL 1 = 0111
- *Λογική ολίσθηση*: Ο αριθμός ολισθαίνει προς τα αριστερά ή δεξιά και οι κενές θέσεις γεμίζουν με 0.
  - Παράδειγμα: 1011 LSR 1 = 0101· 1011 LSL 1 = 0110
- *Αριθμητική ολίσθηση*: Ίδια με τη λογική ολίσθηση, αλλά κατά την ολίσθηση δεξιά γεμίζει τα περισσότερο σημαντικά bit με αντίγραφα του bit προσήμου (για την εφαρμογή του σωστού προσήμου, επεκτείνονται οι αριθμοί σε συμπλήρωμα ως προς 2 όταν χρησιμοποιείται δεξιά ολίσθηση κατά k, για διαίρεση διά 2^k).
  - Παράδειγμα: 1011 ASR 1 = 1101· 1011 ASL 1 = 0110

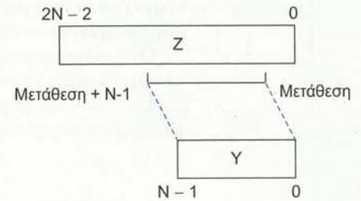
Σε εννοιολογικό επίπεδο, η περιστροφή χρησιμοποιεί μια διάταξη N πολυπλεκτών των N εισόδων για την επιλογή κάθε μιας από τις εξόδους από κάθε μία από τις πιθανές θέσεις εισόδου. Μια τέτοια διάταξη αποκαλείται ολισθητής διάταξης (*array shifter*). Ο ολισθητής διάταξης απαιτεί έναν αποκωδικοποιητή για να παράγει το ποσό ολίσθησης (1 από N, τύπου «one hot») για κάθε μία από τις εισόδους. Στην πράξη, οι πολυπλέκτες με περισσότερες από 4 έως 8 εισόδους έχουν υπερβολική παρασιτική χωρητικότητα· για το λόγο αυτό, είναι ταχύτερη η κατασκευή ολισθητών από $\log_v N$ το πλήθος επίπεδα πολυπλεκτών των v εισόδων. Αυτοί αποκαλούνται *λογαριθμικοί ολισθητές*. Για παράδειγμα, σ' ένα λογαριθμικό ολισθητή 2ης τάξης, το πρώτο επίπεδο εκτελεί ολίσθηση κατά $N/2$, το δεύτερο κατά $N/4$, και ούτω καθεξής, έως το τελικό επίπεδο που εκτελεί ολίσθηση κατά 1. Σ' ένα λογαριθμικό ολισθητή, δεν απαιτείται αποκωδικοποιητής. Ο υλοποιημένος με CMOS πύλες μετάδοσης πολυπλέκτης του Σχήματος 9.47 είναι ιδιαίτερα κατάλληλος για χρήση σε λογαριθμικούς ολισθητές, επειδή η μεγάλη χωρητικότητα αγωγών οδηγείται απευθείας από έναν αντιστροφέα και όχι από ένα ζεύγος εν σειρά τρανζίστορ. Οι υλοποιημένοι με πύλες μετάδοσης πολυπλέκτες 4:1 ή 8:1 μειώνουν τον αριθμό των επιπέδων κατά συντελεστή 2 ή 3, εις βάρος της διασύνδεσης και του fanout. Ζεύγη ή τριάδες του ποσού ολίσθησης αποκωδικοποιούνται και οδηγούνσήματα επιλογής τύπου «one hot» του πολυπλέκτη σε κάθε επίπεδο. Ο [Tharakan92] περιγράφει έναν λογαριθμικό ολισθητή υλοποίησης domino, ο οποίος χρησιμοποιεί πολυπλέκτες 3:1 για να μειώσει τον αριθμό των λογικών σταθμών.

Μια περιστροφή αριστερά κατά k bits είναι ισοδύναμη με μια περιστροφή δεξιά κατά $N - k$ bits. Ο υπολογισμός $N - k$ απαιτεί έναν αφαιρέτη στο κρίσιμο μονοπάτι. Αξιοποιώντας την αριθμητική συμπληρώματος ως προς 2 και το γεγονός ότι η περιστροφή είναι κυκλική ως προς το υπόλοιπο (modulo) N, έχουμε $N - k = N + \bar{k} + 1 = \bar{k} + 1$. Συνεπώς, η περιστροφή αριστερά μπορεί να εκτελείται με προ-ολίσθηση δεξιά κατά 1 και κατόπιν εκτέλεση μιας περιστροφής δεξιά κατά το συμπλήρωμα του ποσού ολίσθησης.

Οι λογικές και αριθμητικές ολισθήσεις είναι παρόμοιες με τις περιστροφές, αλλά πρέπει να αντικαθιστούν bits στο ένα ή στο άλλο άκρο με μια *τιμή θανάτωσης* (kill value, είτε 0 είτε το bit προσήμου). Οι δύο σημαντικότερες αρχιτεκτονικές ολισθητών είναι οι ολισθητές χοάνης και οι περιστροφικοί ολισθητές. Σ' έναν *ολισθητή χοάνης* (funnel shifter), οι τιμές θανάτωσης ενσωματώνονται στην αρχή, ενώ σ' έναν *περιστροφικό ολισθητή* οι τιμές θανάτωσης επιλέγονται στο τέλος. Κάθε μία από αυτές τις αρχιτεκτονικές περιγράφεται στη συνέχεια. Τόσο οι περιστροφικοί όσο και οι ολισθητές χοάνης μπορούν να κατασκευάζονται σε υλοποιήσεις διάταξης ή λογαριθμικές. Ο [Huntzicker08] εξετάζει τους συμβιβασμούς ενέργειας-καθυστέρησης που ισχύουν για τους ολισθητές. Για γενικού σκοπού λειτουργίες ολίσθησης, αμφότερες οι αρχιτεκτονικές είναι συγκρίσιμες όσον αφορά την ενέργεια και την καθυστέρηση. Με δεδομένες τυπικές παρασιτικές χωρητικότητες, η μέθοδος του Λογικού Φόρτου δείχνει ότι η λογαριθμική δομή που χρησιμοποιεί πολυπλέκτες 4:1 είναι η πλέον αποτελεσματική. Εάν απαιτούνται μόνο λειτουργίες ολίσθησης (όχι περιστροφές), η αρχιτεκτονική χοάνης είναι απλούστερη, ενώ εάν απαιτούνται μόνο περιστροφές (όχι ολισθήσεις), είναι απλούστερη η περιστροφική αρχιτεκτονική.

11.8.1 Ολισθητής Χοάνης

Ο *ολισθητής χοάνης* (funnel shifter) δημιουργεί μια λέξη εισόδου Z των $2N - 1$ bit από την είσοδο A και/ή τις τιμές θανάτωσης και κατόπιν επιλέγει ένα πεδίο μεγέθους N bit από αυτή τη λέξη εισόδου, όπως παρουσιάζεται στο Σχήμα 11.61. Παίρνει το όνομά του από τον τρόπο με τον οποίο η λέξη μεταβάλλεται σε μια άλλη μικρότερου εύρους, σαν να περνάει από μια χοάνη. Ο Πίνακας 11.11 δείχνει πώς σχηματίζεται η λέξη Z για κάθε τύπο ολίσθησης. Η λέξη Z ενσωματώνει την προ-ολίσθηση κατά 1 bit για τις αριστερές ολισθήσεις.

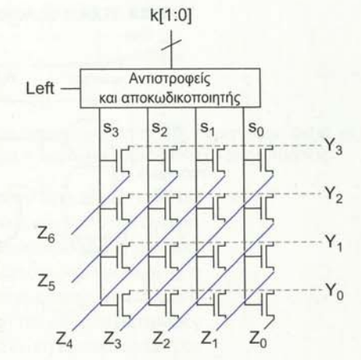


ΣΧΗΜΑ 11.61 Λειτουργία του ολισθητή χοάνης

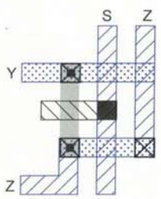
ΠΙΝΑΚΑΣ 11.11 Γεννήτρια λέξεων για τον ολισθητή χοάνης

Τύπος ολίσθησης	$Z_{2N-2:N}$	Z_{N-1}	$Z_{N-2:0}$	Μετάθεση
Περιστροφή, δεξιά	$A_{N-2:0}$	A_{N-1}	$A_{N-2:0}$	k
Λογική, δεξιά	0	A_{N-1}	$A_{N-2:0}$	k
Αριθμητική, δεξιά	πρόσημο	A_{N-1}	$A_{N-2:0}$	k
Περιστροφή, αριστερά	$A_{N-1:1}$	A_0	$A_{N-1:1}$	$\bar{k}$
Λογική/Αριθμητική, αριστερά	$A_{N-1:1}$	A_0	0	$\bar{k}$

Η απλούστερη δυνατή σχεδίαση ολισθητή χοάνης αποτελείται από μια διάταξη N πολυπλεκτών N-εισόδων έκαστος, οι οποίοι δέχονται 1 από τα N σήματα επιλογής (ένας πολυπλέκτης για κάθε bit εξόδου). Μια τέτοια διάταξη παρουσιάζεται στο Σχήμα 11.62, όπου χρησιμοποιούνται nMOS τρανζίστορ περάσματος για έναν ολισθητή των 4 bit. Το ποσό ολίσθησης αντιστρέφεται υπό συνθήκη και αποκωδικοποιείται σε σήματα επιλογής τα οποία τροφοδοτούνται κατακόρυφα στη διάταξη. Οι έξοδοι λαμβάνονται οριζόντια. Κάθε γραμμή από τρανζίστορ συνδεδεμένα σε μία έξοδο σχηματίζει έναν από τους πολυπλέκτες. Οι $2N - 1$ είσοδοι κατευθύνονται διαγώνια, στις κατάλληλες εισόδους των πολυπλεκτών . Το Σχήμα 11.63 παρουσιάζει το συμβολικό διάγραμμα για ένα από τα N^2 τρανζίστορ της διάταξης. Τα nMOS τρανζίστορ περάσματος αντιμετωπίζουν το πρόβλημα της μιας πτώσης τάσης κατωφλίου, αλλά αυτό μπορεί να λυθεί είτε προ-φορτίζοντας τις εξόδους (πράγμα το οποίο έχει γίνει στον Alpha 21164 [Gronowski96]) είτε χρησιμοποιώντας πλήρεις CMOS πύλες μετάδοσης.



ΣΧΗΜΑ 11.62 Διάταξη ολισθητή χοάνης 4-bit.



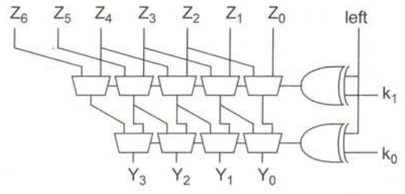
ΣΧΗΜΑ 11.63 Διάγραμμα φυσικής σχεδίασης ολισθητή χοάνης.

Η αρχιτεκτονική ολισθητή διάταξης λειτουργεί καλά για μικρούς ολισθητές σε σχεδιάσεις επιπέδου τρανζίστορ, αλλά επιδεικνύει υψηλή παρασιτική χωρητικότητα σε μεγαλύτερους ολισθητές, γεγονός το οποίο οδηγεί σε υπερβολική καθυστέρηση και ενέργεια. Επιπλέον, οι ολισθητές διάταξης δεν είναι κατάλληλοι για χρήση σε σχεδιάσεις τυποποιημένων κυττάρων. Το Σχήμα 11.64 παρουσιάζει έναν 4-bit λογαριθμικό ολισθητή βασιζόμενο σε πολλαπλά επίπεδα πολυπλεκτών 2:1 (οι οποίοι μπορούν, φυσικά, να είναι πύλες μετάδοσης) [Lim72]. Οι πύλες XOR στις εισόδους ελέγχου αντιστρέφουν υπό συνθήκη το ποσό ολίσθησης για αριστερές ολισθήσεις.

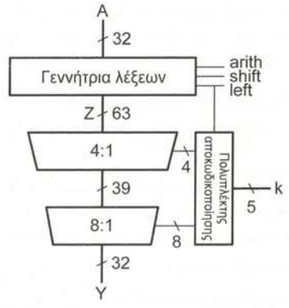
Το Σχήμα 11.65 παρουσιάζει έναν 32-bit ολισθητή χοάνης που χρησιμοποιεί έναν πολυπλέκτη 4:1 ακολουθούμενο από έναν πολυπλέκτη 8:1 [Huntzicker08]. Η γεννήτρια πηγών (λέξεων) επιλέγει την 63-bit λέξη Z. Το πρώτο στάδιο εκτελεί μια «χονδρική» ολίσθηση δεξιά κατά 0, 8, 16 ή 24 bit. Το δεύτερο στάδιο εκτελεί μια υψηλότερης ανάλυσης ολίσθηση δεξιά κατά 0 έως 7 bits. Η βαθμίδα αποκωδικοποίησης του πολυπλέκτη αντιστρέφει υπό συνθήκη το *k* για ολισθήσεις αριστερά, υπολογίζει τα σήματα επιλογής (select) και τα απομονώνει για την οδήγηση των πολυπλεκτών μεγάλου εύρους.

Σε εννοιολογικό επίπεδο, η γεννήτρια πηγών (λέξεων) είναι ένας πολυπλέκτης 5:1 των $2N$–1 bit, ο οποίος ελέγχεται από τον τύπο και την κατεύθυνση της ολίσθησης. Το Σχήμα 11.66 δείχνει πώς μπορεί να απλοποιηθεί η λογική της γεννήτριας πηγών (λέξεων). Οι οριζόντιες γραμμές ελέγχου πρέπει να απομονωθούν ώστε να μπορούν να οδηγήσουν το μεγάλο αριθμό πυλών και βρίσκονται στο κρίσιμο μονοπάτι. Ακόμα κι αν είναι διαθέσιμες νωρίς, το bit προσήμου συνεχίζει να είναι κρίσιμης σημασίας. Εάν υποστηρίζονται μόνο συγκεκριμένοι τύποι ολισθήσεων, η λογική μπορεί να βελτιωθεί ακόμα περισσότερο.

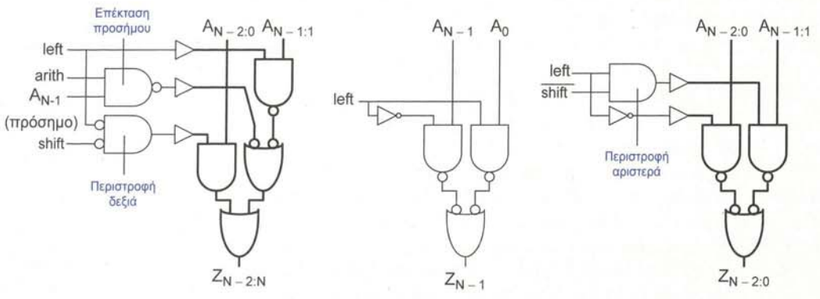
Ο ολισθητή χοάνης παρουσιάζει ένα πρόβλημα χωροθέτησης επειδή η γεννήτρια πηγών (λέξεων) και τα αρχικά στάδια των πολυπλεκτών έχουν μεγαλύτερο εύρος από το υπόλοιπο μονοπάτι δεδομένων. Το Σχήμα 11.67 παρουσιάζει μια χωροθέτηση στην οποία η γεννήτρια πηγών (λέξεων) «διπλώνεται» για να ταιριάξει με το μονοπάτι δεδομένων. Αυτό το «δίπλωμα» μειώνει επίσης τα μήκη των αγωγών, πράγμα το



ΣΧΗΜΑ 11.64 4-bit λογαριθμικός ολισθητής χοάνης.

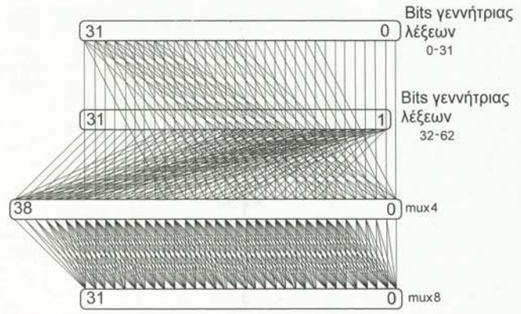


ΣΧΗΜΑ 11.65 32-bit λογαριθμικός ολισθητής χοάνης.



ΣΧΗΜΑ 11.66 Βελτιστοποιημένη λογική για τη γεννήτρια πηγών (λέξεων).

οποίο εξοικονομεί ενέργεια. Ανάλογα με τους περιορισμούς χωροθέτησης, τα επιπλέον επτά περισσότερο σημαντικά bit του πολυπλέκτη πρώτου επιπέδου μπορούν να «διπλωθούν» σε μια επιπλέον γραμμή ή να ενσωματωθούν στη μονάδα zipper.



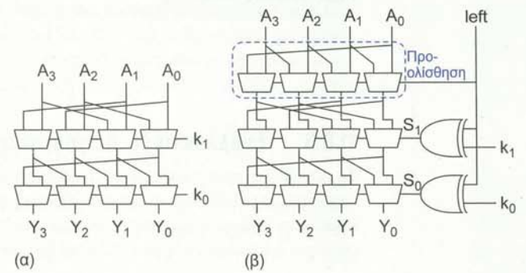
ΣΧΗΜΑ 11.67 Χωροθετήσεις για τον ολισθητή χοάνης.

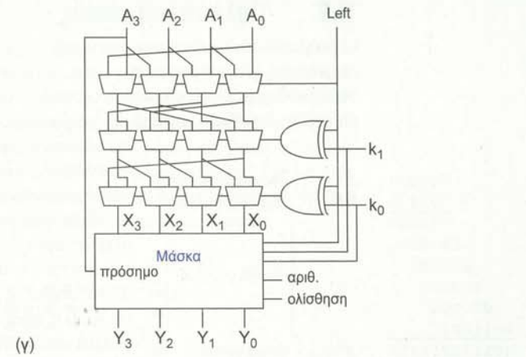
11.8.2 Περιστροφικός Ολισθητής

Ένας *περιστροφικός ολισθητής* (barrel shifter) εκτελεί μια πράξη περιστροφής δεξιά [Davis69]. Όπως αναφέραμε παραπάνω, χειρίζεται τις περιστροφές αριστερά χρησιμοποιώντας το συμπληρωματικό του ποσού ολίσθησης. Οι περιστροφικοί ολισθητές μπορούν επίσης να εκτελούν ολισθήσεις όταν περιλαμβάνεται το κατάλληλο hardware για μασκάρισμα (masking). Οι περιστροφικοί ολισθητές μπορούν να υλοποιούνται σε μορφή διάταξης (array) και σε λογαριθμική μορφή· εδώ θα επικεντρωθούμε στους λογαριθμικούς περιστροφικούς ολισθητές επειδή είναι καταλληλότεροι για μεγάλες ολισθήσεις.

Το Σχήμα 11.68(α) παρουσιάζει έναν απλό περιστροφικό ολισθητή των 4 bit, ο οποίος εκτελεί περιστροφές δεξιά. Παρατηρήστε ότι, ανόμοια με τους ολισθητές χοάνης, οι περιστροφικοί ολισθητές περιέχουν μεγάλου μήκους αγωγούς διασύνδεσης. Σ' ένα μεγάλο ολισθητή, είναι χρήσιμη η αύξηση μεγέθους ή η απομόνωση των οδηγών γι' αυτούς τους αγωγούς. Το Σχήμα 11.68(β) παρουσιάζει μια βελτιωμένη έκδοση, η οποία μπορεί να εκτελεί περιστροφή αριστερά μέσω προ-περιστροφής δεξιά κατά 1 και κατόπιν περιστροφής δεξιά κατά *k*. Η εκτέλεση λογικών ή αριθμητικών ολισθήσεων μ' έναν περιστροφικό ολισθητή απαιτεί έναν τρόπο για το μασκάρισμα των bit που εξέρχονται από το άκρο του ολισθητή, όπως παρουσιάζεται στο Σχήμα 11.68(γ).

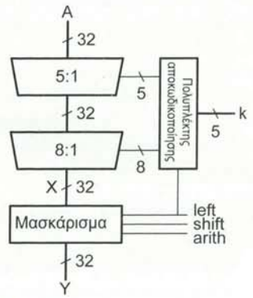
Το Σχήμα 11.69 παρουσιάζει έναν περιστροφικό ολισθητή των 32 bit, ο οποίος χρησιμοποιεί έναν πολυπλέκτη 5:1 κι έναν πολυπλέκτη 8:1. Το πρώτο στάδιο εκτελεί περιστροφή δεξιά κατά 0, 1, 2, 3, ή 4 bit, για το χειρισμό της προ-περιστροφής κατά 1 bit και κατόπιν περιστροφή υψηλότερης ανάλυσης έως τα 3 bit, συνδυασμένες σ' ένα στάδιο. Το δεύτερο στάδιο εκτελεί περιστροφή δεξιά κατά 0, 4, 8, 12, 16, 20, 24, ή 28 bit. Το κρίσιμο μονοπάτι ξεκινά με αποκωδικοποίηση του ποσού ολίσθησης για το πρώτο στάδιο. Εάν το ποσό ολίσθησης είναι διαθέσιμο από νωρίς, η καθυστέρηση από την είσοδο *A* στην έξοδο *Y* βελτιώνεται σημαντικά.



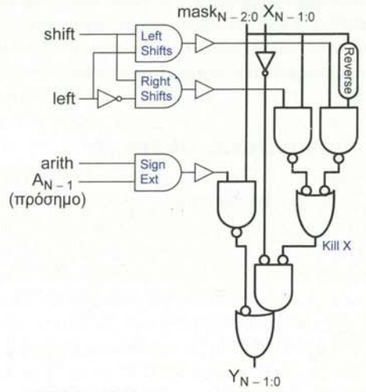


ΣΧΗΜΑ 11.68 Περιστροφικοί ολισθητές: (α) περιστροφή δεξιά, (β) περιστροφή αριστερά ή δεξιά, (γ) περιστροφές και ολισθήσεις

Κατά τη διάρκεια που λαμβάνει χώρα η περιστροφή, η μονάδα μασκαρίσματος παράγει μια μάσκα των *N* bit με ψηφία 1 στις θέσεις όπου θα έπρεπε να εισαχθεί η τιμή θανάτωσης για δεξιές ολισθήσεις. Για την εκτέλεση μιας δεξιάς ολίσθησης κατά *m*, τα *m* περισσότερο σημαντικά bit είναι 1. Αυτό αποκαλείται «κώδικας θερμομέτρου» (thermometer code) και η λογική για τον υπολογισμό του περιγράφεται στην Ενότητα 11.10. Όταν ολοκληρωθεί το αποτέλεσμα *X* της περιστροφής, η μονάδα μασκαρίσματος αντικαθιστά τα μασκαρισμένα bit με την τιμή θανάτωσης. Για αριστερές ολισθήσεις, η μάσκα αντιστρέφεται. Το Σχήμα 11.70 παρουσιάζει τη λογική μασκαρίσματος. Εάν υποστηρίζονται μόνο συγκεκριμένες ολισθήσεις, η μονάδα μπορεί να απλοποιηθεί, ενώ εάν υποστηρίζονται μόνο περιστροφές η μονάδα μασκαρίσματος μπορεί να εξαλειφθεί, πράγμα το οποίο σημαίνει σημαντική εξοικονόμηση σε hardware, ισχύ και καθυστέρηση.



ΣΧΗΜΑ 11.69 32-bit λογαριθμικός περιστροφικός ολισθητής.



ΣΧΗΜΑ 11.69 32-bit λογαριθμικός περιστροφικός ολισθητής.

11.8.3 Εναλλακτικές Λειτουργίες Ολίσθησης

Σε ορισμένες περιπτώσεις απαιτούνται άλλες παραλλαγές της λειτουργίας ολίσθησης, όπως ολίσθηση με ανακάτεμα (shuffle) των bit, αντιστροφές των bit, εναλλαγές θέσεων, εξαγωγές και αποθηκεύσεις των bit, κυρίως σε εφαρμογές κρυπτογραφίας και πολυμέσων [Hilewitz04, Hilewitz07]. Αυτές οι λειτουργίες μπορούν να υλοποιούνται με κατάλληλους συνδυασμούς πολυπλεκτών.

11.9 Πολλαπλασιασμός

Ο πολλαπλασιασμός είναι μια πράξη λιγότερο συνηθισμένη από την πρόσθεση, αλλά δεν παύει να είναι σημαντική για μικροεπεξεργαστές, ψηφιακούς επεξεργαστές σήματος και μηχανές γραφικών. Η πλέον στοιχειώδης μορφή πολλαπλασιασμού συνίσταται στο σχηματισμό του γινομένου δύο μη-προσημασμένων (θετικών) δυαδικών αριθμών. Αυτό μπορεί να πραγματοποιηθεί με την παραδοσιακή τεχνική που διδάσκεται στο δημοτικό σχολείο, απλοποιημένη στη βάση του 2. Για παράδειγμα, ο πολλαπλασιασμός δύο θετικών ακεραίων αριθμών των 6 bit, 25_{10} και 39_{10}, παρουσιάζεται στο Σχήμα 11.71.

```
   011001 : 25₁₀   Πολλαπλασιαστέος
 × 100111 : 39₁₀   Πολλαπλασιαστής
  011001
 011001
011001              Μερικά γινόμενα
000000
000000
+011001
001111001111 : 975₁₀   Τελικό γινόμενο
```

ΣΧΗΜΑ 11.71 Παράδειγμα πολλαπλασιασμού.

Ο πολλαπλασιασμός $M \times N$ bit, $P = Y \times X$, μπορεί να αντιμετωπιστεί ως σχηματισμός N μερικών γινομένων των M bit έκαστο και κατόπιν πρόσθεση των κατάλληλα ολισθημένων μερικών γινομένων για την παραγωγή ενός αποτελέσματος P των $M+N$ bit. Ο δυαδικός πολλαπλασιασμός ισοδυναμεί με τη λογική πράξη AND. Συνεπώς, η παραγωγή των μερικών γινομένων συνίσταται στο λογικό AND των κατάλληλων bit του πολλαπλασιαστή και του πολλαπλασιαστέου. Στη συνέχεια, κάθε στήλη μερικών γινομένων πρέπει να προστεθεί και, εάν είναι αναγκαίο, τυχόν κρατούμενα περνιούνται στην επόμενη στήλη. Ορίζουμε τον πολλαπλασιαστέο ως $Y = \{y_{M-1}, y_{M-2}, \ldots, y_1, y_0\}$ και τον πολλαπλασιαστή ως $X = \{x_{N-1}, x_{N-2}, \ldots, x_1, x_0\}$. Για πολλαπλασιασμό μη-προσημασμένων αριθμών, το αποτέλεσμα δίνεται από την Εξ. (11.31). Το Σχήμα 11.72 απεικονίζει την παραγωγή, ολίσθηση και πρόσθεση των μερικών γινομένων σ' έναν πολλαπλασιαστή 6×6 bit.

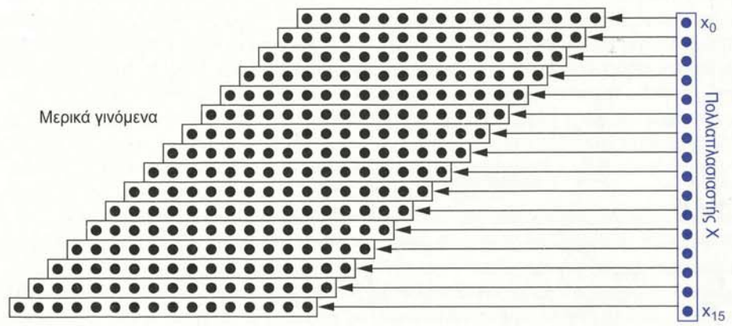
$$P = \left(\sum_{j=0}^{M-1} y_j 2^j \right) \left(\sum_{i=0}^{N-1} x_i 2^i \right) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_i y_j 2^{i+j} \qquad (11.31)$$

						y_5	y_4	y_3	y_2	y_1	y_0	Πολλαπλασιαστέος
						x_5	x_4	x_3	x_2	x_1	x_0	Πολλαπλασιαστής
						x_0y_5	x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0	Μερικά γινόμενα
					x_1y_5	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0		
				x_2y_5	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0			
			x_3y_5	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0				
		x_4y_5	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0					
	x_5y_5	x_5y_4	x_5y_3	x_5y_2	x_5y_1	x_5y_0						
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	Τελικό γινόμενο

ΣΧΗΜΑ 11.72 Παραγωγή των μερικών γινομένων.

Οι πολλαπλασιασμοί μεγάλων αριθμών απεικονίζονται πιο εύκολα με τη χρήση διαγραμμάτων κουκίδων. Το Σχήμα 11.73 παρουσιάζει ένα διάγραμμα κουκίδων για έναν απλό πολλαπλασιαστή 16×16. Κάθε κουκίδα αντιπροσωπεύει μια θέση για ένα μόνο bit, το οποίο μπορεί να είναι 0 ή 1. Τα μερικά γινόμενα αναπαριστώνται από μια γραμμή κουκίδων (μέσα σε πλαίσιο), ολισθημένη σύμφωνα με την αξία τους. Τα bit του πολλαπλασιαστή που χρησιμοποιούνται για την παραγωγή των μερικών γινομένων υποδεικνύονται στα δεξιά.

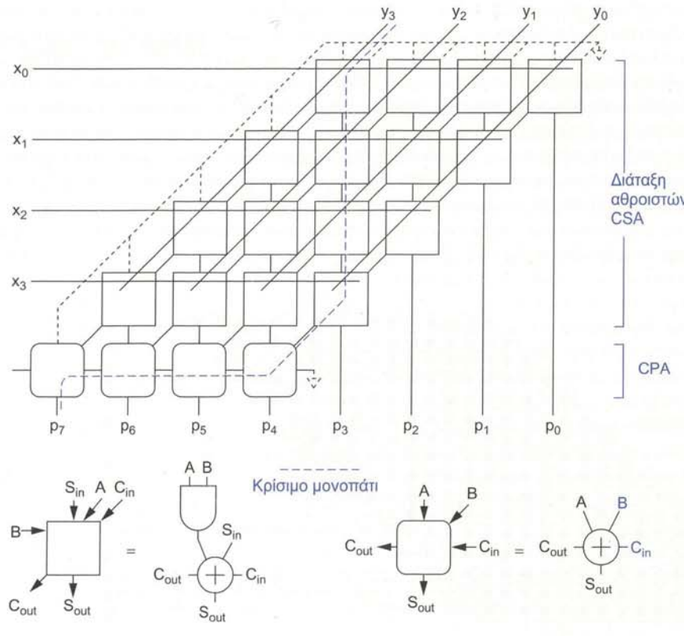
Υπάρχουν διάφορες τεχνικές που μπορούν να χρησιμοποιηθούν για την εκτέλεση του πολλαπλασιασμού. Γενικά, η επιλογή μεταξύ αυτών βασίζεται σε παράγοντες όπως η καθυστέρηση, ο ρυθμός λειτουργίας, η ενέργεια, το εμβαδό του κυκλώματος και η πολυπλοκότητα του hardware. Μια προφανής προσέγγιση είναι να χρησιμοποιήσουμε έναν αθροιστή διάδοσης κρατουμένου (CPA) των M+1 bit για να αθροίσουμε τα δύο πρώτα μερικά γινόμενα, στη συνέχεια άλλον ένα CPA για να προσθέσουμε το τρίτο μερικό γινόμενο στο τρέχον άθροισμα, κ.ο.κ. Μια τέτοια προσέγγιση χρειάζεται N-1 αθροιστές CPA και είναι αργή, ακόμα κι αν χρησιμοποιηθούν γρήγοροι CPA. Πιο αποδοτικά παράλληλα σχήματα χρησιμοποιούν κάποιο είδος πίνακα ή δένδρου πλήρων αθροιστών για την πρόσθεση των μερικών γινομένων. Ξεκινάμε μ' έναν απλό πίνακα για πολλαπλασιαστές μη-προσημασμένων αριθμών και κατόπιν τον τροποποιούμε κατάλληλα για το χειρισμό προσημασμένων αριθμών σε συμπλήρωμα ως προς 2, χρησιμοποιώντας τον αλγόριθμο Baugh-Wooley. Ο αριθμός των μερικών γινομένων που πρόκειται να προστεθούν μπορεί να μειωθεί με χρήση της κωδικοποίησης Booth και ο αριθμός των λογικών επιπέδων που απαιτούνται για την εκτέλεση της πρόσθεσης μπορεί να μειωθεί με χρήση δένδρων Wallace. Δυστυχώς, τα δένδρα Wallace απαιτούν πολύπλοκα φυσικά σχέδια και έχουν μεγάλου μήκους αγωγούς διασύνδεσης, χωρίς κανονικότητα. Για το λόγο αυτό, υβριδικές δομές πινάκων/δένδρων μπορεί να είναι πιο ελκυστικές επιλογές. Για λόγους πληρότητας, θα εξετάσουμε μια αρχιτεκτονική σειριακού πολλαπλασιαστή η οποία ήταν δημοφιλής στο παρελθόν, όταν οι πύλες είχαν σχετικά μεγάλο κόστος, αλλά σήμερα σπανίως είναι αναγκαία.



ΣΧΗΜΑ 11.73 Διάγραμμα κουκίδων για τον πολλαπλασιασμό.

11.9.1 Πολλαπλασιασμός Μη-Προσημασμένων Αριθμών

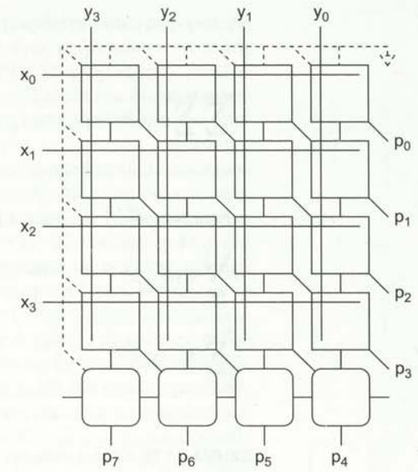
Οι γρήγοροι πολλαπλασιαστές χρησιμοποιούν αθροιστές αποθήκευσης κρατουμένου (CSA, δείτε την Ενότητα 11.2.4) για την πρόσθεση των μερικών γινομένων. Τυπικά, ένας CSA έχει καθυστέρηση της τάξης των 1.5-2 αντιστροφέων FO4 ανεξάρτητα από το εύρος του μερικού γινομένου, ενώ ένας αθροιστής διάδοσης κρατουμένου (CPA) τείνει να έχει καθυστέρηση της τάξης των 4-15+ αντιστροφέων FO4, εξαρτώμενη από το εύρος, την αρχιτεκτονική και την οικογένεια του κυκλώματος. Το Σχήμα 11.74 παρουσιάζει έναν πολλαπλασιαστή 4×4 για μη-προσημασμένους αριθμούς, ο οποίος χρησιμοποιεί μια διάταξη από CSA. Κάθε κύτταρο περιλαμβάνει μια πύλη AND δύο εισόδων, η οποία σχηματίζει ένα μερικό γινόμενο, κι έναν πλήρη αθροιστή (CSA) για την πρόσθεση του μερικού γινομένου στο τρέχον σύνολο. Η πρώτη γραμμή μετατρέπει το πρώτο μερικό γινόμενο σε πλεονασματική μορφή αθροίσματος-κρατουμένου. Κάθε γραμμή που ακολουθεί στη συνέχεια χρησιμοποιεί τον CSA για να προσθέσει το αντίστοιχο μερικό γινόμενο στο αποτέλεσμα (πλεονασματικής μορφής αθροίσματος-κρατουμένου) της προηγούμενης γραμμής και παράγει ένα αποτέλεσμα επίσης σε πλεονασματική μορφή αθροίσματος-κρατουμένου. Τα N λιγότερο σημαντικά bit της εξόδου είναι διαθέσιμα ως έξοδοι αθροίσματος απευθείας από τους CSA. Τα σημαντικότερα bit του αποτελέσματος φτάνουν σε πλεονασματική μορφή αθροίσματος-κρατουμένου και χρειάζονται έναν αθροιστή διάδοσης κρατουμένου (CPA) των M bit για να μετατραπούν σε κανονική δυαδική μορφή. Στο σχήμα, ο CPA υλοποιείται ως αθροιστής κύματος (διάδοσης) κρατουμένου. Η διάταξη έχει κανονικότητα και χρησιμοποιεί έναν μόνο τύπο κυττάρου, πράγμα το οποίο σημαίνει ότι είναι εύκολο να σχεδιαστεί και να παραχθεί το φυσικό σχέδιο. Υποθέτοντας ότι η έξοδος κρατουμένου είναι γρηγορότερη από την έξοδο του αθροίσματος σ' έναν CSA, το κρίσιμο μονοπάτι διαμέσου της διάταξης επισημαίνεται στο σχήμα με μια διακεκομμένη γραμμή. Ο αθροιστής μπορεί εύκολα να γίνει συνεχούς διοχέτευσης (pipeline) με την τοποθέτηση καταχωρητών ανάμεσα στις γραμμές. Στην πράξη, τα κυκλώματα προσδιορίζονται ως ορθογώνια μπλοκ στη χωροθέτηση, πράγμα το οποίο σημαίνει ότι το παραλληλόγραμμο σχήμα σπαταλά χώρο. Το Σχήμα 11.75 παρουσιάζει τους ίδιους αθροιστές μετατοπισμένους ώστε να ταιριάζουν σε ορθογώνιο σχήμα.



ΣΧΗΜΑ 11.74 Διάταξη πολλαπλασιαστή

Το στοιχείο που παίζει ρόλο-κλειδί στη σχεδίαση είναι ένας συμπαγής CSA. Δεν παρέχει πλεονεκτήματα μόνο σε σχέση με την επιφάνεια του κυκλώματος, αλλά βοηθά επίσης την απόδοση, επειδή οδηγεί σε μικρού μήκους αγωγούς με μικρή χωρητικότητα διασύνδεσης. Μια ιδανική σχεδίαση CSA έχει περίπου ίση καθυστέρηση αθροίσματος και κρατουμένου, επειδή η μεγαλύτερη από αυτές τις δύο καθυστερήσεις καθορίζει την απόδοση. Ο κατοπτρικός αθροιστής του Σχήματος 11.4 χρησιμοποιείται συνήθως λόγω της συμπαγούς χωροθέτησής του, παρόλο που η καθυστέρηση του αθροίσματος υπερβαίνει την καθυστέρηση του κρατουμένου. Η έξοδος του αθροίσματος μπορεί να συνδεθεί στην ταχύτερη είσοδο κρατουμένου για να αντισταθμιστεί εν μέρει αυτή η καθυστέρηση [Sutherland99, Hsu06a].

Παρατηρήστε ότι η πρώτη γραμμή των CSA προσθέτει το πρώτο μερικό γινόμενο σ' ένα ζεύγος από «0». Αυτό οδηγεί σε μια δομή με κανονικότητα, αλλά είναι αναποτελεσματικό. Μ' ένα ελάχιστο τίμημα όσον αφορά την κανονικότητα, η πρώτη γραμμή των CSA μπορεί να χρησιμοποιηθεί για την πρόσθεση των πρώτων τριών μερικών γινομένων μαζί. Αυτό μειώνει τον αριθμό των γραμμών κατά 2 και, κατ' επέκταση, μειώνει την καθυστέρηση διάδοσης του αθροιστή. Ένας άλλος τρόπος για να βελτιώσουμε την επίδοση του πολλαπλασιαστή διάταξης είναι να αντικαταστήσουμε την κατώτατη γραμμή μ' έναν ταχύτερο CPA, όπως π.χ. έναν αθροιστή πρόβλεψης κρατουμένου ή έναν αθροιστή δομής δένδρου. Συνολικά, το κρίσιμο μονοπάτι ενός πολλαπλασιαστή διάταξης περιλαμβάνει N-2 αθροιστές CSA κι έναν CPA.



ΣΧΗΜΑ 11.75 Ορθογώνια διάταξη πολλαπλασιαστή

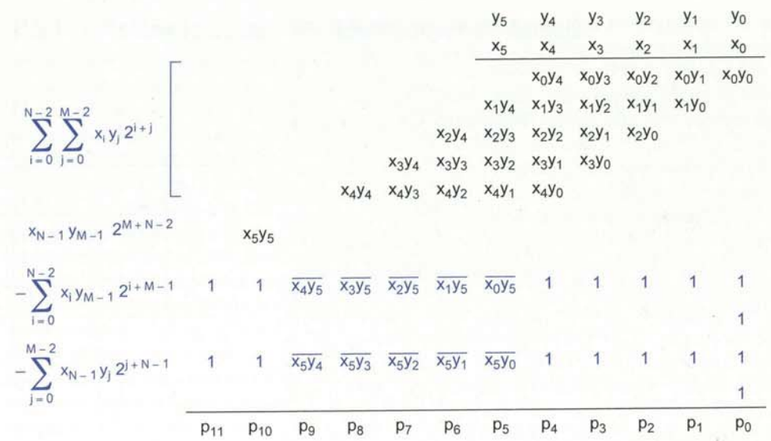
11.9.2 Πολλαπλασιασμός Αριθμών σε Συμπλήρωμα ως Προς 2

Σε πρώτη εξέταση, ο πολλαπλασιασμός αριθμών σε συμπλήρωμα ως προς 2 δείχνει δυσκολότερος, επειδή κάποια μερικά γινόμενα είναι αρνητικά και πρέπει να αφαιρεθούν. Θυμηθείτε ότι το πιο σημαντικό bit ενός αριθμού σε συμπλήρωμα ως προς 2 έχει αρνητική αξία. Συνεπώς, το γινόμενο είναι:

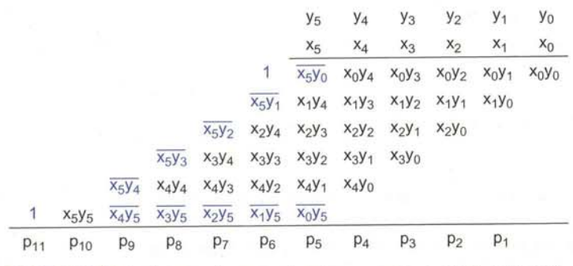
$$P = \left(-y_{M-1}2^{M-1} + \sum_{j=0}^{M-2} y_j 2^j\right)\left(-x_{n-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i\right)$$
$$= \sum_{i=0}^{N-2}\sum_{j=0}^{M-2} x_i y_j 2^{i+j} + x_{N-1}y_{M-1}2^{M+N-2} - \left(\sum_{i=0}^{N-2} x_i y_{M-1} 2^{i+M-1} + \sum_{j=0}^{M-2} x_{N-1} y_j 2^{j+N-1}\right) \quad (11.32)$$

Στην Εξ. (11.32), δύο από τα μερικά γινόμενα έχουν αρνητική αξία, οπότε θα πρέπει να αφαιρεθούν, όχι να προστεθούν. Ο αλγόριθμος *Baugh-Wooley* [Baugh73] του πολλαπλασιαστή χειρίζεται την αφαίρεση υπολογίζοντας το συμπλήρωμα ως προς 2 των όρων που πρόκειται να αφαιρεθούν (για παράδειγμα, αναστρέφοντας τα bit και προσθέτοντας 1). Το Σχήμα 11.76 παρουσιάζει τα μερικά γινόμενα που πρέπει να προστεθούν. Το επάνω παραλληλόγραμμο αντιπροσωπεύει το μη-προσημασμένο πολλαπλασιασμό όλων των bit, εκτός από τα μεγαλύτερης αξίας. Η επόμενη γραμμή αποτελείται από ένα μόνο bit, που αντιστοιχεί στο γινόμενο των πιο σημαντικών bit. Τα δύο επόμενα ζεύγη γραμμών είναι οι ανεστραμμένοι όροι που πρόκειται να αφαιρεθούν. Κάθε όρος έχει (υπονοούμενα) μηδενικά στην αρχή και στο τέλος του, τα οποία με την αντιστροφή μετατρέπονται σε 1. Επιπλέον ψηφία 1 πρέπει να προστίθενται στη μικρότερης αξίας στήλη όταν υπολογίζεται το συμπλήρωμα ως προς 2.

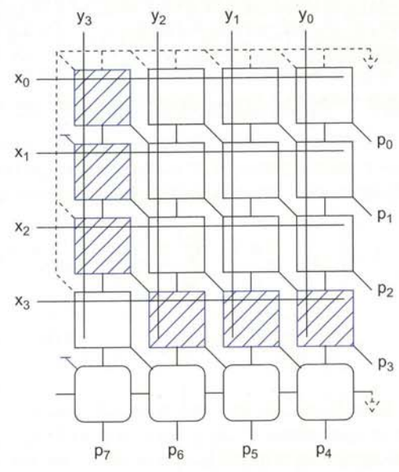
Η καθυστέρηση του πολλαπλασιαστή εξαρτάται από τον αριθμό των γραμμών των μερικών γινομένων που πρόκειται να προστεθούν. Ο τροποποιημένος πολλαπλασιαστής Baugh-Wooley [Hatamian86] μειώνει αυτό τον αριθμό των μερικών γινομένων, προ-υπολογίζοντας τα αθροίσματα των σταθερών 1 και μεταθέτοντας κάποιους όρους προς τα επάνω, σε επιπλέον στήλες. Το Σχήμα 11.77 παρουσιάζει μια τέτοια τοπολογία. Και σ' αυτή την περίπτωση, το παραλληλόγραμμο σχήμα μπορεί να προσαρμοστεί σε ορθογώνιο (βλ. Σχήμα 11.78), δίνοντας μια σχεδίαση σχεδόν πανομοιότυπη μ' αυτή του μη-προσημασμένου



ΣΧΗΜΑ 11.76 Μερικά γινόμενα για τον πολλαπλασιαστή σε συμπλήρωμα ως προς 2.



ΣΧΗΜΑ 11.77 Απλοποιημένα μερικά γινόμενα για τον πολλαπλασιαστή σε συμπλήρωμα ως προς 2.=



ΣΧΗΜΑ 11.78 Τροποποιημένος πολλαπλασιαστής σε συμπλήρωμα ως προς 2, των Baugh-Wooley.

πολλαπλασιαστή του Σχήματος 11.75. Οι πύλες AND αντικαθίστανται από NAND στα διαγραμμισμένα κύτταρα και προστίθενται 1 στη θέση των 0 σε δύο από τις αχρησιμοποίητες εισόδους. Οι διατάξεις πολλαπλασιαστών προσημασμένων και μη-προσημασμένων αριθμών είναι τόσο παρόμοιες, που μπορεί να χρησιμοποιηθεί μόνο μία διάταξη και για τις δύο λειτουργίες, εάν χρησιμοποιηθούν πύλες XOR για την υπό συνθήκη αντιστροφή ορισμένων από τους όρους, ανάλογα με το είδος του πολλαπλασιασμού.

11.9.3 Κωδικοποίηση Booth

Οι πολλαπλασιαστές διάταξης που εξετάσαμε στις προηγούμενες ενότητες υπολογίζουν τα μερικά γινόμενα σε 2η τάξη – δηλαδή, παρατηρώντας ένα ψηφίο του πολλαπλασιαστή ανά πάσα στιγμή. Οι πολλαπλασιαστές τάξης 2^r παράγουν N/r μερικά γινόμενα, καθένα από τα οποία εξαρτάται από

r bits του πολλαπλασιαστή. Τα λιγότερα μερικά γινόμενα οδηγούν σε μικρότερη και γρηγορότερη διάταξη αθροιστών αποθήκευσης κρατουμένου (CSA). Για παράδειγμα, ένας πολλαπλασιαστής 4ης τάξης παράγει $N/2$ μερικά γινόμενα. Κάθε μερικό γινόμενο είναι 0, Y, $2Y$ ή $3Y$, εξαρτώμενο από ένα ζεύγος bit του X. Ο υπολογισμός του $2Y$ είναι μια απλή ολίσθηση, αλλά το $3Y$ είναι ένα δύσκολο στον υπολογισμό πολλαπλάσιο, το οποίο απαιτεί μια αργή πρόσθεση διάδοσης κρατουμένου για τον υπολογισμό του $Y+2Y$ πριν ξεκινήσει η παραγωγή των μερικών γινομένων.

Η *κωδικοποίηση Booth* προτάθηκε αρχικά ως λύση για την επιτάχυνση του σειριακού πολλαπλασιασμού [Booth51]. Η *τροποποιημένη κωδικοποίηση* Booth [MacSorley61] επιτρέπει παράλληλη λειτουργία σε υψηλότερη τάξη, χωρίς την παραγωγή του δύσκολου στον υπολογισμό πολλαπλασίου $3Y$, χρησιμοποιώντας αντ' αυτού αρνητικά μερικά γινόμενα. Παρατηρήστε ότι $3Y = 4Y - Y$ και $2Y = 4Y - 2Y$. Ωστόσο, το $4Y$ σ' ένα πολλαπλασιαστή διάταξης 4ης τάξης είναι ισοδύναμο με το Y στην επόμενη γραμμή της διάταξης, που έχει τετραπλάσια αξία. Έτσι, τα μερικά γινόμενα επιλέγονται λαμβάνοντας υπόψη ένα ζεύγος bit μαζί με το πιο σημαντικό bit από το προηγούμενο ζεύγος. Εάν το πιο σημαντικό bit από το προηγούμενο ζεύγος είναι 1, το Y πρέπει να προστεθεί στο τρέχον μερικό γινόμενο. Εάν το πιο σημαντικό bit του τρέχοντος ζεύγους είναι 1, το τρέχον μερικό γινόμενο επιλέγεται να είναι αρνητικό και αυξάνεται το επόμενο μερικό γινόμενο.

Ο Πίνακας 11.12 δείχνει πώς επιλέγονται τα μερικά γινόμενα, με βάση τα bit του πολλαπλασιαστή. Τα αρνητικά μερικά γινόμενα δημιουργούνται υπολογίζοντας το συμπλήρωμα ως προς 2 του πολλαπλασιαστέου (πιθανώς με αριστερή ολίσθηση κατά μια στήλη για $-2Y$). Ένας κωδικοποιημένος κατά Booth, 4ης τάξης πολλαπλασιαστής μη-προσημασμένων αριθμών απαιτεί [$(N + 1) / 2$] μερικά γινόμενα αντί για N. Κάθε μερικό γινόμενο έχει $M + 1$ bit ώστε να χωρά τα πολλαπλάσια $2Y$ και $-2Y$. Παρότι τα X και Y είναι μη-προσημασμένα, τα μερικά γινόμενα μπορεί να είναι αρνητικά, οπότε πρέπει να επεκτείνονται ως προς το πρόσημο.

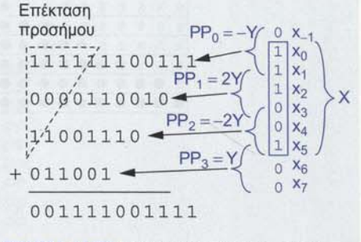
ΠΙΝΑΚΑΣ 11.12 Τιμές κωδικοποίησης Booth για πολλαπλασιαστή Booth 4ης τάξης

Είσοδοι			Μερικό γινόμενο	Σήματα επιλογής		
x_{2i+1}	x_{2i}	x_{2i-1}	PP_i	$SINGLE_i$	$DOUBLE_i$	NEG_i
0	0	0	0	0	0	0
0	0	1	Y	1	0	0
0	1	0	Y	1	0	0
0	1	1	$2Y$	0	1	0
1	0	0	$-2Y$	0	1	1
1	0	1	$-Y$	1	0	1
1	1	0	$-Y$	1	0	1
1	1	1	-0 (= 0)	0	0	1

Παράδειγμα 11.3

Επαναλάβετε τον πολλαπλασιασμό $P = Y \times X = 011001_2 \times 100111_2$ από το Σχήμα 11.71, εφαρμόζοντας κωδικοποίηση Booth για να μειώσετε τον αριθμό των μερικών γινομένων.

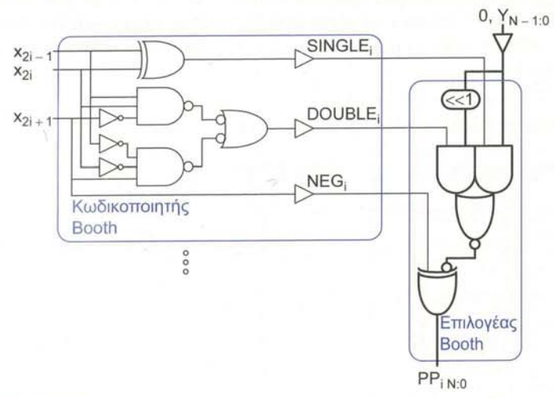
ΛΥΣΗ: Το Σχήμα 11.79 παρουσιάζει τον πολλαπλασιασμό. Το X γράφεται κατακόρυφα και τα bit χρησιμοποιούνται για την επιλογή των τεσσάρων μερικών γινομένων. Κάθε μερικό γινόμενο μετατίθεται δύο στήλες αριστερά από το προηγούμενο επειδή έχει τετραπλάσια αξία. Τα ανώτερα bit επεκτείνονται ως προς το πρόσημο με ψηφία 1 για τα αρνητικά μερικά γινόμενα και 0 για τα θετικά. Στη συνέχεια τα μερικά γινόμενα προστίθενται για τον υπολογισμό του αποτελέσματος.



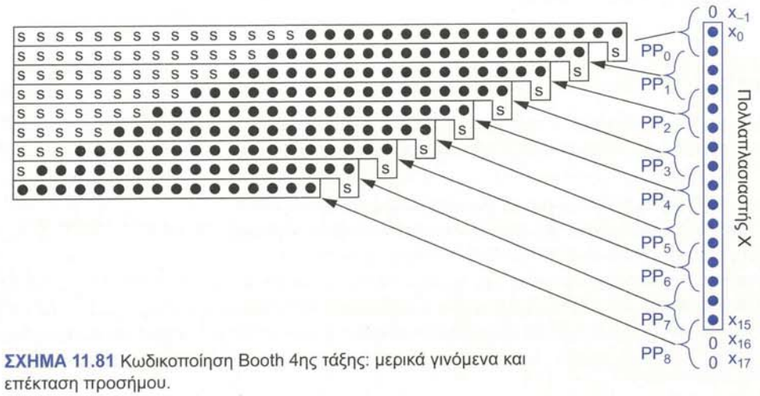
ΣΧΗΜΑ 11.79 Πολλαπλασιασμός με χρήση κωδικοποίησης Booth.

Σε μια τυπική σχεδίαση κωδικοποιημένου κατά Booth πολλαπλασιαστή 4ης τάξης, κάθε ομάδα τριών bit (ένα ζεύγος μαζί με το πιο σημαντικό ψηφίο του προηγούμενου ζεύγους) κωδικοποιείται σε αρκετές γραμμές επιλογής ($SINGLE_i$, $DOUBLE_i$ και NEG_i, οι τρεις δεξιότερες στήλες του Πίνακα 11.12) και οδηγείται κατά μήκος της γραμμής του μερικού γινομένου, όπως φαίνεται στο Σχήμα 11.80. Ο πολλαπλασιαστής Y είναι κατανεμημένος σε όλες τις γραμμές. Οι γραμμές επιλογής ελέγχουν τους επιλογείς Booth που επιλέγουν το κατάλληλο πολλαπλάσιο του Y για κάθε μερικό γινόμενο. Οι επιλογείς Booth αντικαθιστούν τις πύλες AND ενός απλού πολλαπλασιαστή διάταξης για τον υπολογισμό του i-οστού μερικού γινομένου. Το Σχήμα 11.80 παρουσιάζει μια συμβατική σχεδίαση επιλογέα και κωδικοποιητή Booth [Goto92]. Το Y επεκτείνεται ως προς το μηδέν σε $M+1$ bits. Ανάλογα με τα σήματα $SINGLE_i$ και $DOUBLE_i$, η πύλη A22OI επιλέγει 0, Y, ή $2Y$. Τα αρνητικά μερικά γινόμενα θα πρέπει να είναι σε μορφή συμπληρώματος ως προς 2 (δηλαδή, αντιστροφή και πρόσθεση του 1). Στην περίπτωση σήματος NEG_i, το μερικό γινόμενο αντιστρέφεται. Το επιπλέον 1 μπορεί να προστεθεί στη λιγότερο σημαντική στήλη της επόμενης γραμμής για να αποφευχθεί η ανάγκη χρήσης ενός CPA.

Ακόμα και σ' έναν πολλαπλασιαστή μη-προσημασμένων αριθμών, τα αρνητικά μερικά γινόμενα πρέπει να επεκτείνονται ως προς το πρόσημο για να προστίθενται σωστά. Το Σχήμα 11.81 παρουσιάζει έναν 16-bit πίνακα μερικών γινομένων Booth 4ης τάξης για έναν πολλαπλασιαστή μη-προσημασμένων αριθμών, σε μορφή διαγράμματος κουκίδων. Κάθε κουκίδα στον κωδικοποιημένο κατά Booth πολλαπλασιαστή παράγεται από έναν επιλογέα Booth, αντί από μια απλή πύλη AND. Τα μερικά γινόμενα 0–7



ΣΧΗΜΑ 11.80 Κωδικοποιητής και επιλογέας Booth, 4ης τάξης.



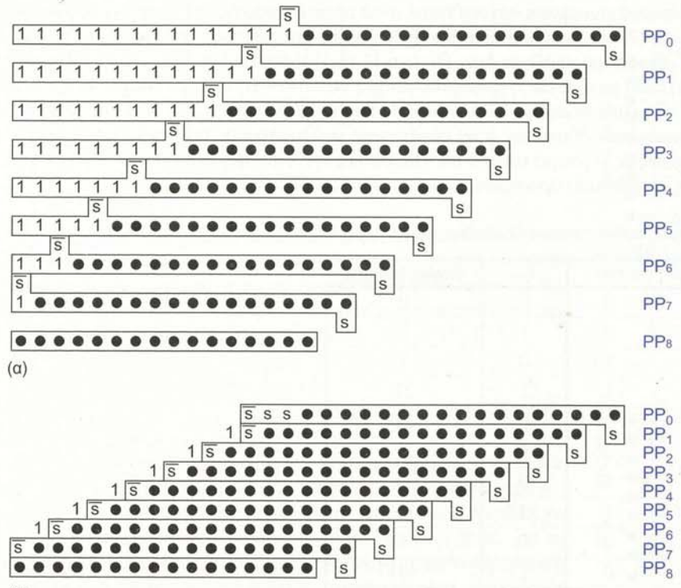
ΣΧΗΜΑ 11.81 Κωδικοποίηση Booth 4ης τάξης: μερικά γινόμενα και επέκταση προσήμου.

είναι 17 bits. Κάθε μερικό γινόμενο i επεκτείνεται ως προς το πρόσημο με $s_i = NEG_i = x_{2i+1}$, το οποίο είναι 1 για αρνητικά πολλαπλάσια (αυτά στο κάτω μισό του Πίνακα 11.12), ή 0 για θετικά πολλαπλάσια. Παρατηρήστε ότι προστίθεται μια επιπλέον μονάδα στο λιγότερο σημαντικό bit στην επόμενη γραμμή για να σχηματιστεί το συμπλήρωμα ως προς 2 των αρνητικών πολλαπλασίων. Η αντιστροφή των (υπονοούμενων) μηδενικών στην αρχή και στο τέλος παράγει μονάδες για τα αρνητικά πολλαπλάσια. Οι επιπλέον όροι αυξάνουν το μέγεθος του πολλαπλασιαστή. Το PP_8 απαιτείται σε περίπτωση που το PP_7 είναι αρνητικό. Αυτό το μερικό γινόμενο είναι πάντα 0 ή Y, επειδή τα x_{16} και x_{17} είναι 0.

Παρατηρήστε ότι τα bit επέκτασης προσήμου είναι όλα είτε 1 είτε 0. Εάν προστίθεται μόνο ένα 1 στη λιγότερο σημαντική θέση σε μια ακολουθία από 1, το αποτέλεσμα είναι μια ακολουθία από 0 συν ένα κρατούμενο εξόδου, το υψηλότερης αξίας bit που μπορεί να απορριφθεί. Συνεπώς, ο μεγάλος αριθμός των s bits σε κάθε μερικό γινόμενο μπορεί να αντικατασταθεί από ισάριθμα σταθερά ψηφία 1 συν την αντιστροφή του s που προστίθεται στη λιγότερο σημαντική θέση, όπως υποδεικνύει το Σχήμα 11.82(α). Μπορούμε να βελτιστοποιήσουμε τη χρήση αυτών των σταθερών κυρίως έξω από τον πίνακα, προ-υπολογίζοντας το άθροισμά τους. Το απλοποιημένο αποτέλεσμα παρουσιάζεται στο Σχήμα 11.82(β). Ως συνήθως, μπορεί να μετατοπιστεί ώστε να έχει ορθογώνια χωροθέτηση.

Το κρίσιμο μονοπάτι του πολλαπλασιαστή περιλαμβάνει τον αποκωδικοποιητή Booth, τους οδηγούς γραμμών επιλογής, τον επιλογέα Booth, περίπου $N/2$ αθροιστές CSA κι έναν τελικό CPA. Κάθε μερικό γινόμενο γεμίζει περίπου M+5 στήλες. Οι 54×54 πολλαπλασιαστές Booth 4ης τάξης για μονάδες IEEE κινητής υποδιαστολής διπλής ακρίβειας είναι συνήθως μικρότερης επιφάνειας, κατά 20% έως 50% (και κατ' επέκταση έως 20% γρηγορότεροι) από τους αντίστοιχους μη-κωδικοποιημένους και γι' αυτό η συγκεκριμένη τεχνική χρησιμοποιείται ευρέως. Ο πολλαπλασιαστής απαιτεί $M \times N/2$ επιλογείς Booth.

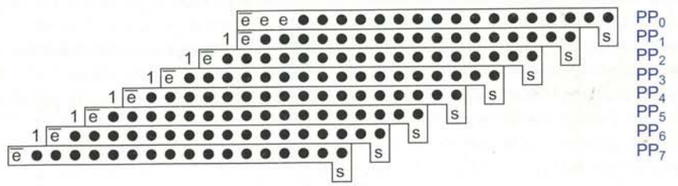
Επειδή οι επιλογείς καταλαμβάνουν σημαντικό μέρος του εμβαδού του κυκλώματος αλλά έχουν μικρή συμμετοχή στο κρίσιμο μονοπάτι, θα πρέπει να είναι βελτιστοποιημένοι περισσότερο ως προς το μέγεθος παρά ως προς την ταχύτητα. Για παράδειγμα, ο [Goto97] περιγράφει έναν κωδικοποιητή και επιλογέα Booth *επιλογής προσήμου* (sign select), ο οποίος χρησιμοποιεί μόνο 10 τρανζίστορ ανά bit του επιλογέα με αντίτιμο έναν πολυπλοκότερο κωδικοποιητή. Ο [Hsu06a] παρουσιάζει έναν κωδικοποιητή και επιλο-



ΣΧΗΜΑ 11.82 Κωδικοποίηση Booth 4ης τάξης: μερικά γινόμενα και απλοποιημένη επέκταση προσήμου.

γέα Booth ο οποίος επιλέγει ένα από τα έξι πιθανά μερικά γινόμενα χρησιμοποιώντας ένα πολυπλέκτη υλοποιημένο με πύλες μετάδοσης. Στην Άσκηση 11.18 θα διερευνήσετε μια άλλη μορφή κωδικοποίησης.

11.9.3.1 Πολλαπλασιαστές προσημασμένων αριθμών με κωδικοποίηση Booth Ο πολλαπλασιασμός προσημασμένων αριθμών σε συμπλήρωμα ως προς 2 είναι παρόμοιος, αλλά επειδή ο πολλαπλασιαστέος θα μπορούσε να είναι αρνητικός θα πρέπει να γίνει επέκταση προσήμου με βάση το bit προσήμου του μερικού γινομένου, PP_{iM} [Bewick94]. Το Σχήμα 11.83 παρουσιάζει μια τέτοια διάταξη, όπου το bit επέκτασης προσήμου είναι το e_i=PP_{iM}. Παρατηρήστε, επίσης ότι το PP_8, το οποίο ήταν είτε Y είτε 0 στον πολλαπλασιασμό μη-προσημασμένων αριθμών, είναι πάντα 0 και μπορεί να αγνοηθεί στον πολλαπλασιασμό προσημασμένων αριθμών, επειδή ο πολλαπλασιαστής x υφίσταται επέκταση προσήμου έτσι ώστε $x_{17} = x_{16} = x_{15}$. Μπορεί να χρησιμοποιηθεί ο ίδιος συνδυασμός επιλογέα και κωδικοποιητή Booth (βλ. Σχ. 11.80), αλλά το Y θα πρέπει να επεκταθεί ως προς το πρόσημο και όχι ως προς το μηδέν στα $M+1$ bits.



ΣΧΗΜΑ 11.83 Κωδικοποίηση Booth 4ης τάξης: μερικά γινόμενα για πολλαπλασιασμό προσημασμένων τιμών.

11.9.3.2 Κωδικοποίηση Booth υψηλότερης τάξης Οι μεγάλοι πολλαπλασιαστές μπορούν να χρησιμοποιούν κωδικοποίηση Booth υψηλότερης τάξης. Για παράδειγμα, ο κοινός πολλαπλασιασμός 8ης τάξης μειώνει τον αριθμό των μερικών γινομένων στο ένα τρίτο, αλλά απαιτεί δύσκολα στον υπολογισμό πολλαπλάσια του τύπου $3Y$, $5Y$ και $7Y$. Η κωδικοποίηση Booth 8ης τάξης απαιτεί μόνο το δύσκολο στον υπολογισμό πολλαπλάσιο $3Y$, όπως παρουσιάζεται στον Πίνακα 11.13. Παρότι αυτό απαιτεί έναν CPA πριν από την παραγωγή του μερικού γινομένου, η χρήση του μπορεί να αιτιολογηθεί λόγω της μείωσης του μεγέθους της διάταξης και της καθυστέρησης. Κωδικοποίηση Booth υψηλότερης τάξης είναι μεν εφικτή, αλλά η παραγωγή των υπολοίπων δύσκολων στον υπολογισμό πολλαπλασίων δείχνει να μην αξίζει τον κόπο για πολλαπλασιαστές με λιγότερα από 64 bit. Παρόμοιες τεχνικές εφαρμόζονται σε υψηλότερης τάξης πολλαπλασιαστές με επέκταση προσήμου.

ΠΙΝΑΚΑΣ 11.13 Τιμές κωδικοποίησης Booth 8ης τάξης

x_{i+2}	x_{i+1}	x_i	x_{i-1}	Μερικό γινόμενο
0	0	0	0	0
0	0	0	1	Y
0	0	1	0	Y
0	0	1	1	$2Y$
0	1	0	0	$2Y$
0	1	0	1	$3Y$
0	1	1	0	$3Y$
0	1	1	1	$4Y$
1	0	0	0	$-4Y$
1	0	0	1	$-3Y$
1	0	1	0	$-3Y$

συνεχίζεται

ΠΙΝΑΚΑΣ 11.13 Τιμές κωδικοποίησης Booth 8ης τάξης (συνέχεια)

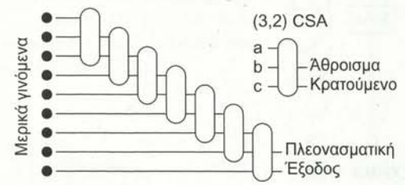
1	0	1	1	$-2Y$
1	1	0	0	$-2Y$
1	1	0	1	$-Y$
1	1	1	0	$-Y$
1	1	1	1	-0

11.9.4 Πρόσθεση Στηλών

Το κρίσιμο μονοπάτι σ' έναν πολλαπλασιαστή απαιτεί την πρόσθεση των κουκίδων σε κάθε στήλη. Παρατηρήστε ότι ένας αθροιστής αποθήκευσης κρατουμένου (CSA) είναι ένας «μετρητής των 1», ο οποίος προσθέτει τον αριθμό των 1 από τις εισόδους A, B και C και τα κωδικοποιεί στις εξόδους του αθροίσματος και του κρατουμένου, όπως συνοψίζεται στον Πίνακα 11.14. Ένας CSA αποκαλείται επίσης μετρητής (3,2) [Dadda65], επειδή μετατρέπει τρεις εισόδους σε μια μέτρηση κωδικοποιημένη σε δύο εξόδους. Το κρατούμενο εξόδου περνά στην επόμενη πιο σημαντική στήλη, ενώ ένα αντίστοιχο κρατούμενο εισόδου λαμβάνεται από την προηγούμενη στήλη. Συνεπώς, για λόγους απλότητας, ένα κρατούμενο αναπαρίσταται σα να διοχετεύεται κατευθείαν προς την επόμενη στήλη. Το Σχήμα 11.84 παρουσιάζει το διάγραμμα κουκίδων για μια στήλη ενός πολλαπλασιαστή διάταξης, ο οποίος αθροίζει διαδοχικά N μερικά γινόμενα χρησιμοποιώντας N-2 CSA. Η έξοδος παράγεται σε πλεονασματική μορφή αθροίσματος-κρατουμένου, κατάλληλη για τον τελικό αθροιστή διάδοσης κρατουμένου (CPA).

ΠΙΝΑΚΑΣ 11.14 Χρήση αθροιστή ως μετρητή των 1

A	B	C	Κρατούμενο	Άθροισμα	Πλήθος των 1
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3



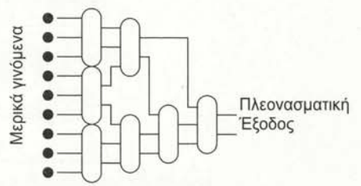
ΣΧΗΜΑ 11.84 Διάγραμμα κουκίδων για διάταξη πολλαπλασιαστή.

Η πρόσθεση στηλών είναι αργή, επειδή κάθε φορά χρησιμοποιείται μόνο ένας CSA. Ένας άλλος τρόπος για να επιταχύνουμε την πρόσθεση στηλών είναι να προσθέσουμε τα μερικά γινόμενα παράλληλα αντί διαδοχικά. Το Σχήμα 11.85 παρουσιάζει ένα δένδρο Wallace που χρησιμοποιεί αυτή την προσέγγιση [Wallace64]. Το δένδρο Wallace απαιτεί

$$\left\lceil \log_{3/2}\left(N/2\right) \right\rceil$$

επίπεδα μετρητών (3,2) για να μειώσει τις N εισόδους σε 2 εξόδους πλεονασματικής μορφής αθροίσματος-κρατουμένου.

Παρά το γεγονός ότι οι CSA στο δένδρο Wallace παρουσιάζονται στις δύο διαστάσεις, σε επίπεδο λογικής είναι «στοιβαγμένοι» σε μια μεμονωμένη στήλη του πολλαπλασιαστή. Αυτό έχει ως αποτέλεσμα η στήλη να διατρέχεται από αγωγούς μεγάλου μήκους, χωρίς κανονικότητα, για τη σύνδεση των CSA. Η χωρητικότητα αγωγών αυξάνει την καθυστέρηση και την ενέργεια του πολλαπλασιαστή· επιπλέον, μπορεί να γίνει πιο δύσκολη η διευθέτηση των αγωγών στο φυσικό σχέδιο.

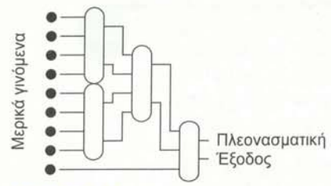


ΣΧΗΜΑ 11.85 Διάγραμμα κουκίδων για πολλαπλασιαστή δένδρου Wallace.

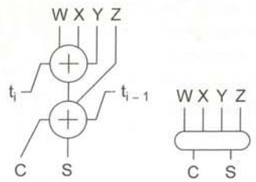
11.9.4.1 Δένδρα συμπιεστών [4:2] Οι συμπιεστές [4:2] μπορούν να χρησιμοποιηθούν σ' ένα δυαδικό δένδρο για να παράγουν μια πολύ πιο κανονική σχεδίαση, όπως υποδεικνύει το Σχήμα 11.86 [Weinberg81, Santoro89]. Ένας συμπιεστής [4:2] λαμβάνει τέσσερις εισόδους ίσης αξίας και παράγει δύο εξόδους. Μπορεί να κατασκευαστεί από δύο μετρητές (3,2) όπως παρουσιάζεται στο Σχήμα 11.87. Κατά τη διάρκεια της διαδικασίας, παράγει ένα ενδιάμεσο κρατούμενο προς την επόμενη στήλη και δέχεται ένα κρατούμενο από την προηγούμενη στήλη, οπότε θα ήταν πιο εύστοχο να αποκαλείται *μετρητής (5,3)*. Αυτό το οριζόντιο μονοπάτι δεν επηρεάζει την καθυστέρηση επειδή η έξοδος του κορυφαίου CSA μιας στήλης είναι η είσοδος του κατώτερου CSA της επόμενης στήλης. Ο συμβολισμός [4:2] CSA αναφέρει μόνο τις αρχικές εισόδους για να υπογραμμίσει την κύρια λειτουργία που επιτελείται – δηλαδή, τη μείωση των τεσσάρων εισόδων σε δύο εξόδους. Απαιτούνται μόνο

$$\left\lceil \log_2\left(N/2\right) \right\rceil$$

επίπεδα συμπιεστών [4:2], αν και ο καθένας έχει μεγαλύτερη καθυστέρηση από έναν CSA. Επίσης, η κανονικότητα του φυσικού σχεδίου και της διασύνδεσης καθιστά το δυαδικό δένδρο ελκυστική επιλογή.

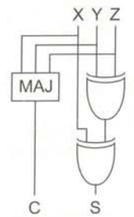


ΣΧΗΜΑ 11.86 Διάγραμμα κουκίδων για πολλαπλασιαστή δένδρου [4:2].

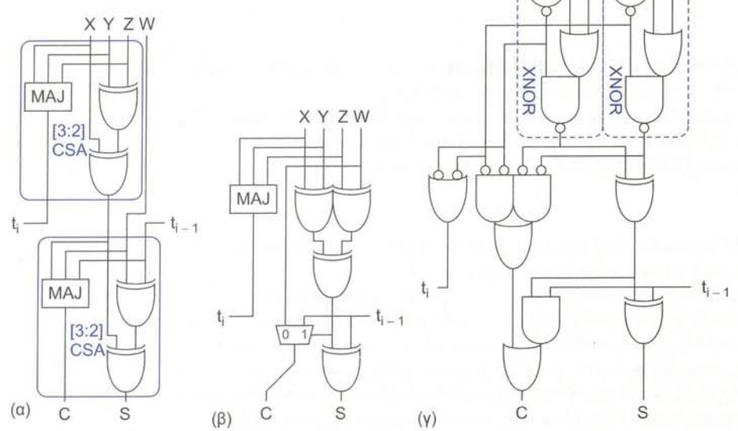


ΣΧΗΜΑ 11.87 Συμπιεστής [4:2] (α) υλοποίηση με δύο CSA (β) σύμβολο.

Για να κατανοήσετε τα πλεονεκτήματα ενός συμπιεστή [4:2], θα εισάγουμε την έννοια των γρήγορων και αργών εισόδων και εξόδων. Το Σχήμα 11.88 παρουσιάζει μια απλή σχεδίαση CSA σε επίπεδο πυλών. Το μεγαλύτερου μήκους μονοπάτι που διέρχεται από τον CSA χρησιμοποιεί δύο επίπεδα πυλών XOR2 για τον υπολογισμό του αθροίσματος. Η *X* αποκαλείται *γρήγορη είσοδος*, ενώ οι *Y* και *Z* είναι *αργές είσοδοι* επειδή περνούν από ένα δεύτερο επίπεδο πυλών XOR. Η *C* είναι η *γρήγορη έξοδος* επειδή απαιτεί μόνο μία καθυστέρηση πύλης, ενώ η *S* είναι η *αργή έξοδος* επειδή απαιτεί δύο καθυστερήσεις πύλης. Ένας συμπιεστής [4:2] θα μπορούσε να σχεδιαστεί ώστε να χρησιμοποιεί τέσσερα επίπεδα πυλών XOR2. Το Σχήμα 11.89 παρουσιάζει διάφορες σχεδιάσεις συμπιεστών [4:2], οι οποίες μειώνουν το κρίσιμο μονοπάτι σε 3 μόνο πύλες XOR2. Στο Σχήμα 11.89(α), η αργή έξοδος του πρώτου CSA συνδέεται στην γρήγορη είσοδο του δεύτερου. Στο Σχήμα 11.89(β), ο συμπιεστής [4:2] έχει μετασχηματιστεί σ' ένα μεμονωμένο κύτταρο, πράγμα το οποίο



ΣΧΗΜΑ 11.88 Αθροιστής αποθήκευσης κρατουμένου (σχηματικό σε επίπεδο πυλών).

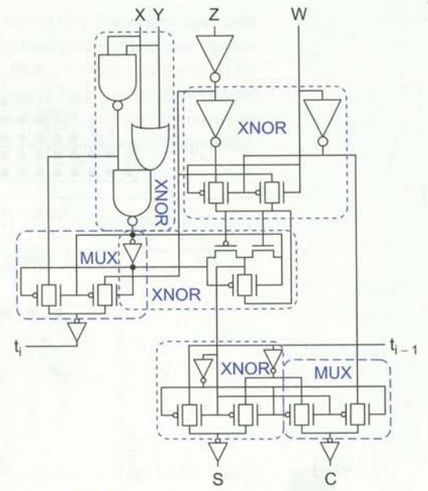


ΣΧΗΜΑ 11.89 Συμπιεστές [4:2].

επιτρέπει την αντικατάσταση μιας πύλης πλειοψηφίας μ' έναν πολυπλέκτη. Στο Σχήμα 11.89(γ), οι αρχικές XOR έχουν αντικατασταθεί από κυκλώματα XNOR 2 επιπέδων, τα οποία επιτρέπουν, σε κάποιο βαθμό, το διαμοιρασμό υπο-λειτουργιών μεταξύ των βαθμίδων, μειώνοντας το πλήθος των τρανζίστορ [Goto92].

Το Σχήμα 11.90 παρουσιάζει μια υλοποίηση συμπιεστή [4:2] με πύλες μετάδοσης, από τον [Goto97]. Χρησιμοποιεί μόνο 48 τρανζίστορ, οπότε επιτρέπει τη χρήση μικρότερης διάταξης, με αγωγούς μικρότερου μήκους. Σημειώστε ότι χρησιμοποιεί τρεις διαφορετικές κυκλωματικές μορφές XNOR και δύο πολυπλέκτες υλοποιημένους με πύλες μετάδοσης.

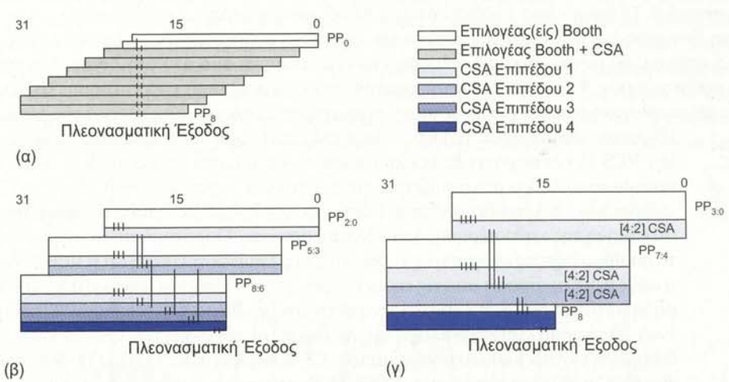
Το Σχήμα 11.91 συγκρίνει τις χωροθετήσεις της 16 x 16 διάταξης πολλαπλασιαστή με κωδικοποίηση Booth από το Σχήμα 11.84, του δένδρου Wallace από το Σχήμα 11.85 και του δένδρου [4:2] από το Σχήμα 11.86. Κάθε γραμμή αντιπροσωπεύει μια οριζόντια «φέτα» του πολλαπλασιαστή, η οποία περιέχει έναν επιλογέα Booth ή έναν CSA. Κατακόρυφοι δίαυλοι συνδέουν τους CSA. Το δένδρο Wallace έχει την πιο ακανόνιστη και μεγάλου μήκους διασύνδεση. Στην πράξη, το παραλληλόγραμμο μπορεί να μετασχηματιστεί σε ορθογώνιο για καλύτερη χρήση του χώρου. Οι [Itoh01n] και [Huang05] περιγράφουν ζητήματα χωροθέτησης τα οποία αφορούν πολλαπλασιαστές με δομή δένδρου.



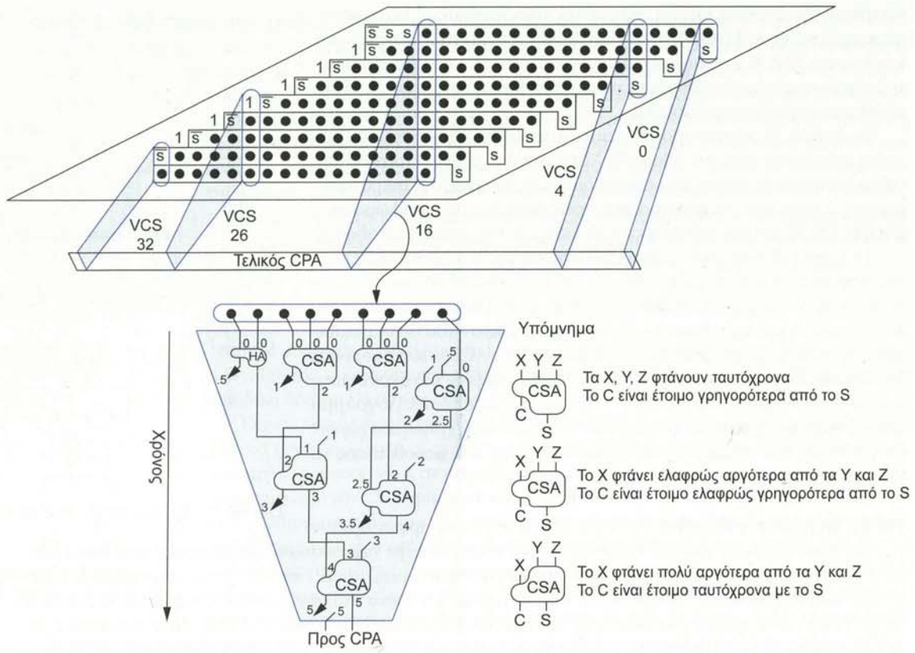
ΣΧΗΜΑ 11.90 Συμπιεστής [4:2] με πύλες μετάδοσης.

11.9.4.2 Τρισδιάστατη Μέθοδος Το σκεπτικό της σύνδεσης αργών εξόδων σε γρήγορες εισόδους μπορεί να γενικευτεί σε συμπιεστές με περισσότερες από τέσσερις εισόδους. Εξετάζοντας ολόκληρη τη διάταξη μερικών γινομένων μονομιάς, μπορεί κανείς να κατασκευάσει δένδρα για κάθε στήλη, τα οποία θα αθροίζουν όλα τα μερικά γινόμενα στο συντομότερο δυνατό χρόνο. Αυτή η προσέγγιση αποκαλείται *τρισδιάστατη μέθοδος* (three-dimensional method, TDM), επειδή αντιμετωπίζει το χρόνο άφιξης ως τρίτη διάσταση, μαζί με τις γραμμές και τις στήλες [Oklobdzija96, Stelling98].

Το Σχήμα 11.92 παρουσιάζει ένα παράδειγμα πολλαπλασιαστή 16 x 16. Το κορυφαίο παραλληλόγραμμο απεικονίζει το διάγραμμα κουκίδων του Σχήματος 11.82(β), το οποίο τώρα περιέχει εννέα γραμμές μερικών γινομένων που λαμβάνονται μέσω κωδικοποίησης Booth. Τα μερικά γινόμενα σε κάθε μία από τις 32 στήλες πρέπει να αθροιστούν για να παραχθεί το 32-bit αποτέλεσμα. Όπως έχουμε δει, αυτό γίνεται μ' ένα συμπιεστή ο οποίος παράγει ένα ζεύγος εξόδων και ακολουθείται από έναν τελευταίο CPA.



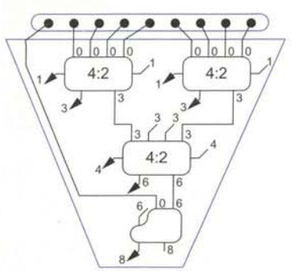
ΣΧΗΜΑ 11.91 Χωροθετήσεις για 16 × 16 πολλαπλασιαστή με κωδικοποίηση Booth: (α) διάταξη, (β) δένδρο Wallace, (γ) δένδρο [4:2].



ΣΧΗΜΑ 11.92 Κατακόρυφες φέτες συμπιεστών σ' έναν πολλαπλασιαστή TDM.

Με την τρισδιάστατη μέθοδο, κάθε στήλη αθροίζεται με τη χρήση μιας *κατακόρυφης φέτας συμπιεστών* (vertical compressor slice, VCS), η οποία αποτελείται από CSA. Στο Σχήμα 11.92, η VCS 16 προσθέτει εννέα μερικά γινόμενα. Στο διάγραμμα εμφανίζονται τα οριζόντια κρατούμενα μεταξύ των φετών συμπιεστών.

Κάθε αγωγός τιτλοφορείται με το χρόνο άφιξής του. Όλες οι είσοδοι μερικών γινομένων φθάνουν τη χρονική στιγμή 0. Το διάγραμμα υποθέτει ότι μια XOR2 και μια πύλη πλειοψηφίας έχουν μοναδιαία καθυστέρηση. Συνεπώς, ένα μονοπάτι διερχόμενο από έναν CSA απ' οποιαδήποτε είσοδο έως το *C*, ή από το *X* έως το *S*, απαιτεί μία μοναδιαία καθυστέρηση, ενώ ένα μονοπάτι από το *Y* στο *Z* ή στο *S* απαιτεί δύο μοναδιαίες καθυστερήσεις. Ένας ημιαθροιστής υποτίθεται ότι έχει τη μισή καθυστέρηση. Τα οριζόντια κρατούμενα αναπαρίστανται με διαγώνιες γραμμές προερχόμενες από την πίσω πλευρά της φέτας ή προεξέχοντας από τη φέτα. Η VCS 16 λαμβάνει πέντε οριζόντια κρατούμενα εισόδου από την VCS 15 και παράγει έξι οριζόντια κρατούμενα εξόδου προς την VCS 17. Το τελικό κρατούμενο εξόδου μετατοπίζεται επίσης κατά μία στήλη πριν οδηγηθεί στον CPA. Οι είσοδοι των CSA διευθετούνται με βάση τους χρόνους άφιξής τους, με στόχο την ελαχιστοποίηση της καθυστέρησης του πολλαπλασιαστή. Παρατηρήστε το σχήμα των CSA, το οποίο τονίζει τις ασύμμετρες καθυστερήσεις. Σημειώστε επίσης ότι η VCS 16 δεν είναι η αργότερη· ορισμένες από τις επόμενες φέτες έχουν μία επιπλέον μοναδιαία καθυστέρηση επειδή τα οριζόντια κρατούμενα φθάνουν αργότερα. Ο [Oklobdzija96] περιγράφει έναν αλγόριθμο για την επιλογή της ταχύτερης διευθέτησης των CSA σε κάθε VCS, με δεδομένες τυχαίες καθυστερήσεις για τους CSA. Συγκριτικά, το Σχήμα 11.93 παρουσιάζει την ίδια VCS 16 να χρησιμοποιεί CSA [4:2]· απαιτούνται περισσότερα επίπεδα πυλών XOR, αλλά η διασύνδεση έχει αυξημένη κανονικότητα.



ΣΧΗΜΑ 11.93 Κατακόρυφη "φέτα" συμπιεστών [4:2].

Ο Πίνακας 11.15 παρουσιάζει τον αριθμό των επιπέδων XOR στο κρίσιμο μονοπάτι για διάφορους αριθμούς μερικών γινομένων. Τα δένδρα [4:2] αποτελούν σημαντική βελτίωση έναντι των δένδρων Wallace από τη σκοπιά των λογικών επιπέδων, καθώς επίσης και της πολυπλοκότητας της διασύνδεσης. Γενικά, η μέθοδος TDM εξοικονομεί ένα επίπεδο πυλών XOR σε σχέση με τα δένδρα [4:2], ή ακόμα περισσότερα για πολύ μεγάλους πολλαπλασιασμούς. Ωστόσο, αυτή η εξοικονόμηση επιτυγχάνεται με αντίτιμο την έλλειψη κανονικότητας της διασύνδεσης· για το λόγο αυτό, τα δένδρα [4:2] και οι διάφορες παραλλαγές τους παραμένουν δημοφιλή.

ΠΙΝΑΚΑΣ 11.15 Σύγκριση επιπέδων XOR σε δένδρα πολλαπλασιαστών

Αρ. μερ. γινομένων	Δένδρο Wallace	Δένδρο 4:2	TDM
8	8	6	5
9	8	8	6
16	12	9	8
24	14	11	10
32	16	12	11
64	20	15	14

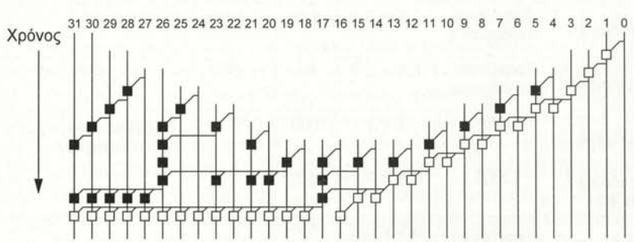
11.9.4.3 Υβριδικός Πολλαπλασιασμός Οι διατάξεις παρέχουν φυσικά σχέδια με κανονικότητα, αλλά πολλά επίπεδα αθροιστών αποθήκευσης κρατουμένου (CSA). Τα δένδρα προσφέρουν λιγότερα επίπεδα CSA, αλλά λιγότερο κανονική φυσική σχεδίαση και μερικούς μεγάλου μήκους αγωγούς διασύνδεσης. Έχουν προταθεί διάφορες υβριδικές σχεδιάσεις, οι οποίες επιτρέπουν συνδυασμούς μεταξύ των δύο ακραίων σχημάτων. Σ' αυτές περιλαμβάνονται περιττές/άρτιες διατάξεις [Henessy90], διατάξεις διατάξεων [Dhanesha95], δομές δένδρου ισορροπημένης καθυστέρησης (balanced delay trees) [Zuras86], και δομές δένδρου ανάποδης σκάλας (overturned-staircase trees) [Mou90]. Μπορούν να επιτύχουν σχεδόν τόσο λίγα επίπεδα λογικής, όσο και τα δένδρα Wallace, ενώ παρέχουν διασύνδεση μεγαλύτερης κανονικότητας (και ταχύτητας).

OPTIONAL

11.9.5 Η Τελική Πρόσθεση

Η έξοδος της διάταξης ή δένδρου μερικών γινομένων είναι ένας αριθμός των $M + N$ bit σε πλεονασματική μορφή αποθήκευσης κρατουμένου. Ένας CPA εκτελεί την τελική πρόσθεση για τη μετατροπή του αποτελέσματος σε μη-πλεονασματική μορφή.

Οι είσοδοι στον CPA έχουν μη-ομοιόμορφους χρόνους άφιξης. Όπως επισημαίνεται στο Σχήμα 11.91, τα μερικά γινόμενα σχηματίζουν ένα παραλληλόγραμμο, με τις μεσαίες στήλες να έχουν περισσότερα μερικά γινόμενα από τις στήλες αριστερά/δεξιά. Συνεπώς, οι μεσαίες στήλες φθάνουν στον CPA αργότερα από τις άλλες. Το γεγονός αυτό μπορεί να αξιοποιηθεί για την απλοποίηση του CPA [Zimmermann96, Oklobdzija96]. Το Σχήμα 11.94 παρουσιάζει ένα παράδειγμα ενός 32-bit δικτύου προθέματος, το οποίο αξιοποιεί τους μη-ομοιόμορφους χρόνους άφιξης σ' έναν πολλαπλασιαστή 16 x 16-bit. Το αρχικό και το τελικό στάδιο για τον bitwise υπολογισμό των σημάτων PG και των αθροισμάτων δεν παρουσιάζονται. Το μονοπάτι από τις αργότερες μεσαίες εισόδους έως την έξοδο απαιτεί μόνο τέσσερα επίπεδα κυττάρων.



ΣΧΗΜΑ 11.94 Δίκτυο προθέματος του αθροιστή CPA με μη-ομοιόμορφους χρόνους άφιξης των εισόδων.

Ο συνολικός αριθμός κυττάρων και η κατανάλωση ενέργειας είναι πολύ μικρότερα συγκριτικά με αυτά ενός συμβατικού Kogge-Stone ή Sklansky CPA.

11.9.6 Συγχωνευμένη Μονάδα Πολλαπλασιασμού-Πρόσθεσης

Πολλοί αλγόριθμοι, ιδιαίτερα στην ψηφιακή επεξεργασία σήματος, απαιτούν τον υπολογισμό $P = X \times Y + Z$. Αν και ο υπολογισμός αυτός μπορεί να γίνεται μ' ένα πολλαπλασιαστή κι έναν αθροιστή, είναι πολύ πιο γρήγορο να χρησιμοποιηθεί μια συγχωνευμένη μονάδα πολλαπλασιασμού-πρόσθεσης, η οποία δεν είναι παρά ένας συμβατικός πολλαπλασιαστής, τροποποιημένος ώστε να δέχεται ένα επιπλέον μερικό γινόμενο Z, το οποίο αθροίζεται όπως και τα άλλα μερικά γινόμενα [Montoye90]. Το επιπλέον μερικό γινόμενο επαυξάνει την καθυστέρηση ενός πολλαπλασιαστή πίνακα μόνο κατά έναν επιπλέον αθροιστή CSA.

11.9.7 Σειριακός Πολλαπλασιασμός

Αυτή η ενότητα περιλαμβάνεται στην ύλη που είναι διαθέσιμη online, μέσω του συνδέσμου «Web Enhanced», στον ιστότοπο `www.cmosvlsi.com`.

11.9.8 Σύνοψη Ενότητας

Τα τρία βήματα του πολλαπλασιασμού είναι η παραγωγή των μερικών γινομένων, η μείωση των μερικών γινομένων και η πρόσθεση με διάδοση κρατουμένου. Ένας απλός πολλαπλασιαστής $M \times N$ παράγει N μερικά γινόμενα χρησιμοποιώντας πύλες AND. Για πολλαπλασιαστές των 16 ή περισσότερων bit, χρησιμοποιείται κατά κανόνα κωδικοποίηση Booth 4ης τάξης, για τη μείωση του αριθμού των μερικών γινομένων σε δύο, πράγμα το οποίο επιφέρει σημαντική εξοικονόμηση σε επιφάνεια και ισχύ. Ορισμένες υλοποιήσεις βρίσκουν την κωδικοποίηση Booth ταχύτερη, ενώ άλλες θεωρούν ότι είναι ελάχιστο το πλεονέκτημα ταχύτητας που προσφέρει. Στη συνέχεια, τα μερικά γινόμενα μειώνονται σ' ένα ζεύγος αριθμών σε πλεονασματική μορφή αποθήκευσης κρατουμένου με τη χρήση μιας διάταξης ή ενός δένδρου από CSA. Τα δένδρα έχουν λιγότερα επίπεδα λογικής, αλλά μεγαλύτερου μήκους και μειωμένης κανονικότητας αγωγούς διασύνδεσης· παρά ταύτα, οι περισσότεροι μεγάλοι πολλαπλασιαστές χρησιμοποιούν δένδρα ή υβριδικές δομές. Οι επιλογείς Booth και οι CSA με τρανζίστορ περάσματος ήταν δημοφιλείς επιλογές κατά τη δεκαετία του '90, αλλά σήμερα, λόγω της κλιμάκωσης της τάσης τροφοδοσίας, οι σχεδιαστές στρέφονται προς στατικές CMOS υλοποιήσεις. Τέλος, ένας CPA μετατρέπει το αποτέλεσμα σε μη-πλεονασματική μορφή. Ο CPA μπορεί να απλοποιηθεί με βάση τους μη-ομοιόμορφους χρόνους άφιξης των bit.

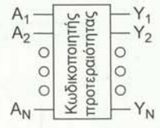
ΠΙΝΑΚΑΣ 11.16 Πολλαπλασιαστές των 54 × 54 bit για αριθμητική κινητής υποδιαστολής, διπλής ακρίβειας

Σχεδίαση	Τεχνολογία κατασκευής (μm)	Μείωση μερ. γινομένων	Κυκλώματα	Επιφάνεια (mm × mm)	Επιφάνεια= ($M\lambda^2$)	Τρανζίστορ	Καθυστέρηση (ns)	Ισχύς (mW)
[Mori91]	0.5	δένδρο 4:2	Τρανζίστορ περάσματος XOR	3.6 × 3.5	200	82k	10	870
[Goto92]	0.8	δένδρο 4:2	Στατικό	3.4 × 3.9	80	83k	13	875
[Heikes94]	0.8	Διάταξη	Domino διπλής γραμμής	2.1 × 2.2	28		20 (διαδ. διοχέτευση 2 σταδίων)	
[Ohkubo95]	0.25	δένδρο 4:2	Τρανζίστορ περάσματος	3.7 × 3.4	805	100k	4.4	
[Goto97]	0.25	δένδρο 4:2	Τρανζίστορ περάσματος	1.0 × 1.3	84	61k	4.1	
[Itoh01]	0.18	δένδρο 4:2	Στατικό	1 × 1	100		3.2 (διαδ. διοχέτευση 2 σταδίων)	
[Belluomini05]	90 nm	δένδρο 3:2 και 4:2	LSDL	0.4 × 0.3	61			1800 @ 8 GHz
[Kuang05]	90 nm	δένδρο 3:2 και 4:2	Τρανζίστορ περάσματος & Domino	0.5 × 0.4	94			426 @ 4 GHz

11.10 Υπολογισμοί Παράλληλου Προθέματος

Πολλές λειτουργίες χειριστών δεδομένων περιλαμβάνουν τον υπολογισμό μιας ομάδας εξόδων από μια ομάδα εισόδων, στην οποία κάθε bit εξόδου εξαρτάται από όλα τα προηγούμενα bit εισόδου. Η πρόσθεση δύο εισόδων των N bit, $A_N \ldots A_1$ και $B_N \ldots B_1$, για την παραγωγή ενός αθροίσματος εξόδου, $Y_N \ldots Y_1$, είναι ένα κλασικό παράδειγμα. Κάθε έξοδος Y_i εξαρτάται από ένα κρατούμενο εισόδου c_{i-1} από το προηγούμενο bit, το οποίο με τη σειρά του εξαρτάται από ένα κρατούμενο εισόδου c_{i-2} από το bit που προηγείται αυτού, κ.ο.κ. Σε πρώτη εξέταση, αυτή η αλυσίδα εξάρτησης δείχνει να υποδηλώνει ότι η καθυστέρηση πρέπει να περιλαμβάνει περίπου N επίπεδα λογικής, όπως σ' έναν αθροιστή κύματος κρατουμένου. Ωστόσο, έχουμε δει ότι με πρόβλεψη κρατουμένου και μπλοκ των οποίων το μέγεθος αυξάνεται προοδευτικά, μπορούμε να κατασκευάσουμε αθροιστές που θα περιλαμβάνουν μόνο $\log N$ επίπεδα. Η Ενότητα 11.2.2.2 εισήγαγε το σκεπτικό της πρόσθεσης ως προθεματικού υπολογισμού, ο οποίος περιλαμβάνει έναν προ-υπολογισμό σε επίπεδο bit, ένα δένδρο λογικής ομάδας για το σχηματισμό των προθεμάτων κι ένα τελικό στάδιο εξόδου, το οποίο παρουσιάζεται στο Σχήμα 11.12. Σ' αυτή την ενότητα θα επεκτείνουμε την ίδια τεχνική και σε άλλους προθεματικούς υπολογισμούς που χρησιμοποιούν συσχετιστικές λειτουργίες σε επίπεδο λογικής ομάδας.

Θα ξεκινήσουμε με τον *κωδικοποιητή προτεραιότητας* (priority encoder) που παρουσιάζεται στο Σχήμα 11.95. Μια συνηθισμένη εφαρμογή ενός κωδικοποιητή προτεραιότητας είναι να εποπτεύει N το πλήθος μονάδες, οι οποίες ζητούν όλες πρόσβαση σ' ένα κοινόχρηστο πόρο. Κάθε μονάδα i στέλνει ένα bit A_i, το οποίο υποδεικνύει μια αίτηση και λαμβάνει ένα bit Y_i, το οποίο υποδεικνύει ότι παραχωρήθηκε πρόσβαση. Η πρόσβαση θα πρέπει να παραχωρείται μόνο σε μια μονάδα, αυτή με την υψηλότερη προτεραιότητα. Εάν το λιγότερο σημαντικό bit της εισόδου αντιστοιχεί στην υψηλότερη προτεραιότητα, η λογική μπορεί να διατυπωθεί ως εξής:



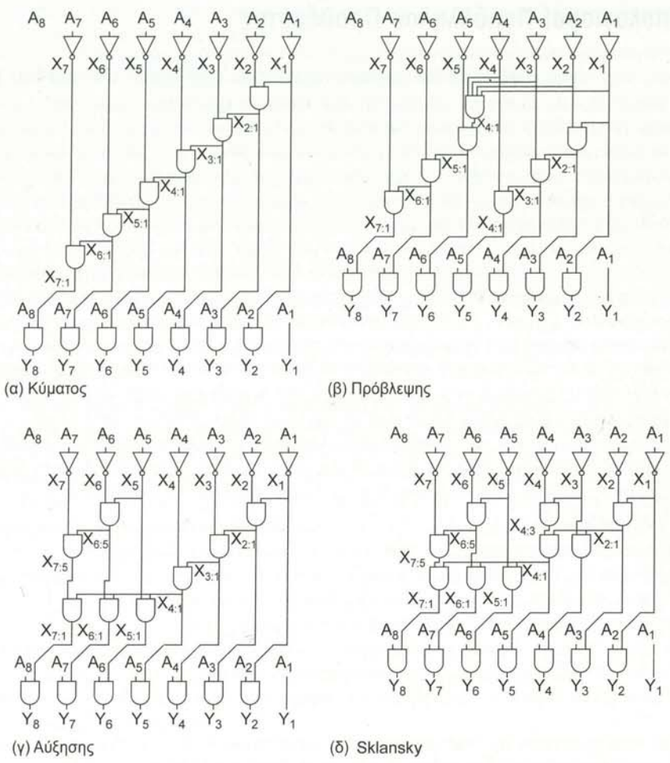
ΣΧΗΜΑ 11.95 Κωδικοποιητής προτεραιότητας.

$$
\begin{aligned}
Y_1 &= A_1 \\
Y_2 &= A_2 \cdot \overline{A_1} \\
Y_3 &= A_3 \cdot \overline{A_2} \cdot \overline{A_1} \\
&\ldots \\
Y_N &= A_N \cdot \overline{A_{N-1}} \cdot \ldots \cdot \overline{A_1}
\end{aligned}
\qquad (11.33)
$$

Μπορούμε να εκφράσουμε την κωδικοποίηση προτεραιότητας ως μια προθεματική λειτουργία, καθορίζοντας ένα πρόθεμα $X_{i:j}$ το οποίο υποδεικνύει ότι καμία από τις εισόδους $A_i \ldots A_j$ δεν έχει τεθεί. Σ' αυτή την περίπτωση, η κωδικοποίηση προτεραιότητας μπορεί να καθοριστεί με προ-υπολογισμό σε επίπεδο bit (bitwise), λογική ομάδας και λογική εξόδου με $i \geq k > j$:

$$
\begin{aligned}
X_{i:i} &= \overline{A}_i && \text{προ-υπολογισμός σε επίπεδο bit} \\
X_{i:j} &= X_{i:k} \cdot X_{k-1:j} && \text{λογική ομάδας} \\
Y_i &= A_i \cdot X_{i-1:1} && \text{λογική εξόδου}
\end{aligned}
\qquad (11.34)
$$

Οποιοδήποτε από τα δίκτυα ομάδας (π.χ., κύματος, παράκαμψης, πρόβλεψης, επιλογής, αύξησης, δένδρου) που εξετάστηκαν στην ενότητα της πρόσθεσης μπορεί να χρησιμοποιηθεί για την κατασκευή της λογικής ομάδας που απαιτείται για τον υπολογισμό του προθέματος $X_{i:0}$. Οι κωδικοποιητές χαμηλής προτεραιότητας χρησιμοποιούν τη δομή κύματος. Οι μεσαίου μήκους κωδικοποιητές μπορούν να χρησιμοποιούν μια δομή παράκαμψης, πρόβλεψης, επιλογής, ή αύξησης. Οι μεγάλου μήκους κωδικοποιητές χρησιμοποιούν δένδρα για την επίτευξη καθυστέρησης της τάξης $\log N$. Το Σχήμα 11.96 παρουσιάζει τέσσερις κωδικοποιητές προτεραιότητας των 8 bit, οι οποίοι απεικονίζουν τις διάφορες λογικές ομάδας. Κάθε σχεδίαση χρησιμοποιεί μια αρχική γραμμή αντιστροφέων για τον προ-υπολογισμό των $X_{i:j}$ και μια τελική γραμμή πυλών AND για τη λογική εξόδου Y_i. Στο ενδιάμεσο, τα δίκτυα κύματος, πρόβλεψης, αύξησης και Sklansky σχηματίζουν τα προθέματα με διάφορους συμβιβασμούς μεταξύ του αριθμού των πυλών και της καθυστέρησης. Συγκρίνετε αυτά τα δένδρα με τα Σχήματα 11.15, 11.22, 11.25 και 11.29(β), αντίστοιχα. Οι [Wang00j, Delgado-Frias00, Huang02] περιγράφουν διάφορες υλοποιήσεις κωδικοποιητών προτεραιότητας.



ΣΧΗΜΑ 11.96 Δένδρα κωδικοποιητών προτεραιότητας

Ένας *προσαυξητής* (incrementer) μπορεί να κατασκευαστεί με παρόμοιο τρόπο. Η πρόσθεση του 1 σε μια λέξη εισόδου συνίσταται στην εύρεση του λιγότερου σημαντικού 0 μέσα στη λέξη και την αντιστροφή όλων των bit έως αυτή τη θέση. Το πρόθεμα X παίζει το ρόλο του σήματος διάδοσης σ' έναν αθροιστή. Και σ' αυτή την περίπτωση μπορούν να χρησιμοποιηθούν οποιαδήποτε από τα δίκτυα προθέματος, με διάφορους συμβιβασμούς μεταξύ επιφάνειας κυκλώματος και ταχύτητας λειτουργίας.

$$\begin{aligned} X_{i:i} &= A_i && \text{προ-υπολογισμός σε επίπεδο bit} \\ X_{i:j} &= X_{i:k} \cdot X_{k-1:j} && \text{λογική ομάδας} \\ Y_i &= A_i \oplus X_{i-1:1} && \text{λογική εξόδου} \end{aligned} \qquad \textbf{(11.35)}$$

Οι *μειωτές* (decrementers) και τα κυκλώματα συμπληρώματος ως προς 2 είναι επίσης παρόμοια [Hashemian92]. Ο μειωτής βρίσκει το λιγότερο σημαντικό 1 και αναστρέφει όλα τα bit έως εκείνη τη θέση. Το κύκλωμα συμπληρώματος ως προς 2 αναστρέφει αριθμητικά έναν προσημασμένο αριθμό, αναστρέφοντας όλα τα bit μεγαλύτερης αξίας πάνω από το λιγότερο σημαντικό 1.

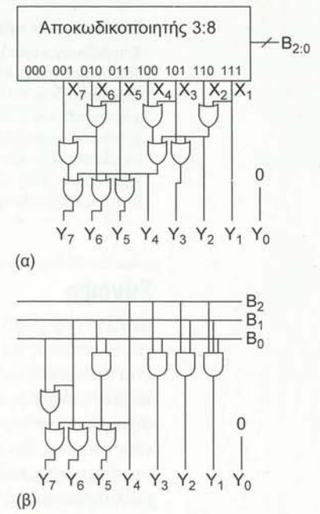
Μια άλλη εφαρμογή προθεματικού υπολογισμού είναι ένας αποκωδικοποιητής από δυαδικό σε κώδικα θερμομέτρου. Η είσοδος B είναι μια k-bit αναπαράσταση του αριθμού M. Η έξοδος Y είναι ένας 2^k-bit αριθμός με τα M περισσότερο σημαντικά bit σε τιμή 1, όπως δίνονται στον Πίνακα 11.17. Μια απλή προσέγγιση είναι να χρησιμοποιηθεί ένας συμβατικός k:2^k αποκωδικοποιητής για την παραγωγή μιας τύπου "one-hot" λέξης A των 2^k bit. Στη συνέχεια, μπορεί να εφαρμοστεί ο ακόλουθος υπολογισμός προθέματος:

$$\begin{aligned} X_{i:i} &= A_{N-i} && \text{προ-υπολογισμός σε επίπεδο bit} \\ X_{i:j} &= X_{i:k} + X_{k-1:j} && \text{λογική ομάδας} \\ Y_i &= X_{i:0} && \text{λογική εξόδου} \end{aligned} \qquad \textbf{(11.36)}$$

Το Σχήμα 11.97(α) παρουσιάζει έναν 8-bit αποκωδικοποιητή από δυαδικό σε κώδικα θερμομέτρου, ο οποίος χρησιμοποιεί ένα δένδρο Sklansky. Ο αποκωδικοποιητής 3:8 περιέχει οκτώ πύλες AND 3 εισόδων, οι οποίες λειτουργούν με τις true και συμπληρωματικές εκδόσεις της εισόδου. Ωστόσο, η λογική μπορεί να απλοποιηθεί σε μεγάλο βαθμό εξαλείφοντας τις συμπληρωματικές εισόδους των AND, όπως παρουσιάζεται στο Σχήμα 11.97(β).

ΠΙΝΑΚΑΣ 11.17 Αποκωδικοποιητής από δυαδικό σε κώδικα θερμομέτρου

B	*Y*
000	00000000
001	10000000
010	11000000
011	11100000
100	11110000
101	11111000
110	11111100
111	11111110



ΣΧΗΜΑ 11.97 Αποκωδικοποιητές από δυαδικό σε κώδικα θερμομέτρου.

Σαν ένα πολυπλοκότερο παράδειγμα, ένας κωδικοποιητής τροποποιημένης προτεραιότητας βρίσκει τα πρώτα δύο 1 σε μια ακολουθία δυαδικών αριθμών. Αυτό θα μπορούσε να είναι χρήσιμο σε μια μνήμη cache με δύο θύρες εγγραφής, στην οποία χρειάζεται να εντοπίζονται οι πρώτες δύο μη-κατειλημμένες θέσεις. Θα χρησιμοποιήσουμε δύο προθέματα: X και W. Κι εδώ, το $X_{i:j}$ υποδεικνύει ότι καμία από τις εισόδους A_i και A_j δεν έχει τεθεί. Το $W_{i:j}$ υποδεικνύει ότι ακριβώς μια από τις εισόδους $A_i \ldots A_j$ έχει τεθεί. Θα παραχθούν δύο τύπου 1-hot έξοδοι, Y και Z, που υποδεικνύουν τα πρώτα δύο ψηφία 1:

$$\begin{aligned} X_{i:i} &= \overline{A_i} \\ W_{i:i} &= A_i && \text{προ-υπολογισμός σε επίπεδο bit} \\ \\ X_{i:j} &= X_{i:k} \cdot X_{k-1:j} \\ W_{i:j} &= W_{i:k} \cdot X_{k-1:j} + X_{i:k} \cdot W_{k-1:j} && \text{λογική ομάδας} \\ \\ Y_i &= A_i \cdot X_{i-1:1} \\ Z_i &= A_i \cdot W_{i-1:1} && \text{λογική εξόδου} \end{aligned} \qquad \textbf{(11.37)}$$

11.11 Κίνδυνοι και Πλάνες

Εξίσωση των επιπέδων λογικής με την καθυστέρηση

Το να συγκρίνει κανείς μια καινοτόμα σχεδίαση με την καλύτερη υπάρχουσα είναι δύσκολο. Ορισμένοι μηχανικοί απλοποιούν το ζήτημα, συγκρίνοντας μόνο τα επίπεδα λογικής. Δυστυχώς, η καθυστέρηση εξαρτάται σε μεγάλο βαθμό από το λογικό φόρτο κάθε σταδίου, το βαθμό οδήγησης εξόδου και τη χωρητικότητα διασύνδεσης. Για παράδειγμα, ο [Srinivas92] υποστηρίζει ότι ένας νέος αθροιστής είναι κατά 20%-28% ταχύτερος από

τον ταχύτερο γνωστό δυαδικό αθροιστή πρόβλεψης κρατουμένου, αλλά δεν αναφέρει αποτελέσματα προσομοίωσης. Επιπλέον, αναφέρει κάποια από τα πλεονεκτήματα του αθροιστή όσον αφορά στην ταχύτητα, σε τρία ή τέσσερα σημαντικά σχήματα. Σε προσεκτικότερη εξέταση [Dobson95], ο αθροιστής αυτός αποδεικνύεται ότι είναι μια υβριδική σχεδίαση δομής δένδρου/επιλογής κρατουμένου, με κάποιο προ-υπολογισμό ο οποίος δεν είναι αναγκαίος.

Σχεδίαση κυκλωμάτων με πτώσεις τάσης κατωφλίου

Στις σύγχρονες μεθόδους σχεδίασης, τα τρανζίστορ ενός περάσματος, τα οποία οδηγούν μια έξοδο σε τάση V_{DD}-V_t, γενικά θεωρούνται μη αποδεκτά, επειδή η πτώση τάσης κατωφλίου (η οποία ενισχύεται από το φαινόμενο σώματος) δίνει ως αποτέλεσμα μια έξοδο με εξαιρετικά μικρό περιθώριο θορύβου. Επιπλέον, όταν οδηγούν τους ακροδέκτες πύλης ενός επόμενου σταδίου, το στάδιο αυτό μεταβαίνει μερικώς σε κατάσταση ON και καταναλώνει στατική ισχύ. Έχουν προταθεί πολλά κύτταρα πλήρων αθροιστών των 10 τρανζίστορ, τα οποία υποφέρουν από αυτό το πρόβλημα.

Επινόηση νέων αθροιστών, εκ του μηδενός

Στη βιβλιογραφία υπάρχει τεράστιος όγκος υλικού για τους αθροιστές, με διάφορες επιλογές/συμβιβασμούς μεταξύ ταχύτητας, επιφάνειας και κατανάλωσης ισχύος. Ο σχεδιαστικός χώρος των αθροιστών έχει διερευνηθεί διεξοδικά, και πολλοί σχεδιαστές (συμπεριλαμβανομένου ενός εκ των συγγραφέων) έχουν αναλώσει αρκετό χρόνο για την ανάπτυξη ενός νέου αθροιστή, μόνο και μόνο για να διαπιστώσουν τελικά ότι δεν είναι παρά μόνο μια «παραλλαγή στο ίδιο θέμα» – μια ήσσονος σημασίας διαφοροποίηση σε σχέση με μια υφιστάμενη σχεδίαση. Στο ίδιο μήκος κύματος, υπάρχουν ορισμένες πρόσφατες δημοσιεύσεις πάνω στους κωδικοποιητές προτεραιότητας, οι οποίες επινοούν εκ νέου τεχνικές δικτύων προθέματος που έχουν ερευνηθεί ήδη στα πλαίσια της πρόσθεσης.

Σύνοψη

Αυτό το κεφάλαιο παρουσίασε διάφορα υποσυστήματα χειριστών δεδομένων. Ο τρόπος με τον οποίο επιχειρεί κάποιος να σχεδιάσει και να υλοποιήσει ένα δεδομένο ολοκληρωμένο CMOS επηρεάζεται άμεσα από τη διαθεσιμότητα των εργαλείων, το χρονοδιάγραμμα, την πολυπλοκότητα του συστήματος και τους τελικούς στόχους ως προς το κόστους της σχεδίασης. Σε γενικό επίπεδο, θα πρέπει να επιλέγεται η απλούστερη και λιγότερο ακριβή (όσον αφορά στο χρόνο και στο κόστος) προσέγγιση, η οποία επιτυγχάνει τους στόχους. Για πολλά συστήματα, αυτό σημαίνει ότι επαρκούν οι διαδικασίες σύνθεσης (synthesis) και τοποθέτησης και διασύνδεσης (place & route). Τα σύγχρονα εργαλεία σύνθεσης βασίζονται σε μια καλή βιβλιοθήκη από αθροιστές και πολλαπλασιαστές με διάφορες επιλογές επιφάνειας/ταχύτητας, που επαρκούν για την κάλυψη ενός μεγάλου εύρους εφαρμογών. Για συστήματα τα οποία πρέπει να ικανοποιούν τις πλέον αυστηρές απαιτήσεις όσον αφορά την απόδοση ή την πυκνότητα σχεδίασης, η «επί τούτου» (custom) σχεδίαση σε σχηματικό επίπεδο συνεχίζει να προσφέρει ένα πλεονέκτημα. Οι δομές δένδρου domino παράλληλου προθέματος παρέχουν τους ταχύτερους αθροιστές, όταν μπορεί να γίνει ανεκτή η υψηλή κατανάλωση ισχύος τους. Οι domino αθροιστές αποθήκευσης κρατουμένου (CSA) χρησιμοποιούνται επίσης σε γρήγορους πολλαπλασιαστές. Ωστόσο, στη σχεδίαση πολλαπλασιαστών η χωρητικότητα διασύνδεσης είναι μείζονος σημασίας, πράγμα το οποίο σημαίνει ότι ένας πολλαπλασιαστής με συμπαγή κύτταρα και μικρού μήκους αγωγούς διασύνδεσης μπορεί να είναι αφενός γρήγορος και αφετέρου χαμηλής κατανάλωσης επιφάνειας και ισχύος.

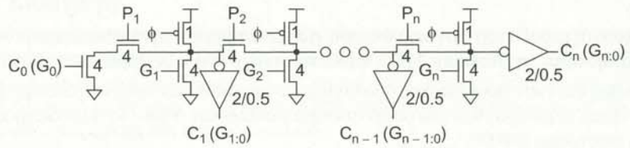
Ασκήσεις

11.1 Σχεδιάστε ένα γρήγορο αθροιστή των 8 bit. Οι είσοδοι μπορούν να οδηγούν όχι περισσότερο από 30λ πλάτους τρανζίστορ έκαστη και η έξοδος πρέπει να οδηγεί έναν 20/10 αντιστροφέα. Προσομοιώστε τον αθροιστή και καθορίστε την καθυστέρησή του.

11.2 Όταν προστίθενται δύο μη-προσημασμένοι αριθμοί, ένα κρατούμενο εξόδου του τελικού σταδίου υποδεικνύει εάν έχει συμβεί υπερχείλιση. Όταν προστίθενται δύο προσημασμένοι αριθμοί σε μορφή συμπληρώματος ως προς 2, η ανίχνευση υπερχείλισης είναι λίγο πιο πολύπλοκη. Αναπτύξτε μια Boolean έκφραση για την υπερχείλιση ως συνάρτηση του κρατουμένου εξόδου και των πιο σημαντικών bit της εξόδου.

11.3 Επαναλάβετε την Άσκηση 11.2 για μια μονάδα πρόσθεσης/αφαίρεσης προσημασμένων αριθμών όπου αυτή του Σχήματος 11.41(β). Η δική έξοδος υπερχείλισης θα πρέπει να είναι συνάρτηση του υπο-σήματος και των πιο σημαντικών bit των δύο εισόδων και της εξόδου.

11.4 Διατυπώστε εξισώσεις για τον υπολογισμό του λογικού φόρτου και της παρασιτικής καθυστέρησης αναφορικά με την είσοδο C_0 μιας αλυσίδας κρατουμένου Manchester n επιπέδων, η οποία υπολογίζει τα $C_1 \ldots C_n$. Κατά την εξαγωγή της παρασιτικής καθυστέρησης, θα πρέπει να συνυπολογίσετε όλες τις εσωτερικές χωρητικότητες διάχυσης. Χρησιμοποιήστε τα πλάτη των τρανζίστορ που αναγράφονται στο Σχήμα 11.98 και υποθέστε ότι τα τρανζίστορ P_i και G_i κάθε σταδίου μοιράζονται την ίδια επαφή διάχυσης.



ΣΧΗΜΑ 11.98 Αλυσίδα κρατουμένου Manchester.

11.5 Χρησιμοποιώντας τα αποτελέσματα της Άσκησης 11.4, ποια αλυσίδα κρατουμένου Manchester δίνει τη μικρότερη καθυστέρηση για έναν αθροιστή μεγάλου μήκους;

11.6 Ο αθροιστής πρόβλεψης κρατουμένου του Σχ. 11.26(β) με μεταβλητό μέγεθος μπλοκ απαιτεί πέντε στάδια κυττάρων ομάδας PG 2ης τάξης για την πρόσθεση 16 bit. Πόσα στάδια απαιτούνται για πρόσθεση 32 bit; Για πρόσθεση 64 bit;

11.7 Σχεδιάστε το δίκτυο PG για έναν τροποποιημένο αθροιστή Sklansky των 16 bit με βαθμό οδήγησης [8,1,1,1] αντί για [8,4,2,1]. Χρησιμοποιήστε απομονωτές (buffers) για να εμποδίσετε τα λιγότερο σημαντικά ψηφία να επιβαρύνουν το κρίσιμο μονοπάτι.

11.8 Το Σχήμα 11.29 παρουσιάζει δίκτυα PG για διάφορους αθροιστές των 16 bit και το Σχήμα 11.30 απεικονίζει πώς αυτά τα δίκτυα μπορούν να κατηγοριοποιηθούν ως τομή της επίπεδης επιφάνειας l+f+t = 3 με την έδρα ενός κύβου. Η επίπεδη επιφάνεια διατέμνει επίσης ένα σημείο στο εσωτερικό του κύβου, στο (l,f,t) = (1,1,1). Σχεδιάστε το δίκτυο PG γι' αυτό τον αθροιστή των 16 bit.

11.9 Σχεδιάστε ένα διάγραμμα του δένδρου PG ομάδας για έναν αθροιστή Ladner-Fischer των 32 bit.

11.10 Διατυπώστε την Boolean έκφραση για το C_{out} στο κύκλωμα του Σχήματος 11.6(β). Απλοποιήστε την έκφραση για να αποδείξετε ότι τα κυκλώματα με τρανζίστορ περάσματος υπολογίζουν πράγματι τη συνάρτηση πλειοψηφίας.

11.11 Αποδείξτε την Εξ. (11.21).

11.12 Σχεδιάστε ένα συγκριτή, ο οποίος θα υπολογίζει το A-B = k.

11.13 Δείξτε πώς το φυσικό σχέδιο της γεννήτριας ισοτιμίας του Σχήματος 11.57 μπορεί να σχεδιαστεί σαν μια γραμμική στήλη πυλών XOR μ' ένα κανάλι διασύνδεσης δομής δένδρου.

11.14 Σχεδιάστε έναν αποκωδικοποιητή ECC για κώδικες Hamming απόστασης 3 με c = 3. Το κύκλωμά σας θα πρέπει να δέχεται μια λέξη των 7 bit και να παράγει μια διορθωμένη λέξη των 4 bit. Σχεδιάστε μια υλοποίηση σε επίπεδο πυλών.

11.15 Πόσα bit ελέγχου απαιτούνται σ' έναν κώδικα Hamming απόστασης 3 για λέξεις δεδομένων των 8 bit; Σχεδιάστε έναν πίνακα ελέγχου ισοτιμίας και γράψτε τις εξισώσεις για τον υπολογισμό του κάθε bit ελέγχου.

τον ταχύτερο γνωστό δυαδικό αθροιστή πρόβλεψης κρατουμένου, αλλά δεν αναφέρει αποτελέσματα προσομοίωσης. Επιπλέον, αναφέρει κάποια από τα πλεονεκτήματα του αθροιστή όσον αφορά στην ταχύτητα, σε τρία ή τέσσερα σημαντικά σχήματα. Σε προσεκτικότερη εξέταση [Dobson95], ο αθροιστής αυτός αποδεικνύεται ότι είναι μια υβριδική σχεδίαση δομής δένδρου/επιλογής κρατουμένου, με κάποιο προ-υπολογισμό ο οποίος δεν είναι αναγκαίος.

Σχεδίαση κυκλωμάτων με πτώσεις τάσης κατωφλίου

Στις σύγχρονες μεθόδους σχεδίασης, τα τρανζίστορ ενός περάσματος, τα οποία οδηγούν μια έξοδο σε τάση V_{DD}-V_t, γενικά θεωρούνται μη αποδεκτά, επειδή η πτώση τάσης κατωφλίου (η οποία ενισχύεται από το φαινόμενο σώματος) δίνει ως αποτέλεσμα μια έξοδο με εξαιρετικά μικρό περιθώριο θορύβου. Επιπλέον, όταν οδηγούν τους ακροδέκτες πύλης ενός επόμενου σταδίου, το στάδιο αυτό μεταβαίνει μερικώς σε κατάσταση ON και καταναλώνει στατική ισχύ. Έχουν προταθεί πολλά κύτταρα πλήρων αθροιστών των 10 τρανζίστορ, τα οποία υποφέρουν από αυτό το πρόβλημα.

Επινόηση νέων αθροιστών, εκ του μηδενός

Στη βιβλιογραφία υπάρχει τεράστιος όγκος υλικού για τους αθροιστές, με διάφορες επιλογές/συμβιβασμούς μεταξύ ταχύτητας, επιφάνειας και κατανάλωσης ισχύος. Ο σχεδιαστικός χώρος των αθροιστών έχει διερευνηθεί διεξοδικά, και πολλοί σχεδιαστές (συμπεριλαμβανομένου ενός εκ των συγγραφέων) έχουν αναλώσει αρκετό χρόνο για την ανάπτυξη ενός νέου αθροιστή, μόνο και μόνο για να διαπιστώσουν τελικά ότι δεν είναι παρά μόνο μια «παραλλαγή στο ίδιο θέμα» – μια ήσσονος σημασίας διαφοροποίηση σε σχέση με μια υφιστάμενη σχεδίαση. Στο ίδιο μήκος κύματος, υπάρχουν ορισμένες πρόσφατες δημοσιεύσεις πάνω στους κωδικοποιητές προτεραιότητας, οι οποίες επινοούν εκ νέου τεχνικές δικτύων προθέματος που έχουν ερευνηθεί ήδη στα πλαίσια της πρόσθεσης.

Σύνοψη

Αυτό το κεφάλαιο παρουσίασε διάφορα υποσυστήματα χειριστών δεδομένων. Ο τρόπος με τον οποίο επιχειρεί κάποιος να σχεδιάσει και να υλοποιήσει ένα δεδομένο ολοκληρωμένο CMOS επηρεάζεται άμεσα από τη διαθεσιμότητα των εργαλείων, το χρονοδιάγραμμα, την πολυπλοκότητα του συστήματος και τους τελικούς στόχους ως προς το κόστους της σχεδίασης. Σε γενικό επίπεδο, θα πρέπει να επιλέγεται η απλούστερη και λιγότερο ακριβή (όσον αφορά στο χρόνο και στο κόστος) προσέγγιση, η οποία επιτυγχάνει τους στόχους. Για πολλά συστήματα, αυτό σημαίνει ότι επαρκούν οι διαδικασίες σύνθεσης (synthesis) και τοποθέτησης και διασύνδεσης (place & route). Τα σύγχρονα εργαλεία σύνθεσης βασίζονται σε μια καλή βιβλιοθήκη από αθροιστές και πολλαπλασιαστές με διάφορες επιλογές επιφάνειας/ταχύτητας, που επαρκούν για την κάλυψη ενός μεγάλου εύρους εφαρμογών. Για συστήματα τα οποία πρέπει να ικανοποιούν τις πλέον αυστηρές απαιτήσεις όσον αφορά την απόδοση ή την πυκνότητα σχεδίασης, η «επί τούτου» (custom) σχεδίαση σε σχηματικό επίπεδο συνεχίζει να προσφέρει ένα πλεονέκτημα. Οι δομές δένδρου domino παράλληλου προθέματος παρέχουν τους ταχύτερους αθροιστές, όταν μπορεί να γίνει ανεκτή η υψηλή κατανάλωση ισχύος τους. Οι domino αθροιστές αποθήκευσης κρατουμένου (CSA) χρησιμοποιούνται επίσης σε γρήγορους πολλαπλασιαστές. Ωστόσο, στη σχεδίαση πολλαπλασιαστών η χωρητικότητα διασύνδεσης είναι μείζονος σημασίας, πράγμα το οποίο σημαίνει ότι ένας πολλαπλασιαστής με συμπαγή κύτταρα και μικρού μήκους αγωγούς διασύνδεσης μπορεί να είναι αφενός γρήγορος και αφετέρου χαμηλής κατανάλωσης επιφάνειας και ισχύος.

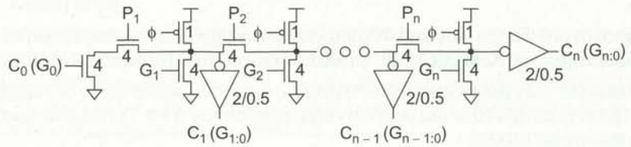
Ασκήσεις

11.1 Σχεδιάστε ένα γρήγορο αθροιστή των 8 bit. Οι είσοδοι μπορούν να οδηγούν όχι περισσότερο από 30λ πλάτους τρανζίστορ έκαστη και η έξοδος πρέπει να οδηγεί έναν 20/10 αντιστροφέα. Προσομοιώστε τον αθροιστή και καθορίστε την καθυστέρησή του.

11.2 Όταν προστίθενται δύο μη-προσημασμένοι αριθμοί, ένα κρατούμενο εξόδου του τελικού σταδίου υποδεικνύει εάν έχει συμβεί υπερχείλιση. Όταν προστίθενται δύο προσημασμένοι αριθμοί σε μορφή συμπληρώματος ως προς 2, η ανίχνευση υπερχείλισης είναι λίγο πιο πολύπλοκη. Αναπτύξτε μια Boolean έκφραση για την υπερχείλιση ως συνάρτηση του κρατουμένου εξόδου και των πιο σημαντικών bit της εξόδου.

11.3 Επαναλάβετε την Άσκηση 11.2 για μια μονάδα πρόσθεσης/αφαίρεσης προσημασμένων αριθμών όπου αυτή του Σχήματος 11.41(β). Η δική έξοδος υπερχείλισης θα πρέπει να είναι συνάρτηση του υπο-σήματος και των πιο σημαντικών bit των δύο εισόδων και της εξόδου.

11.4 Διατυπώστε εξισώσεις για τον υπολογισμό του λογικού φόρτου και της παρασιτικής καθυστέρησης αναφορικά με την είσοδο C_0 μιας αλυσίδας κρατουμένου Manchester n επιπέδων, η οποία υπολογίζει τα $C_1 \ldots C_n$. Κατά την εξαγωγή της παρασιτικής καθυστέρησης, θα πρέπει να συνυπολογίσετε όλες τις εσωτερικές χωρητικότητες διάχυσης. Χρησιμοποιήστε τα πλάτη των τρανζίστορ που αναγράφονται στο Σχήμα 11.98 και υποθέστε ότι τα τρανζίστορ P_i και G_i κάθε σταδίου μοιράζονται την ίδια επαφή διάχυσης.



ΣΧΗΜΑ 11.98 Αλυσίδα κρατουμένου Manchester.

11.5 Χρησιμοποιώντας τα αποτελέσματα της Άσκησης 11.4, ποια αλυσίδα κρατουμένου Manchester δίνει τη μικρότερη καθυστέρηση για έναν αθροιστή μεγάλου μήκους;

11.6 Ο αθροιστής πρόβλεψης κρατουμένου του Σχ. 11.26(β) με μεταβλητό μέγεθος μπλοκ απαιτεί πέντε στάδια κυττάρων ομάδας PG 2ης τάξης για την πρόσθεση 16 bit. Πόσα στάδια απαιτούνται για πρόσθεση 32 bit; Για πρόσθεση 64 bit;

11.7 Σχεδιάστε το δίκτυο PG για έναν τροποποιημένο αθροιστή Sklansky των 16 bit με βαθμό οδήγησης [8,1,1,1] αντί για [8,4,2,1]. Χρησιμοποιήστε απομονωτές (buffers) για να εμποδίσετε τα λιγότερο σημαντικά ψηφία να επιβαρύνουν το κρίσιμο μονοπάτι.

11.8 Το Σχήμα 11.29 παρουσιάζει δίκτυα PG για διάφορους αθροιστές των 16 bit και το Σχήμα 11.30 απεικονίζει πώς αυτά τα δίκτυα μπορούν να κατηγοριοποιηθούν ως τομή της επίπεδης επιφάνειας l+f+t = 3 με την έδρα ενός κύβου. Η επίπεδη επιφάνεια διατέμνει επίσης ένα σημείο στο εσωτερικό του κύβου, στο (l_i,f,t) = (1,1,1). Σχεδιάστε το δίκτυο PG γι' αυτό τον αθροιστή των 16 bit.

11.9 Σχεδιάστε ένα διάγραμμα του δένδρου PG ομάδας για έναν αθροιστή Ladner-Fischer των 32 bit.

11.10 Διατυπώστε την Boolean έκφραση για το C_{out} στο κύκλωμα του Σχήματος 11.6(β). Απλοποιήστε την έκφραση για να αποδείξετε ότι τα κυκλώματα με τρανζίστορ περάσματος υπολογίζουν πράγματι τη συνάρτηση πλειοψηφίας.

11.11 Αποδείξτε την Εξ. (11.21).

11.12 Σχεδιάστε ένα συγκριτή, ο οποίος θα υπολογίζει το A-B = k.

11.13 Δείξτε πώς το φυσικό σχέδιο της γεννήτριας ισοτιμίας του Σχήματος 11.57 μπορεί να σχεδιαστεί σαν μια γραμμική στήλη πυλών XOR μ' ένα κανάλι διασύνδεσης δομής δένδρου.

11.14 Σχεδιάστε έναν αποκωδικοποιητή ECC για κώδικες Hamming απόστασης 3 με c = 3. Το κύκλωμά σας θα πρέπει να δέχεται μια λέξη των 7 bit και να παράγει μια διορθωμένη λέξη των 4 bit. Σχεδιάστε μια υλοποίηση σε επίπεδο πυλών.

11.15 Πόσα bit ελέγχου απαιτούνται σ' έναν κώδικα Hamming απόστασης 3 για λέξεις δεδομένων των 8 bit; Σχεδιάστε έναν πίνακα ελέγχου ισοτιμίας και γράψτε τις εξισώσεις για τον υπολογισμό του κάθε bit ελέγχου.

11.16 Βρείτε τις τιμές του δυαδικά-κατοπτρικού κώδικα Gray των 4 bit για τους αριθμούς 0-15.

11.17 Σχεδιάστε ένα μετρητή με κωδικοποίηση Gray, στον οποίο μόνο ένα bit θα αλλάζει σε κάθε κύκλο.

11.18 Στον Πίνακα 11.12 και το Σχήμα 11.80 παρουσιάστηκε η κωδικοποίηση Booth 4ης τάξης χρησιμοποιώντας τα *SINGLE*, *DOUBLE* και *NEG*. Μια εναλλακτική κωδικοποίηση είναι να χρησιμοποιήσουμε τα *POS*, *NEG* και *DOUBLE*. Το *POS* είναι true για τα πολλαπλάσια *Y* και *2Y*. Το *NEG* είναι true για τα πολλαπλάσια *-Y* και *-2Y*. Το *DOUBLE* είναι true για τα πολλαπλάσια *2Y* και *-2Y*. Σχεδιάστε έναν κωδικοποιητή και επιλογέα Booth χρησιμοποιώντας αυτό το σχήμα κωδικοποίησης.

11.19 Προσαρμόστε τη λογική του κωδικοποιητή προτεραιότητας της Εξ. (11.37) ώστε να παράγονται τρεις έξοδοι «επιλογής ενός ψηφίου» (τύπου «one-hot»), οι οποίες θα αντιστοιχούν στα πρώτα τρία 1 μιας συμβολοσειράς εισόδου.

11.20 Σχεδιάστε έναν κωδικοποιητή προτεραιότητας των 16 bit, χρησιμοποιώντας ένα δίκτυο προθέματος Kogge-Stone.

11.21 Χρησιμοποιήστε τη μέθοδο του Λογικού Φόρτου για να εκτιμήσετε την καθυστέρηση του κωδικοποιητή προτεραιότητας της Άσκησης 11.20. Υποθέστε ότι ο ηλεκτρικός φόρτος μονοπατιού είναι 1.

11.22 Γράψτε εξισώσεις για έναν προθεματικό υπολογισμό, ο οποίος θα καθορίζει τη δεύτερη θέση στην οποία εμφανίζεται το μοτίβο 10 σε μια συμβολοσειρά εισόδου των *N* bit. Για παράδειγμα, η 010010 θα πρέπει να επιστρέφει 010000.

11.23 Ο [Jackson04] προτείνει μια επέκταση του αθροιστή Ling για την απλοποίηση των κυττάρων που έπονται στο δίκτυο προθέματος. Σχεδιάστε έναν 16-bit αθροιστή χρησιμοποιώντας αυτή την τεχνική και συγκρίνετέ τον μ' ένα συμβατικό, 16-bit αθροιστή Ling.

12 Υποσυστήματα Διατάξεων

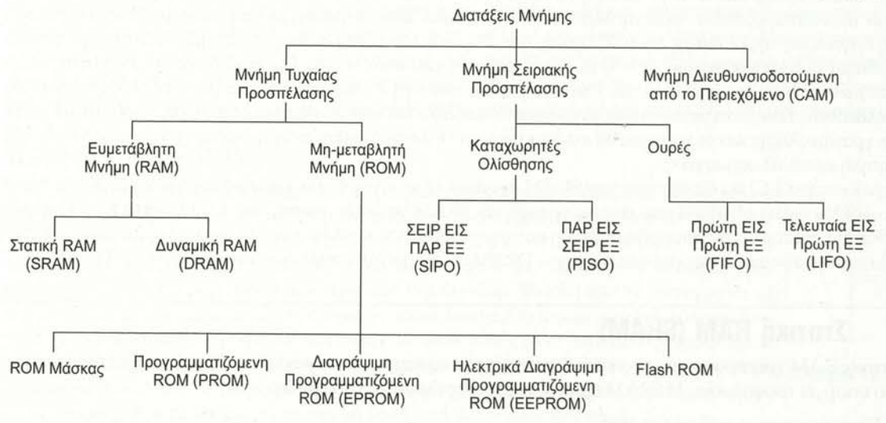
12.1 Εισαγωγή

Συχνά, οι διατάξεις μνήμης καταλαμβάνουν την πλειονότητα των τρανζίστορ σ' ένα σύστημα σε ψηφίδα (system-on-chip) τεχνολογίας CMOS. Οι διατάξεις μνήμης μπορούν να ταξινομηθούν σε κατηγορίες, οι οποίες παρουσιάζονται στο Σχήμα 12.1. Οι *προγραμματιζόμενες διατάξεις λογικής* (Programmable Logic Arrays, PLA), παρά το γεγονός ότι εκτελούν κατά βάση λογικές λειτουργίες και όχι λειτουργίες αποθήκευσης, εξετάζονται επίσης στο παρόν κεφάλαιο.

Η *μνήμη τυχαίας προσπέλασης* (random access memory, RAM) προσπελάζεται μέσω μιας *διεύθυνσης* και έχει καθυστέρηση ανεξάρτητη της διεύθυνσης. Εν αντιθέσει, οι *μνήμες σειριακής προσπέλασης* (serial access memories) προσπελάζονται ακολουθιακά, οπότε δεν είναι αναγκαία η χρήση διεύθυνσης. Οι *διευθυνσιοδοτούμενες από το περιεχόμενο μνήμες* (content addressable memories) εξακριβώνουν ποια ή ποιες διευθύνσεις περιέχουν τα δεδομένα που ταιριάζουν με ένα καθοριζόμενο *κλειδί* (key).

Η μνήμη τυχαίας προσπέλασης κατηγοριοποιείται συνήθως σε δύο τύπους: *μνήμη μόνο ανάγνωσης* (ROM) και *μνήμη ανάγνωσης/εγγραφής*, η οποία έχει καθιερωθεί να αποκαλείται RAM, γεγονός το οποίο μπορεί να προκαλέσει σύγχυση. Ακόμα και ο όρος ROM δεν είναι απολύτως ακριβής, επειδή πολλές ROM έχουν δυνατότητα εγγραφής. Ένα πιο χρήσιμο σχήμα κατηγοριοποίησης θα ήταν ως *ευμετάβλητες* (volatile, μη-διατηρητικές) και *μη-μεταβλητές* (nonvolatile, διατηρητικές) μνήμες. Η ευμετάβλητη μνήμη διατηρεί τα δεδομένα της μόνο για όσο χρόνο τροφοδοτείται με ρεύμα, ενώ η μη-μεταβλητή μνήμη κρατάει τα δεδομένα επ' άπειρον. Ο όρος RAM κατέληξε να είναι συνώνυμος με την ευμετάβλητη μνήμη, ενώ ο όρος ROM με τη μη-μεταβλητή μνήμη

Όμοια με τα ακολουθιακά στοιχεία, τα κύτταρα μνήμης που χρησιμοποιούνται σε ευμετάβλητες μνήμες μπορούν να κατηγοριοποιηθούν περαιτέρω σε *στατικές* και *δυναμικές* δομές. Τα στατικά κύτταρα χρησιμοποιούν μια μορφή ανάδρασης για τη διατήρηση της κατάστασής τους, ενώ τα δυναμικά κύτταρα χρη-



ΣΧΗΜΑ 12.1 Κατηγορίες διατάξεων μνήμης.