

Σχεδιασμός Ολοκληρωμένων Κυκλωμάτων VLSI – II

1

1



Tree Adders

- Για τους αθροιστές μεγάλου μήκους ($N > 16$ bits), η καθυστέρηση εξαρτάται από την καθυστέρηση του διάδοσης του κρατούμενου διαμέσου των σταδίων πρόβλεψης
- Μπορεί να κατασκευαστεί ένα πολυεπίπεδο δένδρο δομών πρόβλεψης => η καθυστέρηση διάδοσης του κρατούμενου να είναι $\log N$
- Τέτοιοι αθροιστές αναφέρονται συνήθως ως αθροιστές δένδρου
- Υπάρχουν πολλοί τρόποι κατασκευής του δένδρου πρόβλεψης με συμβιβασμούς ανάμεσα στο
 - πλήθος των επιπέδων λογικής
 - πλήθος των λογικών πυλών
 - μέγιστο βαθμό οδήγησης εξόδου κάθε πύλης και
 - ποσότητα καλωδίωσης μεταξύ των σταδίων
- Τρία θεμελιώδη δένδρα είναι οι αρχιτεκτονικές: Brent-Kung, Sklansky και Kogge-Stone

2

2

Βασικά δομικά στοιχεία

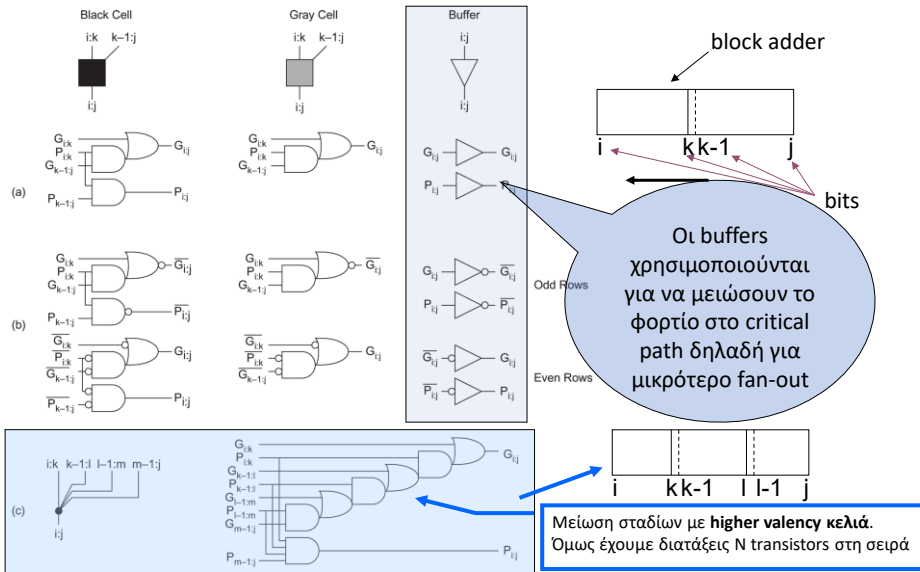
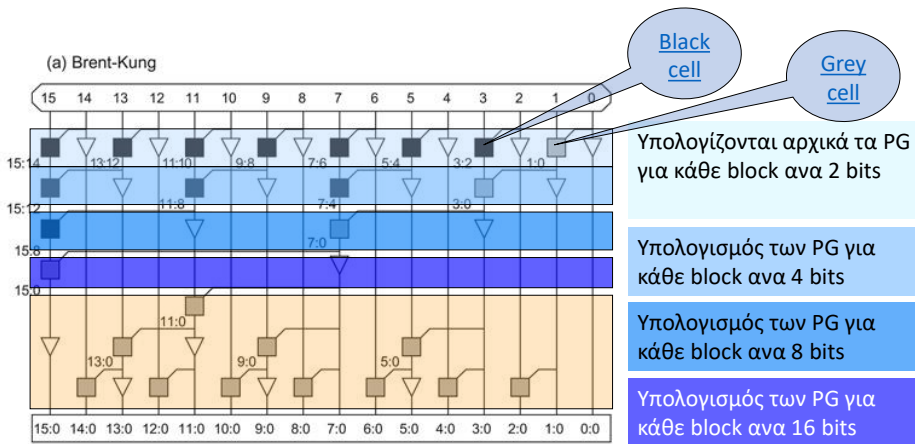


FIG 10.17 Group PG cells

3

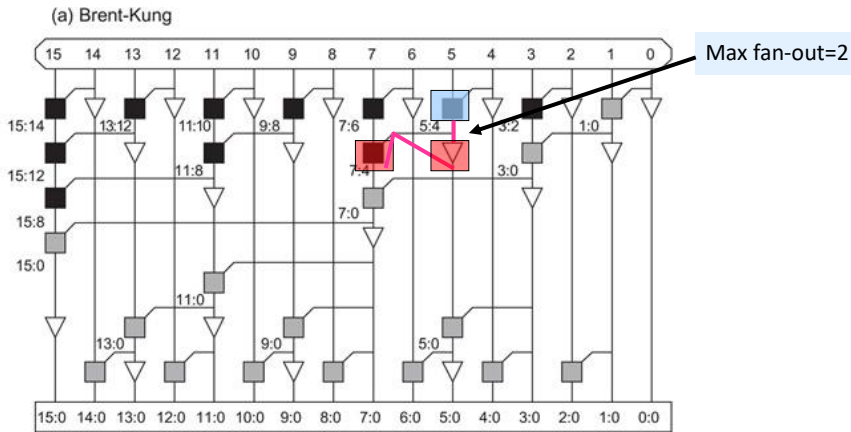
Brent Kung (1/3)



Υπολογισμός σε μορφή δέντρου των απαραίτητων G των ενδιάμεσων bit για το τελικό αποτέλεσμα

4

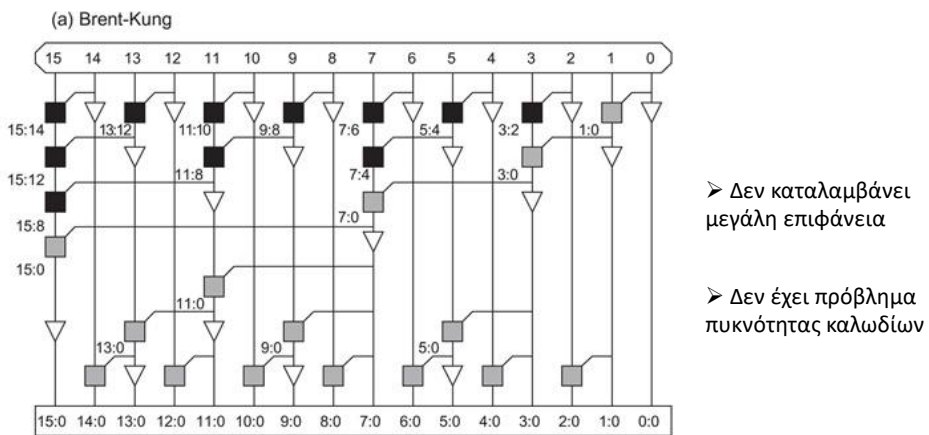
Brent Kung (2/3)



VLSI – II

5

Brent Kung (3/3)

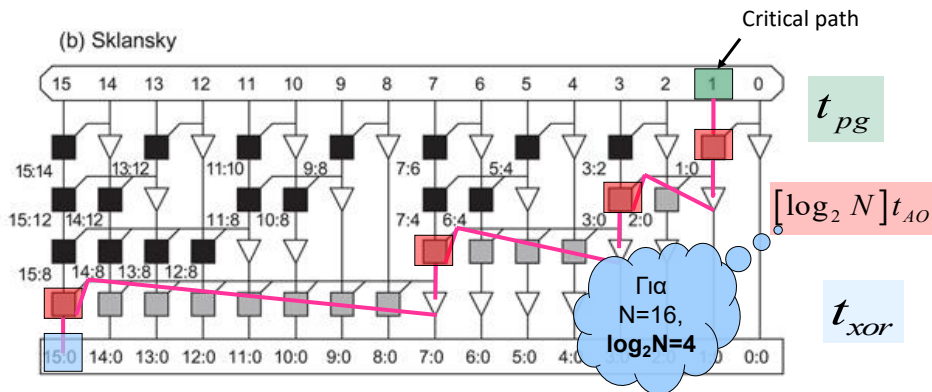


➤ Απαιτεί $2\log_2 N - 1$ επίπεδα για τον τελικό υπολογισμό – δεν είναι η καλύτερη επιλογή για την καθυστέρηση

VLSI – II

6

Sklansky (3/4)



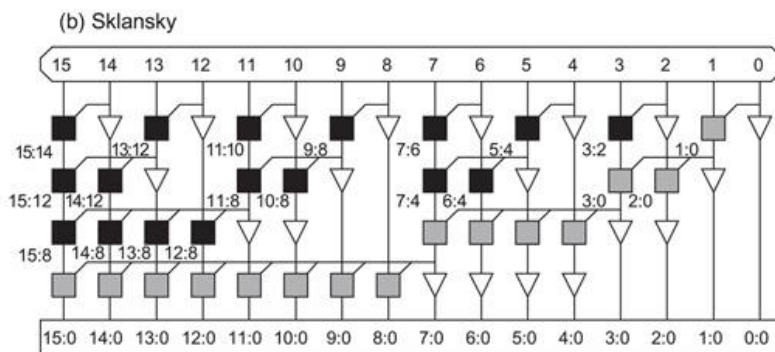
Critical path delay του Sklansky (ισχύει και για [Kogge-Stone tree adder](#)):

$$t_{tree} = t_{pg} + [\log_2 N] t_{AO} + t_{xor}$$

VLSI – II

9

Sklansky (4/4)

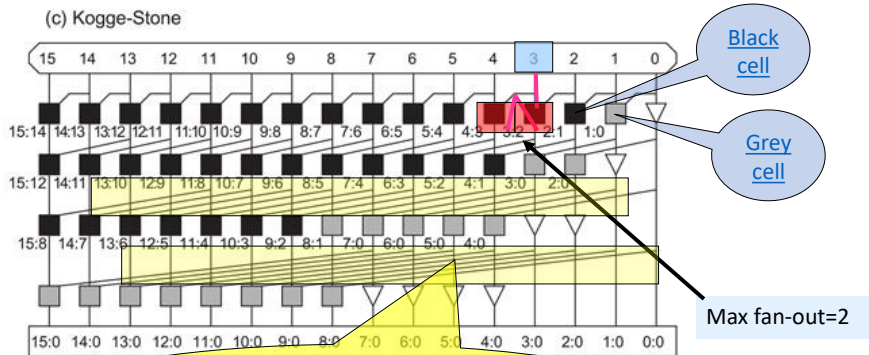


- Απαιτεί $\log_2 N$ επίπεδα για τον τελικό υπολογισμό (μικρότερο delay από τον Brent-Kung)
- Καταλαμβάνει μεγαλύτερη επιφάνεια σε σχέση με τον Brent-Kung
- Δεν έχει πρόβλημα πυκνότητας καλωδίων

VLSI – II

10

Kogge-Stone (1/3)



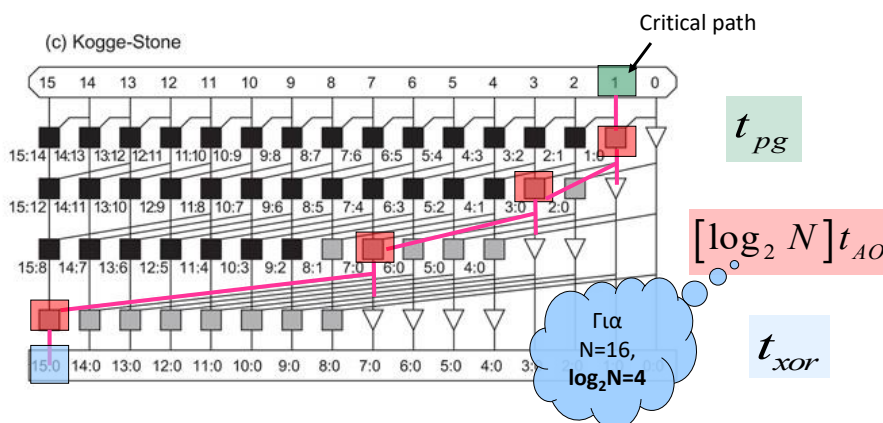
Με αντίστοιχο κόστος τη **μεγάλη πυκνότητα των καλωδίων** για τα οποία πρέπει να ευρεθούν διαδρομές (πρόβλημα)

Μειώνει το max fan-out σε 2 (μία από τις διαδρομές με max fan-out)

VLSI – II

11

Kogge-Stone (2/3)



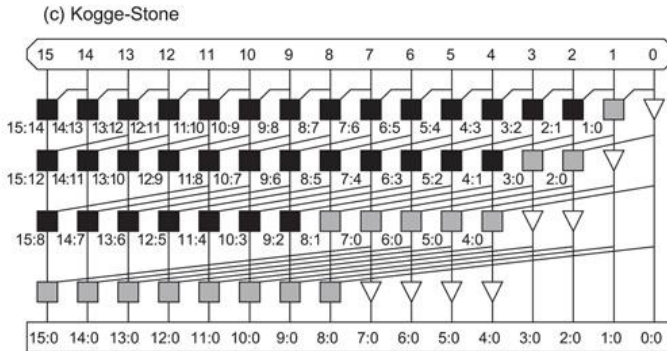
Critical path delay του Kogge-Stone tree adder (ισχύει και για [Slansky](#)):

$$t_{tree} = t_{pg} + [\log_2 N] t_{AO} + t_{xor}$$

VLSI – II

12

Kogge-Stone (3/3)



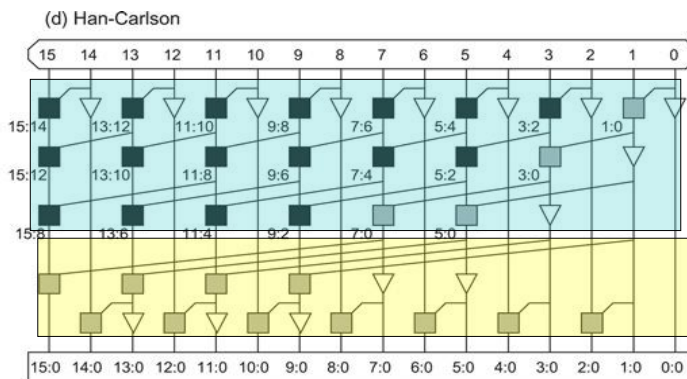
- Απαιτεί $\log_2 N$ επίπεδα για τον τελικό υπολογισμό
- Καταλαμβάνει μεγάλη επιφάνεια (μεγάλος αριθμός από black cells)
- Έχει μεγάλο αριθμό καλωδίων για τα οποία πρέπει να ευρεθούν διαδρομές (πρόβλημα)

VLSI – II

13

13

Han – Carlson (1/4)



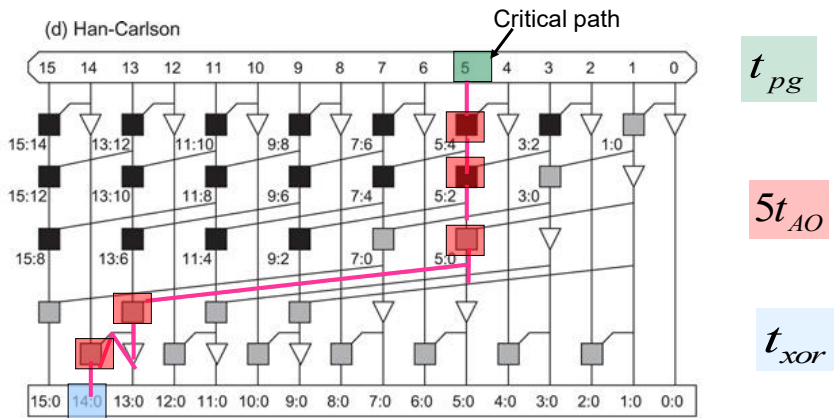
- Οι Han-Carlson trees είναι μία οικογένεια από δίκτυα μεταξύ Kogge-Stone και Brent-Kung
- Στο διάγραμμα χρησιμοποιεί την τεχνική Brent-Kung
- Μείωση επιπέδων με χρήση τεχνικής (μακρών καλωδίων) του Kogge-Stone

VLSI – II

14

14

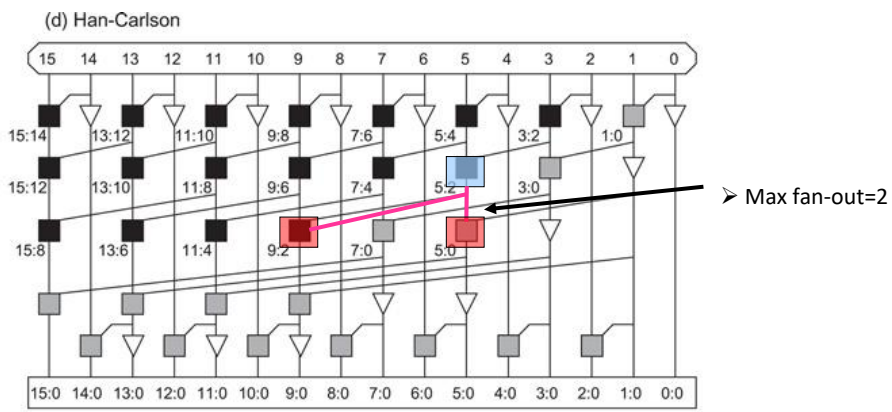
Han – Carlson (2/4)



VLSI – II

15

Han – Carlson (3/4)

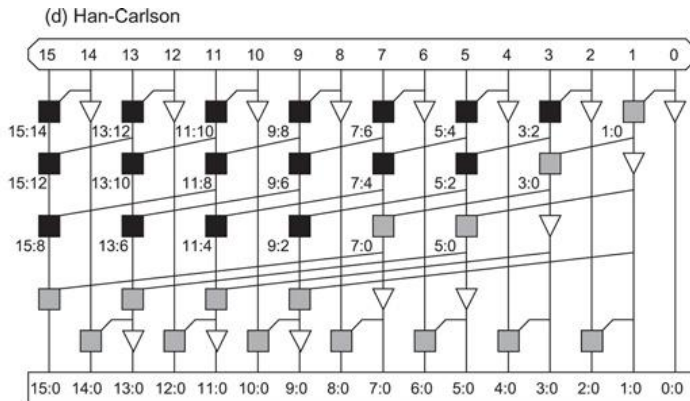


➤ Μία από τις διαδρομές με max fan-out

VLSI – II

16

Han – Carlson (4/4)



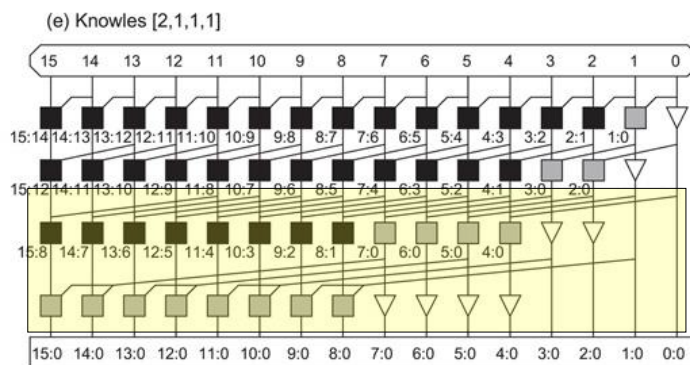
- Έχει ικανοποιητική καθυστέρηση
- Δεν καταλαμβάνει μεγάλη επιφάνεια (αριθμός πυλών)
- Δεν έχει μεγάλη πυκνότητα καλωδίων

VLSI – II

17

17

Knowles (1/4)



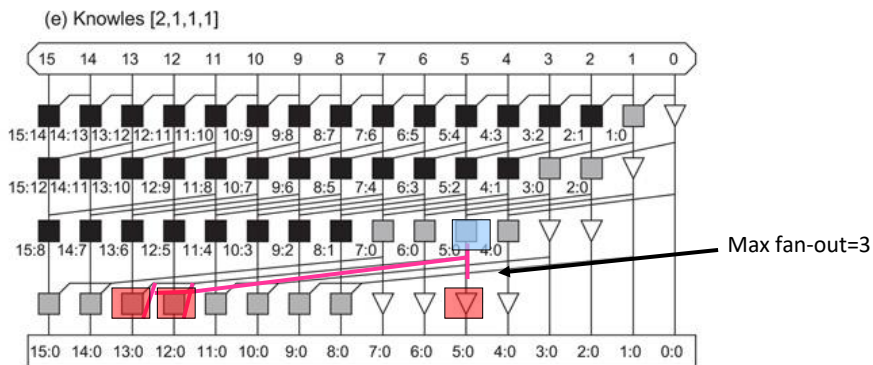
- Είναι συνδυασμός Kogge-Stone και Sklansky.
- Το πρόβλημα του μεγάλου [fan-out που παρουσιάζει ο Sklansky](#) επιλύεται με τη μέθοδο [\(μεγάλων καλωδίων\) Kogge-Stone](#)

VLSI – II

18

18

Knowles (2/4)

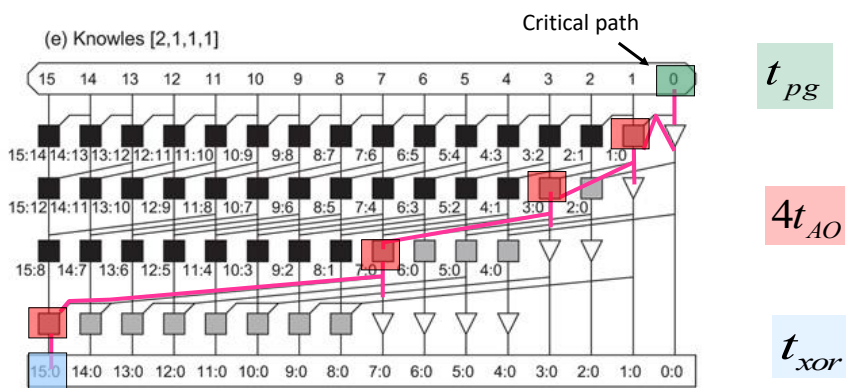


Μία από τις διαδρομές με max fan-out

VLSI – II

19

Knowles (3/4)

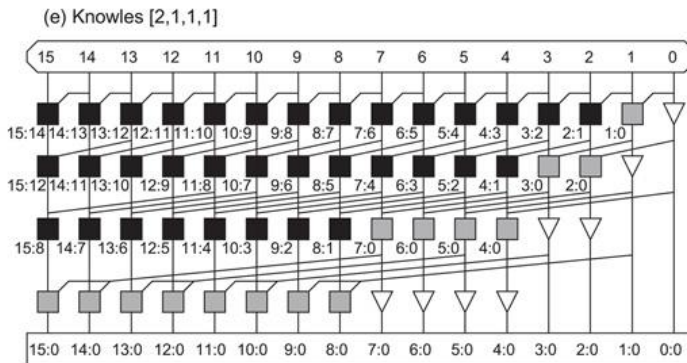


Μία από τις διαδρομές με τη μέγιστη καθυστέρηση

VLSI – II

20

Knowles (4/4)

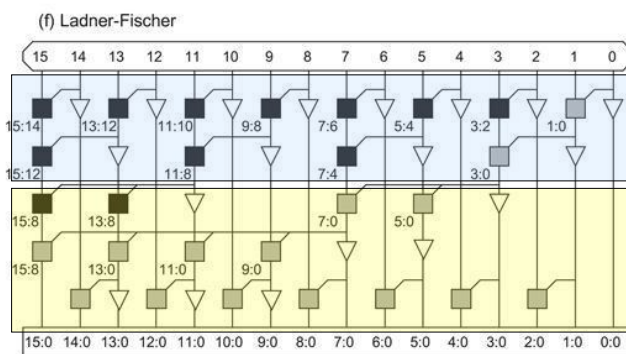


- Απαιτεί $\log_2 N$ επίπεδα για τον τελικό υπολογισμό (η μικρότερη καθυστέρηση)
- Καταλαμβάνει μεγάλη επιφάνεια (αριθμός πυλών)
- Δεν έχει μεγάλο πρόβλημα πυκνότητας καλωδίων

VLSI – II

21

Ladner – Fischer (1/4)



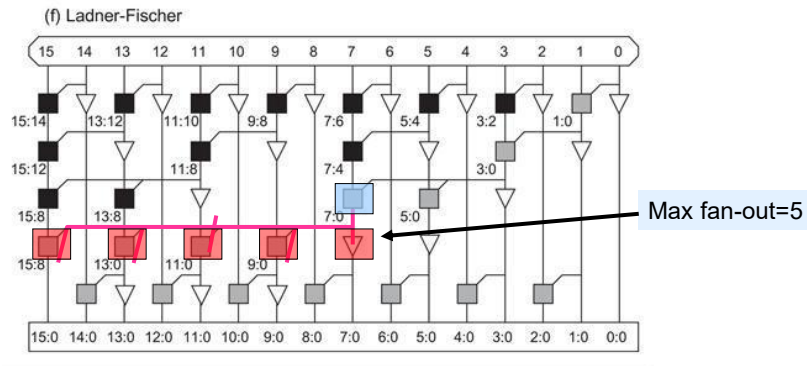
➤ Οι Ladner-Fischer trees είναι οικογένεια δικτύων μεταξύ [Slansky](#) και [Brent-Kung](#)

- Αρχικά γίνονται οι υπολογισμοί όπως στους Brent-Kung trees
- Ενώ οι υπόλοιποι υπολογισμοί γίνονται με τη μέθοδο των Slansky trees

VLSI – II

22

Lander – Fischer (2/4)

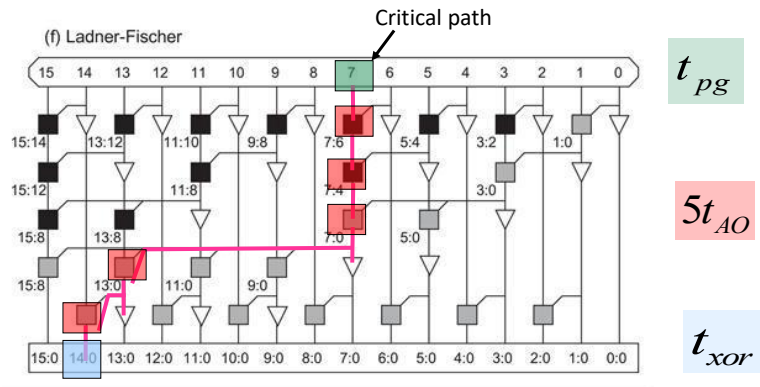


Η διαδρομή με max fan-out

VLSI – II

23

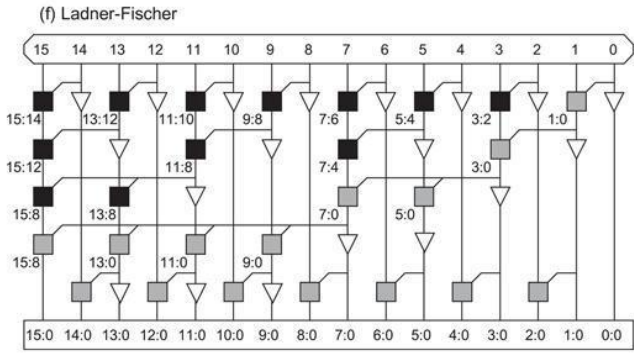
Lander – Fischer (3/4)



VLSI – II

24

Lander – Fischer (4/4)



- Απαιτεί $2\log_2 N - 1$ επίπεδα για τον τελικό υπολογισμό (όχι τόσο καλή καθυστέρηση)
- Καταλαμβάνει πολύ μικρή επιφάνεια
- Δεν έχει πρόβλημα πυκνότητας καλωδίων

VLSI – II

25

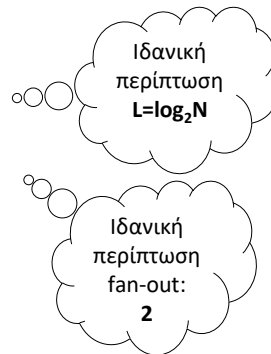
Ομαδοποίηση tree adders (1/2)

Θέτοντας $L = \log_2 N$ γίνεται η περιγραφή του κάθε tree με 3 μεταβλητές:

(l, f, t) στο σύνολο $[0, L-1]$

Αντιστοίχιση χαρακτηριστικών με μεταβλητές:

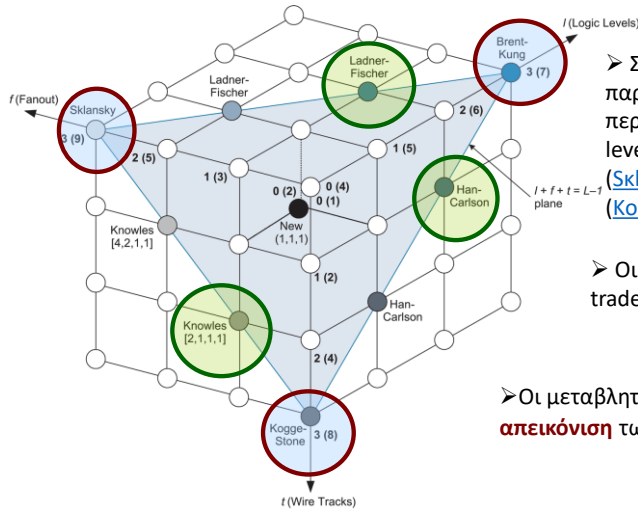
- Λογικά επίπεδα: $L + l$
- Fan-out: $2^f + 1$
- Διαδρομή Καλωδίων: 2^t



VLSI – II

26

Ομαδοποίηση tree adders (2/2)



➤ Στις 3 άκρες του κύβου παρουσιάζονται οι ακραίες περιπτώσεις των trees σε Logic levels (**Brent-Kung**), Fanout (**Sklansky**) και wire Tracks (**Kogge-Stone**)

➤ Οι υπόλοιποι έχουν πιο ισορροπημένα tradeoffs

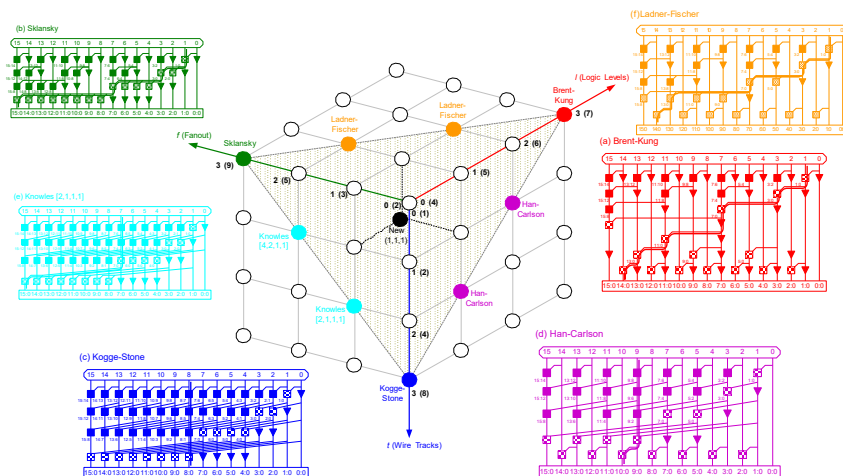
➤ Οι μεταβλητές (**l, f, t**) δίνουν μία **τριδιάστατη απεικόνιση** των tree adders

FIG 10.35 Taxonomy of prefix networks

VLSI – II

27

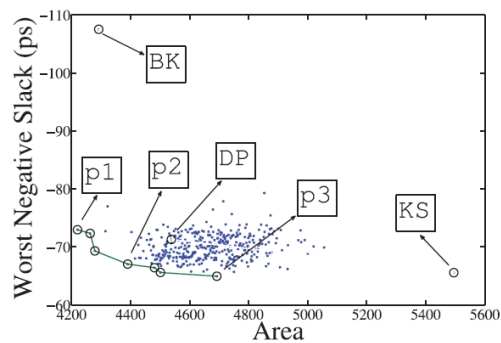
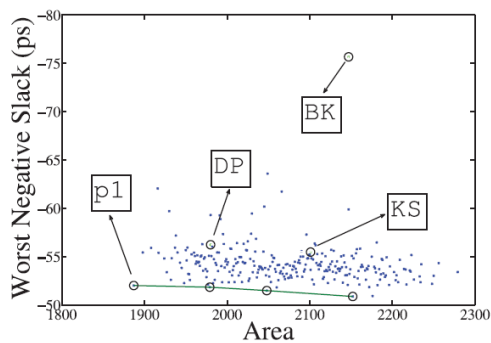
Taxonomy Revisited



VLSI – II

28

Αλγοριθμική κατασκευή αθροιστών παράλληλου προθέματος

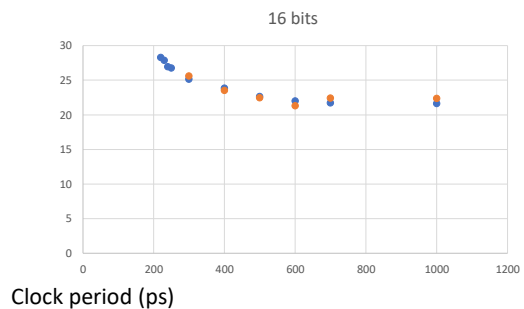
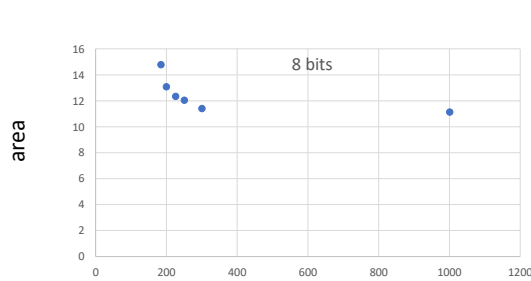


- S. Roy, M. Choudhury, R. Puri and D. Z. Pan, "Towards Optimal Performance-Area Trade-Off in Adders by Synthesis of Parallel Prefix Structures," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1517-1530, Oct. 2014, doi: 10.1109/TCAD.2014.2341926.

29

29

Αποτελέσματα σύνθεσης και βελτιστοποίησης

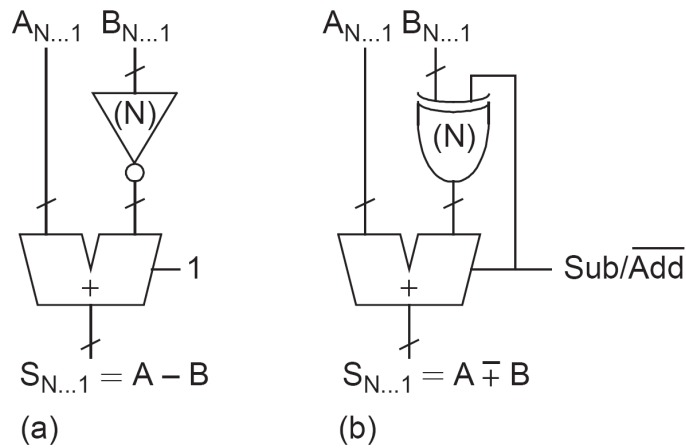


- Διαγράμματα area – clock period μετά από σύνθεση και βελτιστοποίηση
 - Για διαφορετικά clock period constraints
 - Carry-select vs ripple carry
 - 7 nm standard cell library

30

30

Αφαίρεση

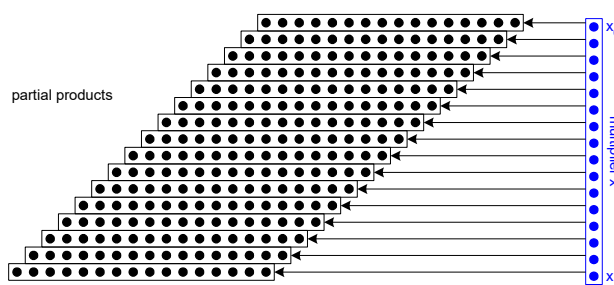


VLSI - II

31

31

Διάγραμμα κουκίδων (dot diagram)



Τα δυαδικά ψηφία του πολλαπλασιαστή $X = x_{15}x_{14} \dots x_0$ που χρησιμοποιούνται για την παραγωγή των μερικών γινομένων

- Πολλαπλασιασμοί μεγάλων αριθμών απεικονίζονται ευκολότερα με τα διαγράμματα κουκίδων
- Κάθε κουκίδα αντιπροσωπεύει μια θέση για ένα δυαδικό ψηφίο
- Τα μερικά γινόμενα αναπαριστώνται από μια οριζόντια γραμμή κουκίδων, ολισθημένη σύμφωνα με το βάρος τους

VLSI - II

32

32

Γενικές αρχές υλοποίησης πολλαπλασιασμού (1/2)

- Πλήθος τεχνικών για την εκτέλεση του πολλαπλασιασμού
- Η επιλογή βασίζεται πάνω σε μετρικές σχεδιασμού
 - η καθυστέρηση, ο ρυθμός λειτουργίας (throughput rate), επιφάνεια και πολυπλοκότητα
- Η προφανής λύση είναι η χρήση αθροιστή διάδοσης κρατουμένου (CPA) $M+1$ bits σε δομή αλυσίδας
 - Χρειάζεται $N-1$ CPAs και είναι αργή, ακόμα κι αν χρησιμοποιηθεί ένας γρήγορος CPA
- Αποδοτικότερες δομές με χρήση ορισμένου τύπου πίνακα ή δένδρων αθροιστών για την πρόσθεση των μερικών γινομένων

33

33

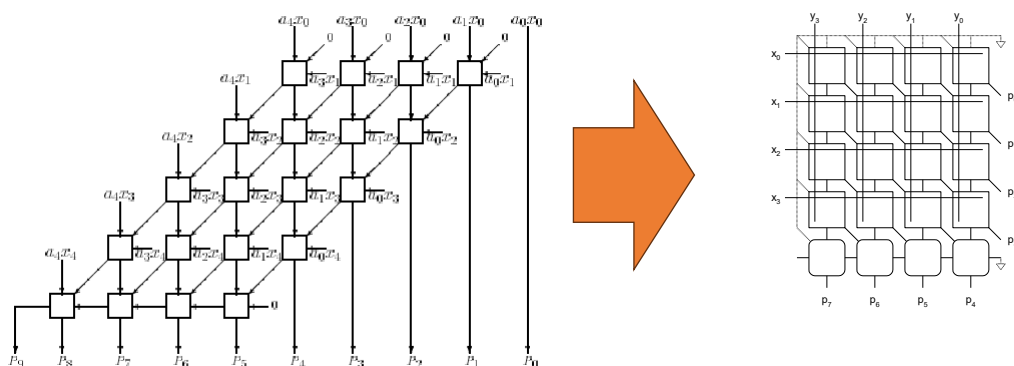
Γενικές αρχές υλοποίησης πολλαπλασιασμού (2/2)

- Κλασσική δομή πίνακα για μη προσημασμένους αριθμούς
- Τροποποίηση πίνακα για προσημασμένους αριθμούς σε συμπλήρωμα ως προς 2 – αλγόριθμος Baugh-Woolley
- Κωδικοποίησης Booth για μείωση πλήθους μερικών γινομένων
- Δένδρα Wallace για μείωση λογικών επιπέδων πρόσθεσης
 - Τα δένδρα Wallace οδηγούν σε πολύπλοκα layouts και έχουν μεγάλου μήκους, μη κανονικές διασυνδέσεις
- Υβριδικές δομές πινάκων / δένδρων

34

34

Ορθογώνια διάταξη πίνακα (Rectangular Array)



- Ίδιοι αθροιστές με CSA, μετατοπισμένοι για να ταιριάζουν σε ένα ορθογώνιο σχήμα

35

Πολλαπλασιασμός σε αριθμών σε αναπαράσταση συμπληρώματος του δύο

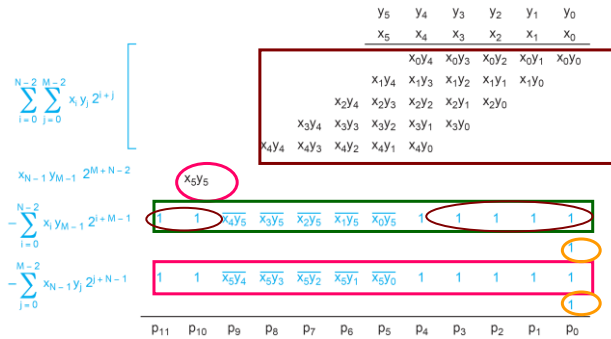
$$x = -x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i \quad y = -y_{M-1}2^{M-1} + \sum_{j=0}^{M-2} y_j 2^j$$

$$x \cdot y = \left(-x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i \right) \left(-y_{M-1}2^{M-1} + \sum_{j=0}^{M-2} y_j 2^j \right)$$

$$= x_{N-1}y_{M-1}2^{M+N-2} + \sum_{i=0}^{N-2} \sum_{j=0}^{M-2} x_i y_j 2^{i+j} - x_{N-1} \sum_{j=0}^{M-2} y_j 2^j - y_{M-1} \sum_{i=0}^{N-2} x_i 2^i$$

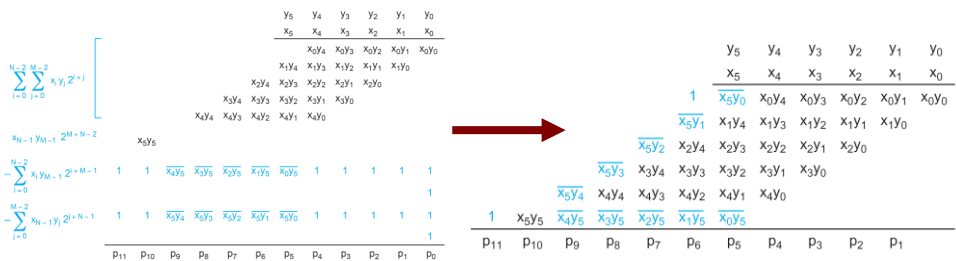
36

Πολλαπλασιασμός αριθμών σε αναπαράσταση συμπληρώματος ως προς 2 με Baugh – Wooley (2/4)



- Γινόμενων μη αρνητικών τμημάτων
- Γινόμενο των δύο MSBs
- **Υπό συνθήκη** αφαιρέσεις ως άθροισμα συμπληρώματος του δύο
 - Ανεστραμμένα bits
 - Πρόσθεση 1 στο λιγότερο σημαντικού ψηφίου
- Συνθήκες είναι οι τιμές των MSBs

Πολλαπλασιασμός σε συμπλήρωμα ως προς 2 (3/4)



- Η καθυστέρηση εξαρτάται από τον αριθμό των γραμμών των μερικών γινομένων που θα προστεθούν
- Ο τροποποιημένος πολλαπλασιαστής Baugh-Wooley μειώνει τον αριθμό των μερικών γινομένων προ-υπολογίζοντας τα αθροίσματα των σταθερών 1 και μεταθέτοντας κάποιους προς τα πάνω σε επιπλέον στήλες

Πολλαπλασιασμός σε συμπλήρωμα ως προς 2 (4/4)

The diagram illustrates a 4x4 array multiplier circuit. The inputs are x_3, x_2, x_1, x_0 and y_3, y_2, y_1, y_0 . The outputs are $p_7, p_6, p_5, p_4, p_3, p_2, p_1, p_0$. The circuit uses a grid of 16 AND gates. The inputs x_3, x_2, x_1, x_0 are connected to the top of the grid, and y_3, y_2, y_1, y_0 are connected to the left. The outputs of the AND gates are connected to the outputs $p_7, p_6, p_5, p_4, p_3, p_2, p_1, p_0$. The output p_0 is the least significant bit, and p_7 is the most significant bit. The circuit is designed to multiply two 4-bit numbers in two's complement representation.

	y_5	y_4	y_3	y_2	y_1	y_0
x_5	x_5	x_4	x_3	x_2	x_1	x_0
1	$\overline{x_5}y_0$	x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0
$\overline{x_5}y_1$	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0	
$\overline{x_5}y_2$	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0	
$\overline{x_5}y_3$	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0	
$\overline{x_5}y_4$	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0	
1	x_5y_5	$\overline{x_4}y_5$	$\overline{x_3}y_5$	$\overline{x_2}y_5$	$\overline{x_1}y_5$	$\overline{x_0}y_5$
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5
						p_4
						p_3
						p_2
						p_1

39

Κωδικοποίηση Booth – Βασική ιδέα

- Έστω μια δυαδική ακολουθία

Θέση		$i+k$	$i+k-1$	$i+k-2$	...	$i+1$	i	$i-1$	...	...
Τιμή		0	1	1	...	1	1	0		

k συνεχόμενοι "1"

➤ Με βάση τη σχέση $2^{i+k} - 2^i = 2^{i+k-1} + 2^{i+k-2} + \dots + 2^{i+1} + 2^i$

Θέση		$i+k$	$i+k-1$	$i+k-2$	...	$i+1$	i	$i-1$	...	...
Τιμή		1	0	0	...	0	-1	0		

Αφαίρεση

Πρόσθεση

k συνεχόμενα "0"

➤ Απαιτούνται κατάλληλες ολισθήσεις και **1 πρόσθεση (+A)** και **1 αφαίρεση (-A)** αντί k προσθέσεις (**A+A+...+A**) του πολλαπλασιαστέου A

➤ Εισάγονται 2 βοηθητικά (dummy) bits $b_n, b_{-1}=0$ και στο $B=b_{n-1}, b_{n-2}, \dots, b_1, b_0$

VLSI – II

41

41

Πίνακας κωδικοποίησης Booth (1/2)

Πολλαπλασιαστής		
Bit i	Bit $i+1$	Λειτουργία
0	0	$0 \times \text{πολλαπλασιαστέος } (0 \times A)$
0	1	$+1 \times \text{πολλαπλασιαστέος } (+1 \times A)$
1	0	$-1 \times \text{πολλαπλασιαστέος } (-1 \times A)$
1	1	$0 \times \text{πολλαπλασιαστέος } (0 \times A)$

➤ Χρησιμοποιεί μόνο τους όρους 0, +A, -A και κατάλληλες ολισθήσεις

➤ Με βάση τον παραπάνω πίνακα ο αριθμός 0011110 (+30) κωδικοποιείται σε 0+1000-10 ($32-2=30$)

VLSI – II

42

42

Πίνακας κωδικοποίησης Booth (2/2)

Worst case	0	1	0	1	0	1	0	1
Κωδικοποίηση	+1	-1	+1	-1	+1	-1	+1	-1
Best case	0	0	1	1	1	1	0	0
Κωδικοποίηση	0	+1	0	0	0	-1	0	0

43

43

Πολλαπλασιασμός Booth – Παράδειγμα

0 1 0 1 0 1 1
0 0 1 1 1 1 0

0 0 0 0 0 0 0
0 1 0 1 0 1 1
0 1 0 1 0 1 1
0 1 0 1 0 1 1
0 1 0 1 0 1 1
0 1 0 1 0 1 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0

0 0 1 0 1 0 0 0 0 1 0 1 0

2's
complement

Συμβατικός

0 1 0 1 0 1 1
0 +1 0 0 0 -1 0

0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 1 1 1
0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 0 0 0 1 0 1 0

Booth

44

44

Κωδικοποίηση Booth – Radix 4 (1/3)

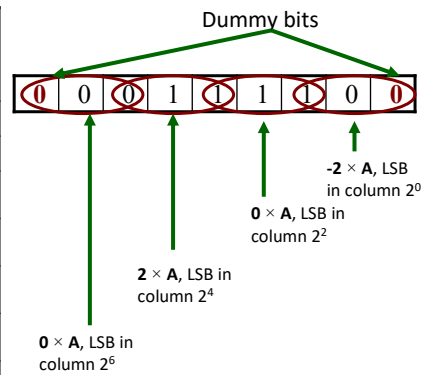
- Η προηγούμενη κωδικοποίηση επιταχύνει τον πολλαπλασιασμό παρακάμπτοντας ακολουθίες από συνεχόμενους “1...1”
- Η διαδικασία μπορεί να επιταχυνθεί περισσότερο συνδυάζοντας τριάδες ψηφίων του πολλαπλασιαστή
 - Εξετάζει ένα ζεύγος ψηφίων λαμβάνοντας υπόψη το αμέσως προηγούμενο ψηφίο δεξιά
- Οδηγεί στην παραγωγή το πολύ $n/2$ μερικών γινομένων για έναν n-bit πολλαπλασιαστή
- Ισχύει για προσημασμένους και μη προσημασμένους αριθμούς (όπως και η προηγούμενη)

Πίνακας κωδικοποίησης Booth radix 4 (2/3)

Ζεύγος ψηφίων πολλαπλασιαστή		Ψηφίο δεξιά	Λειτουργία	Εξήγηση
2^1	2^0			
$i + 1$	i	$i - 1$		
0	0	0	$0 \times A$	No string
0	0	1	$+1 \times A$	End of string
0	1	0	$+1 \times A$	Single 1 ($+2-1$)
0	1	1	$+2 \times A$	End of string
1	0	0	$-2 \times A$	Beginning of string
1	0	1	$-1 \times A$	End/ beginning of string
1	1	0	$-1 \times A$	Beginning of string
1	1	1	$0 \times A$	String of 1s

Κωδικοποίηση Radix-4 (3/3)

Ζεύγος ψηφίων πολλαπλασιαστή			Λειτουργία	Εξήγηση
2^1	2^0			
$i+1$	i	$i-1$		
0	0	0	$0 \times A$	No string
0	0	1	$+1 \times A$	End of string
0	1	0	$+1 \times A$	Single 1 ($+2^{-1}$)
0	1	1	$+2 \times A$	End of string
1	0	0	$-2 \times A$	Beginning of string
1	0	1	$-1 \times A$	End/ beginning of string
1	1	0	$-1 \times A$	Beginning of string
1	1	1	$0 \times A$	String of 1s



VLSI – II

2011-2012

47

47

Πολλαπλασιασμός Booth – Παράδειγμα

0 0 0 0 1 1

0 1 1 1 0 1

0 0 0 0 1 1

0 0 0 0 0 0

0 0 0 0 1 1

0 0 0 0 1 1

0 0 0 0 0 0

0 0 0 0 1 0 1 0 1 1 1

Συμβατικός

Booth Radix2

2's complement

0 0 0 0 1 1

0 1 0 0 -1 0 1

0 0 0 0 0 0 0 0 0 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 1 1

0 0 0 0 0 1 0 1 0 1 1 1

0 0 0 0 0 1 1

0 1 1 1 0 1 1

2×A -1×A 1×A

Booth Radix4

0 0 0 0 0 0 0 0 0 0 0 1 1

1 1 1 1 1 1 1 1 0 1

0 0 0 0 0 0 1 1 0

0 0 0 0 0 0 0 1 0 1 0 1 1 1

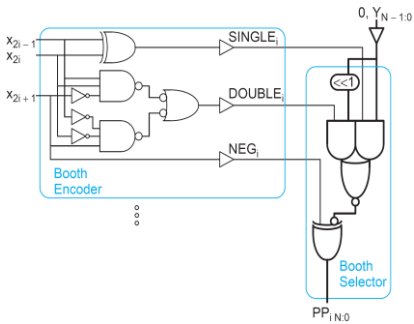
VLSI – II

48

48

Κυκλώματα Κωδικοποίησης & Επιλογής Booth

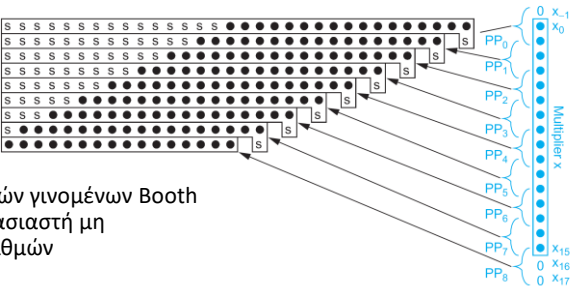
Inputs			Partial Product	Booth Selects		
x_{2i+1}	x_{2i}	x_{2i-1}	PP_i	$SINGLE_i$	$DOUBLE_i$	NEG_i
0	0	0	0	0	0	0
0	0	1	Y	1	0	0
0	1	0	Y	1	0	0
0	1	1	$2Y$	0	1	0
1	0	0	$-2Y$	0	1	1
1	0	1	$-Y$	1	0	1
1	1	0	$-Y$	1	0	1
1	1	1	$-0 (= 0)$	0	0	1



- Το κύκλωμα κωδικοποίησης παράγει τα σήματα (single, double, neg)
- Το κύκλωμα επιλογής δέχεται τα σήματα (single, double, neg) και τον πολλαπλασιαστέο Y εκτεταμένο ως προς το μηδέν σε $N + 1$ bits –έξοδος τιμές $0, Y, 2Y$
- Αν το μερικό γινόμενο είναι αρνητικό (neg=1) χρησιμοποιείται το συμπλήρωμα του δύο

49

Επέκταση προσήμου



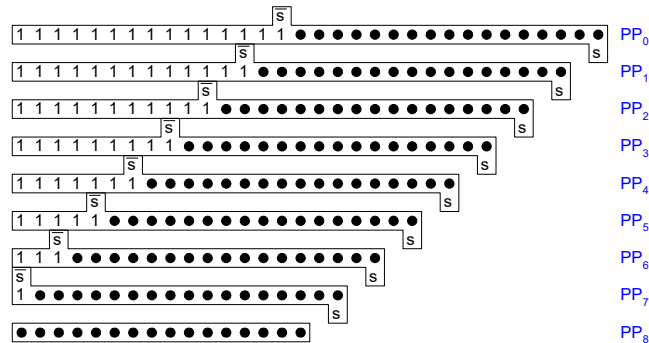
16-bit πίνακας μερικών γινομένων Booth radix-4 για πολλαπλασιαστή μη προσημασμένων αριθμών

- Ακόμα και σε μη προσημασμένους αριθμούς τα αρνητικά μερικά γινόμενα πρέπει να επεκταθούν ως προς το πρόσημο για να προστεθούν σωστά
- Κάθε μερικό γινόμενο επεκτείνεται ως προς το πρόσημο με βάση το σήμα neg_i
- Προστίθεται s στο LSB στην επόμενη γραμμή (το συμπλήρωμα ως προς 2)
- Μεγάλες απαιτήσεις fanout για τα MSBs

50

Απλοποιημένη επέκταση προσήμου (1/2)

- Τα Sign bits είναι είτε όλα 0's είτε όλα 1's
 - Όμως το όλα 0's είναι ισοδύναμο με το όλα 1's + 1 στην κατάλληλη στήλη
 - Η ιδέα αυτή χρησιμοποιείται για να ελαττώσει το φορτίο του MSB



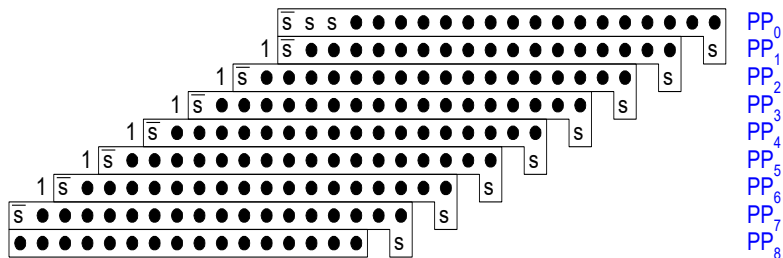
VLSI – II

51

51

Απλοποιημένη επέκταση προσήμου (2/2)

- Δε χρειάζεται να γίνονται όλες οι προσθέσεις των 1's in hardware
 - Προϋπολογισμός έξω από τον πίνακα



VLSI – II

52

52

Τροποποιημένος πίνακας για αρνητικούς αριθμούς

- Τα ψηφία προσήμου πρέπει να επεκταθούν κατάλληλα
 - Στη 1^η γραμμή έχουμε 11 αντί 6 ψηφία κοκ
- Αυξάνει την πολυπλοκότητα των multi-operand adder
- Αν χρησιμοποιηθεί 1's complement και πρόσθεση 1 στο LSB => ακόμη μεγαλύτερη αύξηση των στηλών και πολυπλοκότητα των multi-operand adder

10	9	8	7	6	5	4	3	2	1	0
●	●	●	●	●	○	○	○	○	○	○
●	●	●	●	○	○	○	○	○	○	
●	●	●	○	○	○	○	○	○		
●	●	○	○	○	○	○	○			
●	○	○	○	○	○	○				
○	○	○	○	○	○					

53

53

Μείωση πολυπλοκότητας

- Two's complement αριθμός **SSSSSS Z₄Z₃Z₁Z₀** με τιμή

$$-s \cdot 2^{10} + s \cdot 2^9 + s \cdot 2^8 + s \cdot 2^7 + s \cdot 2^6 + s \cdot 2^5 + z_4 \cdot 2^4 + z_3 \cdot 2^3 + z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0$$

- Αντικαθίσταται από 00000 (-s) z₄z₃z₂z₁z₀ αφού

$$\begin{aligned} & -s \cdot 2^{10} + s \cdot (2^9 + 2^8 + 2^7 + 2^6 + 2^5) \\ &= -s \cdot 2^{10} + s \cdot (2^{10} - 2^5) = -s \cdot 2^5. \end{aligned}$$

VLSI – II

54

54

Πρόσθεση μερικών γινομένων

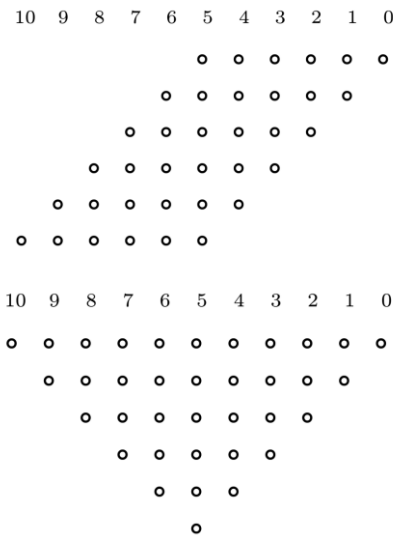
- Τα μερικά γινόμενα πρέπει να προστεθούν για την παραγωγή του τελικού αποτελέσματος
- Χρήση αθροιστών πολλαπλών ορισμάτων
 - Fast multi-operand adder
- Η δομή των μερικών γινομένων πρέπει να ληφθεί υπόψη ώστε να ελαττωθεί η πολυπλοκότητα
- Μερικά μερικά γινόμενα έχουν μικρότερο πλήθος ψηφίων από το μέγιστο
 - πρέπει να ευθυγραμμιστούν κατάλληλα
 - απαιτούν λιγότερους και απλούστερους αθροιστές / μετρητές

55

55

Παράδειγμα - 6 Partial Products

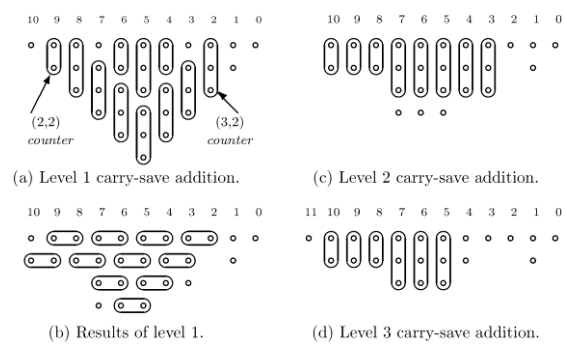
- Παράγονται όταν πολ/νται μη προσημασμένοι 6-bit αριθμοί
- 6 operands μπορούν να προστεθούν χρησιμοποιώντας 3 επίπεδα CSAs (Wallace tree)
- Το πλήθος των (3,2) μετρητών μπορεί να μειωθεί δραστικά εκμεταλλευόμενοι το γεγονός ότι μόνο μια στήλη έχει 6 ψηφία
- Επανασχεδίαση του διαγράμματος κουκίδων για την επιλογή των (3,2) μετρητών



56

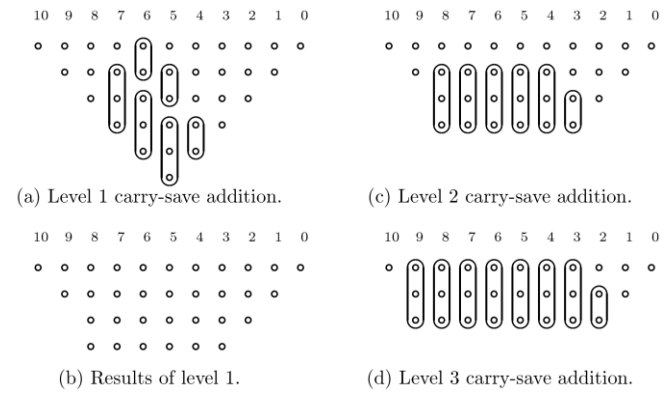
56

Μείωση πολυπλοκότητας - Χρήση (2,2) Counters (HAs)



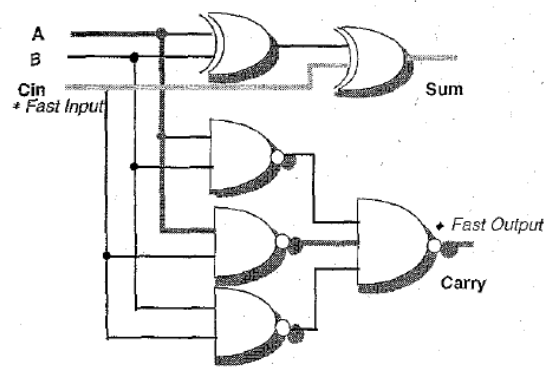
- Ο αριθμός των επιπέδων παραμένει 3 αλλά λιγότεροι counters

Επιπλέον μείωση του πλήθους των μετρητών



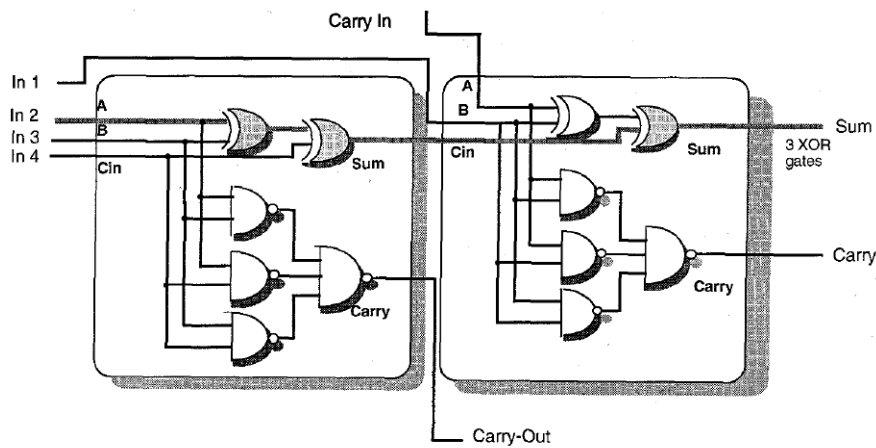
- Reduce # of bits to closest element of 3,4,6,9,13,19,...
- 15 (3,2) and 5 (2,2) vs. 16 (3,2) and 9 (2,2) counters

Γρήγορες είσοδοι – γρήγορες έξοδοι



59

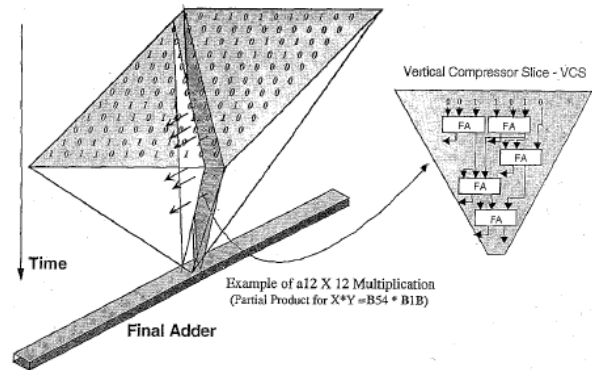
Μια υλοποίηση του [4:2] compressor



V.G. Oklobdzija, D.Villeger, and S.S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," *IEEE Transactions on Computers*, pp. 294 – 306, March 1996.

60

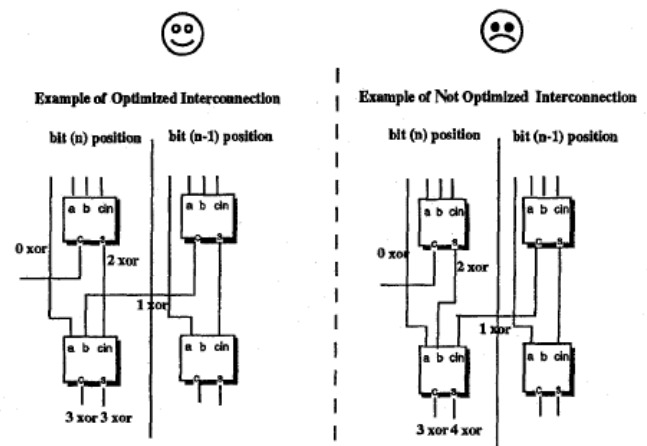
3D άποψη της αναγωγής μερικών γινομένων



61

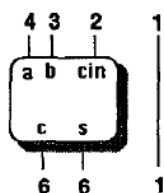
Ελαχιστοποίηση καθυστερήσης με βέλτιστη ανάθεση εισόδων

Example of Delay Optimization

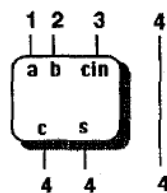


62

Μοντέλο καθυστέρησης πλήρους αθροιστή ενός ψηφίου



Worst Case



TDM Arrangement

$$\begin{aligned} & \text{Delay}(S) \\ &= \text{MAX} \{ \text{Delay}(A) + D_{A-S}, \text{Delay}(B) + D_{B-S}, \text{Delay}(C_{in}) + D_{Cin-S} \} \\ & \text{Delay}(C) \\ &= \text{MAX} \{ \text{Delay}(A) + D_{A-C}, \text{Delay}(B) + D_{B-C}, \text{Delay}(C_{in}) + D_{Cin-C} \} \end{aligned}$$

63

Αλγόριθμος κατασκευής δένδρου αναγωγής

Αρχικοποίηση:

Form $2N - 1$ lists L_i ($i = 0, 1, \dots, 2N - 2$) each consisting of p_i elements where:

$$p_i = i + 1 \text{ for } i \leq N - 1 \text{ and}$$

$$p_i = 2N - 1 - i \text{ for } i \geq N$$

An element of a list L_i ($j = 0, 1, \dots, p_i - 1$) is a pair: $\langle d_j, n_j \rangle_i$

where:

n_j : is a unique node identifying name

d_j : is a delay associated with that node representing a delay of a signal arriving to the node n_j with respect to some reference point.

For $i = 0, 1$ and $2N - 2$:

connect nodes from the corresponding lists L_i directly to the CPA;

64

Αλγόριθμος (2)

Partial Product Array Generation:

For $I = 2$ to $I = 2N - 3$ Begin For

if length of L_i is even Then

Begin If

Sort the elements of L_i in ascending order by the values of delay d_1 ;

connect an HA to the first 2 elements of L_i starting with the slowest input;

$D_s = \max \{d_A + d_{A-s}, d_B + d_{B-s}\}$

$D_c = \max \{d_A + d_{A-c}, d_B + d_{B-c}\}$

remove 2 elements from L_i ;

insert the pair $\langle D_s, \text{NetName} \rangle$ into L_i ;

insert the pair $\langle D_c, \text{NetName} \rangle$ into L_{i+1} ;

decrement the length of L_i ;

increment the length of L_{i+1} ;

End If;

while length of $L_i > 3$

Begin While

sort the elements of L_i in ascending order by the values of delay d_1 ;

connect an FA to the first 3 elements of L_i starting with the slowest input of the FA:

$D_s = \max \{dcA + dcA-s, dcB + dcB-s\}$;

$D_c = \max \{dcA + dcA-c, dcB + dcB-c\}$;

remove 3 elements from L_i ;

insert the pair $\langle D_s, \text{NetName} \rangle$ into L_i ;

insert the pair $\langle D_c, \text{NetName} \rangle$ into L_{i+1} ;

subtract 2 from the length of L_i ;

increment the length of L_{i+1} ;

End While;

sort the elements of L_i ;

connect an FA to the last 3 nodes of L_i ;

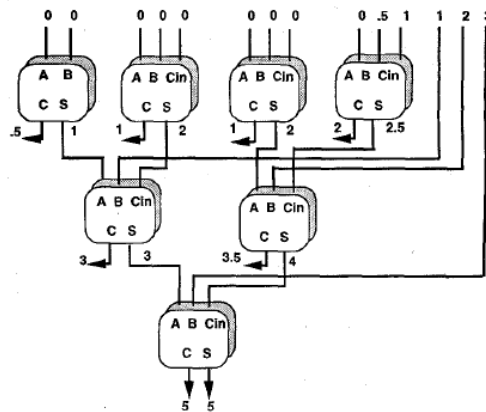
connect the S and C to the bit i and $I + 1$ of the CPA;

End For;

End Method;

65

Παράδειγμα TDM



66

TABLE 1
COMPARISON BETWEEN TDM
AND OTHER REPRESENTATIVE SCHEMES,
IN XOR LEVELS USED IN THE PARTIAL PRODUCT ARRAY

Multiplier Word-length	Wallace Tree [7]	4:2 Tree [11]*	Fadavi-Ardekani [16]	TDM
3	2	2	2	2
4	4	3	3	3
6	6	6 (5)	5	5
8	8	6	7	5
9	8	8	7	6
11	10	9 (8)	8	7
12	10	9 (8)	8	7
16	12	9	10	8
19	12	12 (11)	11	9
24	14	12 (11)	12	10
32	16	12	13	11
42	16	15 (14)	14	12
53	18	15	15	13
64	20	15	16	14
95	20	18 (17)	17	15

* Number in parenthesis represent delays when a Full Adder is used (instead of 4:2 compressor) every time the column size is found to be three.

67

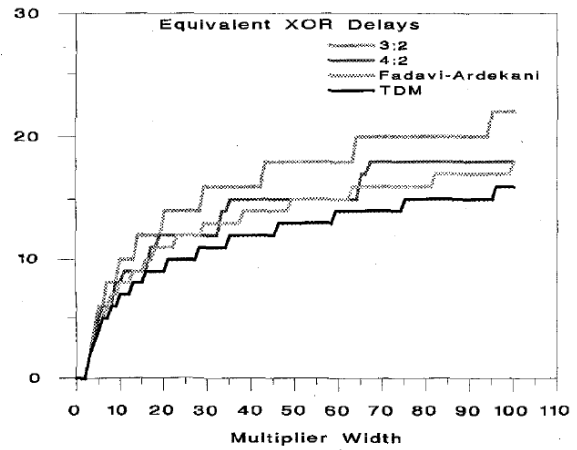
Μέγιστη
καθυστέρηση
διαδρομής

TABLE 2
CRITICAL PATH DELAY [CMOS: $L_{EFF} = 1\mu$, $T = 25^\circ C$, $V_{CC} = 5V$]

N = 24 bits	4:2 Design	9:2 Design	Fadavi- Ardekani	TDM Design
Delay [nS]	14.0	13.0	11.7	10.5

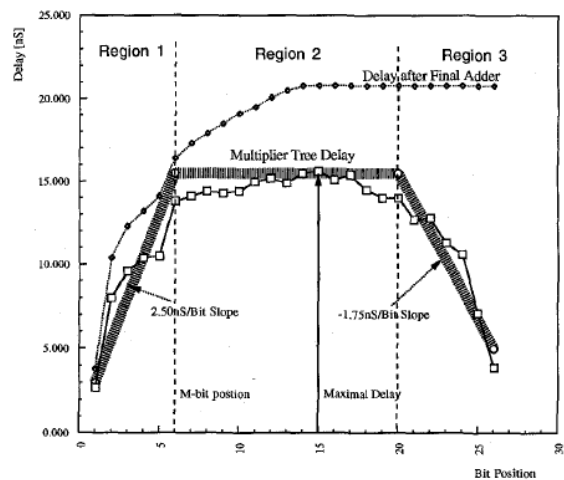
68

Σύγκριση (3,2), [4:2], F-A, TDM



69

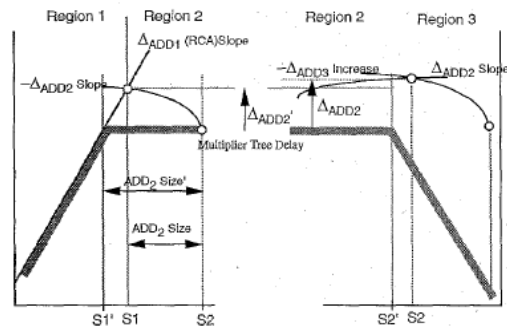
Ο ρόλος του τελικού αθροιστή



70

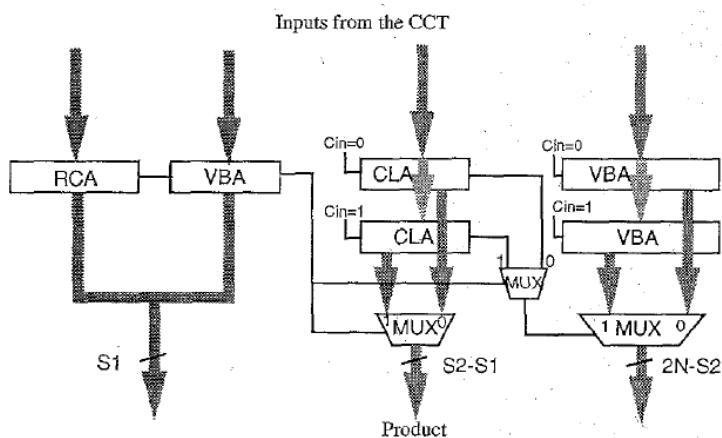
Οργάνωση CPA: Καθορισμός περιοχών

- Περιοχή 1: Γραμμική καθυστέρηση με τη θέση του δυαδικού ψηφίου.
- Περιοχή 2: Τα ψηφία καταφθάνουν περίπου ταυτόχρονα.
- Περιοχή 3: Τα περισσότερα σημαντικά ψηφία καταφθάνουν νωρίτερα.



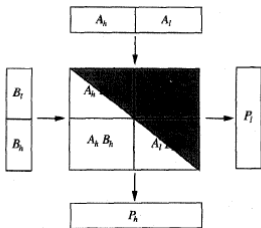
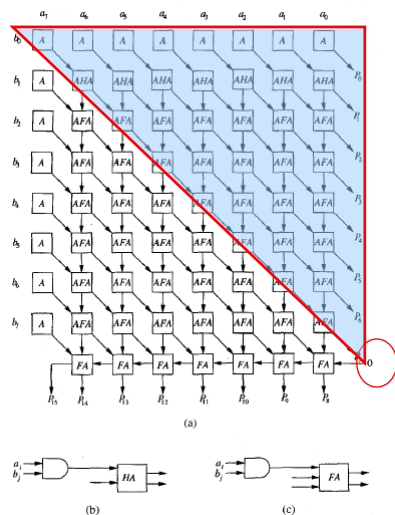
71

Τελική οργάνωση CPA



72

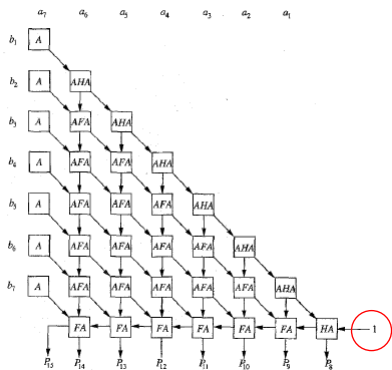
Ο πολλαπλασιαστής των Kidambi, El-Guibaly και Antoniou¹



¹“Area-Efficient Multipliers for Digital Signal Processing Applications,” *IEEE Transactions on Circuits and Systems – Part II*, 43(2), σελ. 90 - 95, Feb. 96.

73

Πολλαπλασιαστές Σταθερού Μήκους

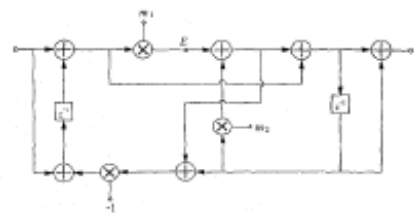
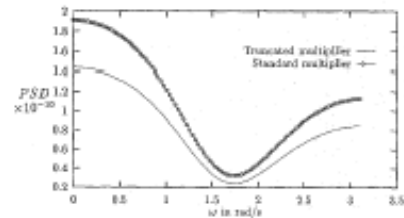


74

Επιπτώσεις σε φίλτρα

- Εφαρμογή του συγκεκριμένου πολλαπλασιαστή (16×16) σε κατωδιαβατό Butterworth wave φίλτρο δεύτερης τάξης και δομής GIC (generalized-immittance converter)

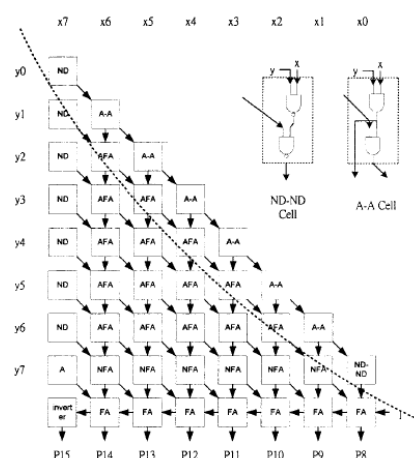
- A. Antoniou, *Digital filters analysis and design*, McGraw Hill, 1993.



75

Το ίδιο πρόβλημα από άλλη πλευρά...²

- ² Van, Wang, και Feng, "Design of the lower-error fixed-width multiplier and its application," *IEEE Transactions on Circuits and Systems – Part II*, 47(10), Oct 2000.

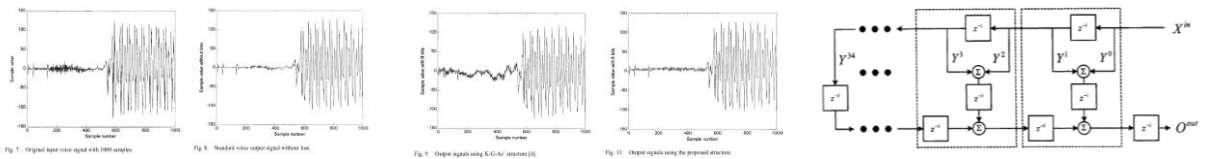


76

Επιδόσεις του δεύτερου πολλαπλασιαστή σε ψηφιακά φίλτρα φωνής

FIR φίλτρο

35 συντελεστών



77

Κατανεμημένη (Distributed) Αριθμητική

- Υπολογισμός εσωτερικού γινομένου σταθερού διανύσματος $C = [c_1 c_2 \dots c_N]$ με μεταβλητό διάνυσμα $X = [x_1 x_2 \dots x_N]$.

$$Y = C \cdot X = \sum_{i=1}^{N-1} c_i x_i$$

- Εφαρμογές: φίλτρα σταθερών συντελεστών, διακριτοί μετασχηματισμοί όπως ο DCT...

78

Παράδειγμα Κατανεμημένης Αριθμητικής

$$(001)_2 = 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ = (0.125)_{10}$$

$$Y = c_0 x_0 + c_1 x_1 + c_2 x_2 + c_3 x_3$$

$$Y = c_0 (001)_2 + c_1 (010)_2 + c_2 (110)_2 + c_3 (101)_2$$

$$= 2^{-1} (c_0 \cdot 0 + c_1 \cdot 0 + c_2 \cdot 1 + c_3 \cdot 1) + \\ + 2^{-2} (c_0 \cdot 0 + c_1 \cdot 1 + c_2 \cdot 1 + c_3 \cdot 0) + \\ + 2^{-3} (c_0 \cdot 1 + c_1 \cdot 0 + c_2 \cdot 0 + c_3 \cdot 1) \\ = 2^{-1} f(0, 0, 1, 1) + 2^{-2} f(0, 1, 1, 0) + 2^{-3} f(1, 0, 0, 1)$$

$$f(x_{0,j}, x_{1,j}, x_{2,j}, x_{3,j}) \triangleq c_0 x_{0,j} + c_1 x_{1,j} + c_2 x_{2,j} + c_3 x_{3,j}$$

Τρεις κύκλοι ρολογιού

Δεν απαιτούνται πολλαπλασιασμοί.

$x_{0,j}$	$x_{1,j}$	$x_{2,j}$	$x_{3,j}$	Περιεχόμενα Θέσης Μνήμης
0	0	0	0	0
0	0	0	1	c_3
0	0	1	0	c_2
0	0	1	1	$c_2 + c_3$
0	1	0	0	c_1
0	1	0	1	$c_1 + c_3$
0	1	1	0	$c_1 + c_2$
0	1	1	1	$c_1 + c_2 + c_3$
1	0	0	0	c_0
1	0	0	1	$c_0 + c_3$
1	0	1	0	$c_0 + c_2$
1	0	1	1	$c_0 + c_2 + c_3$
1	1	0	0	$c_0 + c_1$
1	1	0	1	$c_0 + c_1 + c_3$
1	1	1	0	$c_0 + c_1 + c_2$
1	1	1	1	$c_0 + c_1 + c_2 + c_3$

79

Αρχές Κατανεμημένης Αριθμητικής

- c_i : M -bit σταθερές
- x_i : W -bit δεδομένα συμπληρώματος δύο
- $x_{i,k}$ είναι 0 ή 1

$$\left. \begin{aligned} x_i &= -x_{i,W-1} + \sum_{j=1}^{W-1} x_{i,W-1-j} 2^{-j} \\ Y &= \sum_{i=0}^{N-1} c_i x_i \end{aligned} \right\} \Rightarrow Y = \sum_{i=0}^{N-1} c_i \left(-x_{i,W-1} + \sum_{j=1}^{W-1} x_{i,W-1-j} 2^{-j} \right) = - \sum_{i=0}^{N-1} c_i x_{i,W-1} + \sum_{i=0}^{N-1} \sum_{j=1}^{W-1} c_i x_{i,W-1-j} 2^{-j}$$

$$Y = \sum_{i=0}^{N-1} c_i x_i = \sum_{j=0}^{W-1} C_{W-1-j} 2^{-j}, \quad C_{W-1-j} = \sum_{i=1}^{N-1} c_i x_{i,W-1-j}, \quad j \neq 0 \\ C_{W-1} = - \sum_{i=0}^{N-1} c_i x_{i,W-1}$$

80

Αλγόριθμος Εσωτερικού Γινομένου με Κατανεμημένη Αριθμητική

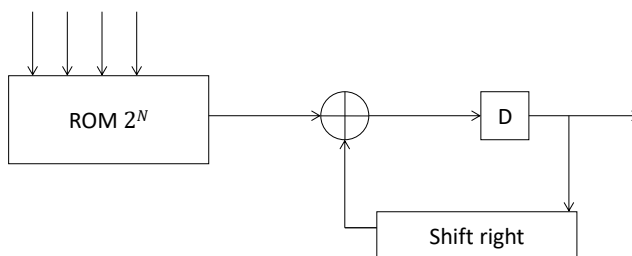
$$Y = \sum_{i=0}^{N-1} c_i x_i = \sum_{j=0}^{W-1} C_{W-1-j} 2^{-j}, \quad \begin{aligned} C_{W-1-j} &= \sum_{i=1}^{N-1} c_i x_{i,W-1-j}, \quad j \neq 0 \\ C_{W-1} &= - \sum_{i=0}^{N-1} c_i x_{i,W-1} \end{aligned}$$

• Παρατηρήσεις

- Οι W σταθερές C_j εξαρτώνται από τα N bits $x_{i,j}$
- Κάθε C_j μπορεί να λάβει 2^N διαφορετικές τιμές.
- Το εσωτερικό γινόμενο Y προκύπτει ως άθροισμα ολισθημένων C_j .

81

Τυπική Αρχιτεκτονική Κατανεμημένης Αριθμητικής

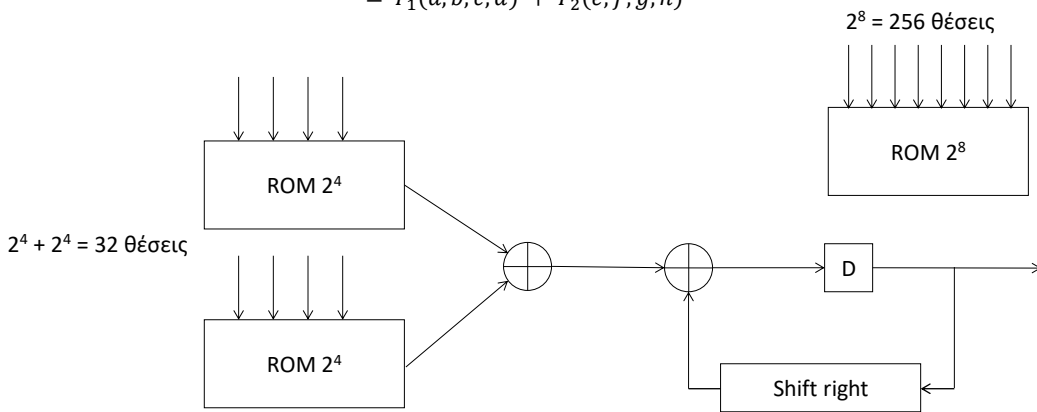


- Σε W επαναλήψεις υπολογίζεται το εσωτερικό γινόμενο, *χωρίς πολλαπλασιασμούς*.

82

Μείωση μεγέθους μνημών σε κατανεμημένη αριθμητική

$$F(a, b, c, d, e, f, g, h) = (a \cdot c_0 + b \cdot c_1 + c \cdot c_2 + d \cdot c_3) + (e \cdot c_4 + f \cdot c_5 + g \cdot c_6 + h \cdot c_7) \\ = F_1(a, b, c, d) + F_2(e, f, g, h)$$



VLSI – II

83