



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση Μετασχηματισμών Wavelet σε Υλικό

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χαράλαμπος Ν. Βλατάκης

Επιβλέπων : Κιαμάλ Ζ. Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2007



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση Μετασχηματισμών Wavelet σε Υλικό

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χαράλαμπος Ν. Βλατάκης

Επιβλέπων : Κιαμάλ Ζ. Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30^η Οκτωβρίου 2007.

.....
Κ. Πεκμεστζή
Καθηγητής Ε.Μ.Π.

.....
Η. Κουκούτσης
Επ. Καθηγητής Ε.Μ.Π.

.....
Γ. Οικονομάκος
Λέκτορας Ε.Μ.Π.

Αθήνα, Οκτώβριος 2007

.....
Χαράλαμπος Ν. Βλατάκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Χαράλαμπος Ν. Βλατάκης 2007.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

στους γονείς μου

Περίληψη

Σκοπός της διπλωματικής αυτής εργασίας είναι η σχεδίαση, η ανάλυση, η προσομοίωση και η σύγκριση εναλλακτικών αρχιτεκτονικών ψηφιακών αριθμητικών κυκλωμάτων που υλοποιούν τον ευθύ διακριτό μετασχηματισμό wavelet του wavelet Cohen-Daubechies-Feauveau (CDF) 9/7.

Αρχικά αναπτύσσουμε περιγραφικά τους ορισμούς και τις βασικές έννοιες των wavelets καθώς και εισάγουμε το wavelet CDF 9/7. Στη συνέχεια παρουσιάζουμε μια σειρά από εναλλακτικές αρχιτεκτονικές υλικού που υλοποιούν το μετασχηματισμό του wavelet αυτού και διερευνώνται μέθοδοι βελτιστοποίησης μέσω μείωσης του εσωτερικού ρυθμού λειτουργίας και δίπλωσης. Οι αρχιτεκτονικές που παρουσιάζουμε βασίζονται στη συνέλιξη, στην b-spline παραγοντοποίηση και στο lifting scheme.

Ακολούθως, επιλέγουμε τις πιο σημαντικές από τις αρχιτεκτονικές αυτές που περιέχουν σταθερούς πολλαπλασιαστές, τις αναπτύσσουμε με στοιχεία διάδοσης κρατουμένου και τις περιγράφουμε ως το επίπεδο πύλης με γλώσσα verilog. Η περιγραφή γίνεται με τη χρήση αυτοματοποιημένων εργαλείων που αναπτύχθηκαν ειδικά για τους σκοπούς της εργασίας. Τέλος, προσομοιώνουμε τα αποτελέσματα της λειτουργίας τους και τις συγκρίνουμε με κριτήρια την ακρίβεια, την ταχύτητα και το μέγεθος του κυκλώματος

Λέξεις-Κλειδιά

Αριθμητικά Κυκλώματα, Αυτοματοποιημένη Σχεδίαση Υλικού, B-Spline Παραγοντοποίηση, Διάδοση Κρατουμένου, Διακριτός Μετασχηματισμός Wavelet, Cohen-Daubechies-Feauveau 9/7, Lifting Scheme, Flipping Structure, Verilog, Φίλτρα Πεπερασμένης Κρουστικής Απόκρισης, Ψηφιακά Φίλτρα, Σταθερός Πολλαπλασιαστής

Abstract

The scope of this thesis is the design, analysis, simulation and comparison of digital arithmetic circuits that perform the forward Discrete Wavelet Transform of the Cohen-Daubechies-Feaveau (CDF) 9/7 wavelet.

Initially the fundamental concepts of wavelets as well as the CDF 9/7 wavelet are presented. Next, a multitude of different hardware architectures that implement the forward CDF 9/7 transform are analyzed in component-level abstraction, and ways of improvement via internal speed reduction and folding are explored. Hardware implementations of convolution, b-spline factorization and lifting scheme algorithms are considered.

Afterwards, the most prominent of the previous architectures that use fixed multipliers are chosen to be analyzed to gate-level using carry-propagate arithmetic. An automated tool, developed specifically for this thesis, creates a description for each circuit using Verilog HDL. Finally, the operation of each architecture is simulated and results on precision, speed and circuit size are compared.

Keywords

Digital Arithmetic, Automated Hardware Design, B-Spline Factorization, Lifting Scheme, Filipping Structure, Carry Propagate, Discrete Wavelet Transform, Cohen-Daubechies-Feauveau 9/7, Digital Filters, FIR Filter, Fixed Multiplier, Verilog

Πίνακας Περιεχομένων

<i>Περίληψη</i>	7
<i>Abstract</i>	9
<i>Πίνακας Περιεχομένων</i>	11
<i>Πίνακας Σχημάτων</i>	13
<i>Πίνακας Πινάκων</i>	15
Κεφάλαιο 1: Εισαγωγή	16
<i>A. Σκοπός και Δομή</i>	16
<i>B. Σήματα και αναπαραστάσεις</i>	16
1. Σήματα και Πληροφορία	16
2. Αναπαράσταση στο χρόνο και τη συχνότητα	17
3. Μια εναλλακτική αναπαράσταση	20
4. Ανάλυση	26
<i>Γ. Εισαγωγή στα Wavelets</i>	27
1. Σύντομη επισκόπηση βασικών εννοιών	27
2. Wavelets και συστοιχίες φίλτρων	32
3. Επιλογή wavelet	35
<i>Δ. Εφαρμογές</i>	38
Κεφάλαιο 2: Αρχιτεκτονικές σε επίπεδο βασικών μονάδων	40
<i>A. Εισαγωγή</i>	40
<i>B. Συμμετρικά FIR φίλτρα με υποδειγματοληψία στην έξοδο</i>	41
1. Εισαγωγή	41
α. Γενικά	41
β. Η έννοια της υποδειγματοληψίας ανα δυο στοιχεία	42
γ. Σύμβολα και αξιώματα κυκλωματικών διαγραμμάτων	43
2. Κανονική μορφή	45
α. Γενικά	45
β. Μείωση του ρυθμού λειτουργίας των στοιχείων του φίλτρου	47
γ. Folding	53
δ. Τεχνικές για περαιτέρω βελτίωση των επιδόσεων	59
3. Transpose μορφή	62
α. Γενικά	62
β. Μείωση του ρυθμού λειτουργίας των στοιχείων του φίλτρου	64
γ. Folding	66
4. Συμπεράσματα	70
<i>Γ. Αρχιτεκτονικές βασιμμένες στη συνέλιξη</i>	72
<i>Δ. Αρχιτεκτονικές βασιμμένες στη b-spline παραγοντοποίηση</i>	76
1. Γενικά	76
2. Κατανεμημένο τμήμα	78
3. B-spline τμήμα	80
4. Συνολικά αποτελέσματα	81
<i>E. Αρχιτεκτονικές βασιμμένες στο Lifting Scheme</i>	82
<i>ΣΤ. Συμπεράσματα</i>	88

Κεφάλαιο 3 Υλοποίηση σε επίπεδο πύλης	90
A. Εισαγωγή	90
B. Αριθμητικά συστήματα	90
1. Εισαγωγή	90
2. Μορφή συμπληρώματος ως προς 2	91
3. Κανονική αναπαράσταση προσημασμένου ψηφίου (<i>csd</i>)	91
4. Μη ακέραιοι αριθμοί: κλιμάκωση και μήκος λέξης	92
Γ. Αρχιτεκτονικές βασικών μονάδων	94
1. Εισαγωγή	94
2. Αθροιστές-Αφαιρέτες	95
α. Πρόσθεση σε επίπεδο bit: Πλήρεις αθροιστές	95
β. Αθροιστής συμπληρώματος ως προς 2	97
γ. Αφαιρέτης συμπληρώματος ως προς 2	98
δ. Κλιμάκωση και μήκος λέξης	99
ε. Κρίσιμο μονοπάτι και αθροιστές	101
3. Πολλαπλασιαστές	102
α. Εισαγωγή	102
β. Σταθεροί πολλαπλασιαστές δυαδικού δένδρου (<i>carry-propagate</i>)	102
γ. Κλιμάκωση και μήκος λέξης σε πολλαπλασιαστές	108
δ. Κρίσιμο μονοπάτι πολλαπλασιαστή	108
4. Στοιχεία καθυστέρησης	109
Δ. Υλοποίηση σε επίπεδο πύλης	110
1. Εισαγωγή	110
2. Περιβάλλον δημιουργίας και προσομοίωσης κυκλωμάτων	111
α. Δημιουργία κώδικα: το πρόγραμμα <i>sfc.c</i>	111
β. Πορεία προσομοίωσης και εξαγωγής αποτελεσμάτων	113
3. Υλοποίηση	116
α. Αρχιτεκτονική βασισμένη στη συνέλιξη, <i>direct</i> μορφή	116
β. Αρχιτεκτονική βασισμένη στη συνέλιξη, <i>transpose</i> μορφή	119
γ. Αρχιτεκτονική βασισμένη στη <i>b-spline</i> παραγοντοποίηση, <i>direct b-spline</i> τμήμα	122
δ. Αρχιτεκτονική βασισμένη στη <i>b-spline</i> παραγοντοποίηση, <i>transpose b-spline</i> τμήμα	125
ε. Αρχιτεκτονική βασισμένη στο <i>lifting scheme</i>	128
4. Συμπεράσματα	131
α. Γενικά	131
β. Σχολιασμός των σφαλμάτων	132
γ. Σχολιασμός των επιδόσεων	136
δ. Συμπεράσματα-Προτάσεις για περαιτέρω έρευνα	137
Παράρτημα Α: Βιβλιογραφία	138

Πίνακας σχημάτων

Σχήμα 1.1 Ένα απλό διακριτό σήμα διακριτού χρόνου, αναπαράσταση στο χρόνο	17
Σχήμα 1.2 Αναπαράσταση του σήματος του σχ. 1.1 στο χώρο της συχνότητας (με χρήση ταχύ μετασχηματισμού Fourier-FFT)	18
Σχήμα 1.3 Το αρχικό σήμα συν ένας ισχυρός μοναδιαίος παλμός	18
Σχήμα 1.4 Εφαρμογή FFT στο σχ. 1.3	19
Σχήμα 1.5 Αρχική εικόνα (αριστερά) και εμφάνιση συνοριακών ασυνεχειών στα άκρα των παραθύρων ανάλυσης (δεξιά) σε υψηλή συμπίεση JPEG	19
Σχήμα 1.6 Επιλογή παραθύρου για το Μετασχηματισμό Fourier Βραχέως Χρόνου (STFT)	20
Σχήμα 1.7 Προσέγγιση της $a(n)$ με την $\varphi(n)$	21
Σχήμα 1.8 Προσέγγιση της $a(n)$ με την $\varphi(2n)$	22
Σχήμα 1.9 Προσέγγιση της $a(n)$ με την $\varphi(4n)$	23
Σχήμα 1.10 Διαδοχικές προσεγγίσεις του $a(n)$ με συνεισφορές από τα $\psi_k(2^m n)$	25
Σχήμα 1.11 Το $\psi(n)$ στο χρόνο και στη συχνότητα (με χρήση FFT)	25
Σχήμα 1.12 Γεωμετρικό ανάλογο μιας ορθογώνιας (a) και μιας διορθογώνιας (b) βάσης	31
Σχήμα 1.13 Υπολογισμός των σημάτων προσέγγισης και λεπτομερειών μέσω μιας συστοιχίας φίλτρων	34
Σχήμα 1.14 Μετασχηματισμός wavelet τριών σταδίων	35
Σχήμα 1.15 Συνάρτηση κλίμακας και wavelet Cohen-Daubechies-Feauveau 9/7	36
Σχήμα 2.1 Direct μορφή γενικού φίλτρου	45
Σχήμα 2.2 Εκμετάλλευση της συμμετρικότητας των συντελεστών	46
Σχήμα 2.3a Διαχωρισμός σε άρτια-περιττά	47
Σχήμα 2.3b Μετακίνηση υποδειγματοληπτών	48
Σχήμα 2.4a Συμμετρία και μείωση ταχύτητας στοιχείων (polyphase decomposition) σε περιττά φίλτρα	49
Σχήμα 2.4b Συμμετρία και polyphase decomposition σε άρτια φίλτρα	52
Σχήμα 2.5 Βασική αρχή δίπλωσης multirate φίλτρου	54
Σχήμα 2.6a Η i-οστή τετράδα του ομαλού μέρους	55
Σχήμα 2.6b Folding της i-οστής τετράδας του ομαλού μέρους	56
Σχήμα 2.7a Υπόλοιπο για $(k \text{ modulo } 4)=0$	57
Σχήμα 2.7b Υπόλοιπο για $(k \text{ modulo } 4)=1$	57
Σχήμα 2.7c Υπόλοιπο για $(k \text{ modulo } 4)=2$	58
Σχήμα 2.7d Υπόλοιπο για $(k \text{ modulo } 4)=3$	58
Σχήμα 2.8a Τυπική αλυσίδα άθροισης γινομένων	59
Σχήμα 2.8b Οργάνωση των αθροιστών σε δένδρο	59
Σχήμα 2.9 Transpose μορφή γενικού φίλτρου	62
Σχήμα 2.10 Εκμετάλλευση συμμετρίας στην transpose μορφή	63
Σχήμα 2.11a Διαχωρισμός αρτίων-περιττών	65
Σχήμα 2.11b Συμμετρία και polyphase decomposition σε transpose μορφή	66
Σχήμα 2.12 Ισοδύναμες μορφές folding του i-οστού ζεγαριού	67
Σχήμα 2.13 Διακριτός Μετασχηματισμός Wavelet	72
Σχήμα 2.14 Direct Half-Rate του CDF 9/7	74
Σχήμα 2.15 Direct Folded του CDF 9/7	75
Σχήμα 2.16 Transpose Half-Rate του CDF 9/7	75
Σχήμα 2.17 Transpose Folded του CDF 9/7	76
Σχήμα 2.18 Αρχή b-spline παραγοντοποίησης	77
Σχήμα 2.19 Direct Half-Rate του κατανεμημένου τμήματος του CDF 9/7	79
Σχήμα 2.20 Direct Folded του κατανεμημένου τμήματος του CDF 9/7	79
Σχήμα 2.21 Transpose Half-Rate του κατανεμημένου τμήματος του CDF 9/7	80
Σχήμα 2.22 Transpose Folded του κατανεμημένου τμήματος του CDF 9/7	80
Σχήμα 2.23 Direct υλοποίηση του b-spline τμήματος του CDF 9/7	81
Σχήμα 2.24 Transpose υλοποίηση του b-spline τμήματος του CDF 9/7	81
Σχήμα 2.25 Γενική αρχιτεκτονική lifting scheme	84
Σχήμα 2.26 Lifting υλοποίηση του CDF 9/7	84
Σχήμα 2.27 Lifting υλοποίηση του CDF 9/7 + pipelining	85
Σχήμα 2.28a Αρχικό κύκλωμα	86
Σχήμα 2.28b Αναδιάταξη γράφου σηματοροής και καθορισμός «περιοχών διαίρεσης»	86

Σχήμα 2.28c Πολλαπλασιασμός με τους ανάστροφους συντελεστές, μετακίνηση καταχωρητών και συγχώνευση διαδοχικών πολλαπλασιαστών	86
Σχήμα 2.28d Αλλαγή σειράς προσθέσεων και μετακίνηση των καταχωρητών από τους προσθετέους στο αποτέλεσμα	86
Σχήμα 2.28e Τελική βελτιωμένη <i>lifting-based, flipping structure</i> αρχιτεκτονική	87
Σχήμα 2.29 <i>Folding</i> της αποδοτικότερης <i>lifting</i> αρχιτεκτονικής	88
Σχήμα 3.1 Μορφή συμπληρώματος ως προς 2	91
Σχήμα 3.2 Δυαδική αναπαράσταση προσημασμένου ρητού αριθμού σε μορφή συμπληρώματος ως προς 2	93
Σχήμα 3.3 Εσωτερική δομή πλήρη αθροιστή	96
Σχήμα 3.4 Κυκλωματικό σύμβολο πλήρη αθροιστή με θετικές εισόδους	96
Σχήμα 3.5 Κυκλωματικό σύμβολο πλήρη αθροιστή με δύο αρνητικές εισόδους	97
Σχήμα 3.6 Παράλληλος αθροιστής αριθμών σε μορφή συμπληρώματος ως προς 2	97
Σχήμα 3.7 Παράλληλος αφαιρετής αριθμών σε μορφή συμπληρώματος ως προς 2 ($Y-X$)	99
Σχήμα 3.8 Άθροιση τριών αριθμών σε μορφή συμπληρώματος ως προς 2 με αθροιστές διάδοσης κρατούμενου	101
Σχήμα 3.9 Απλή υλοποίηση σταθερού πολλαπλασιαστή με χρήση αθροιστών/αφαιρετών σταθερού μήκους	103
Σχήμα 3.10 Βελτίωση της ακρίβειας	104
Σχήμα 3.11 Αρχικό δένδρο πολλαπλασιαστή (ο συντελεστής είναι ο αριθμός 0.922423 σε μορφή <i>csd</i> και κλιμάκωση 0)	105
Σχήμα 3.12 Εφαρμογή του αλγορίθμου βέλτιστου δένδρου πολλαπλασιαστή στο σχ. 3.11	108
Σχήμα 3.13 Ένα απλό δικτύωμα	111
Σχήμα 3.14 Διάγραμμα του περιβάλλοντος δημιουργίας, προσομοίωσης και ελέγχου των ψηφιακών φίλτρων που υλοποιούν το μετασχηματισμό <i>wavelet CDF 9/7</i>	115
Σχήμα 3.15 Γράφος του <i>Direct Half-Rate</i> του <i>CDF 9/7</i>	116
Σχήμα 3.16 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στη συνέλιξη, <i>direct</i> μορφή	118
Σχήμα 3.17 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στη συνέλιξη, <i>direct</i> μορφή	118
Σχήμα 3.18 Γράφος του <i>Transpose Half-Rate</i> του <i>CDF 9/7</i>	119
Σχήμα 3.19 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στη συνέλιξη, <i>transpose</i> μορφή	121
Σχήμα 3.20 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στη συνέλιξη, <i>transpose</i> μορφή	121
Σχήμα 3.21 Γράφος της <i>Direct</i> υλοποίησης του <i>b-spline</i> τμήματος του <i>CDF9/7</i>	122
Σχήμα 3.22 Γράφος του βέλτιστου κατανεμημένου τμήματος (<i>transpose</i> μορφή) του <i>CDF 9/7</i>	123
Σχήμα 3.23 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στην <i>b-spline</i> παραγοντοποίηση, με <i>direct b-spline part</i>	124
Σχήμα 3.24 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στην <i>b-spline</i> παραγοντοποίηση, με <i>direct b-spline part</i>	125
Σχήμα 3.25 Γράφος της <i>Transpose</i> υλοποίησης του <i>b-spline</i> τμήματος του <i>CDF9/7</i>	125
Σχήμα 3.26 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στην <i>b-spline</i> παραγοντοποίηση, με <i>transpose b-spline part</i>	127
Σχήμα 3.27 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στην <i>b-spline</i> παραγοντοποίηση, με <i>transpose b-spline part</i>	127
Σχήμα 3.28a Καθορισμός «περιοχών διαίρεσης»	128
Σχήμα 3.28b Εκτέλεση πράξεων, συγχώνευση πολλαπλασιαστών, μετακίνηση καταχωρητών	128
Σχήμα 3.28c Τελική μορφή με λιγότερους καταχωρητές και μικρότερο <i>critical path</i> με αλλαγή της σειράς των αθροίσεων	129
Σχήμα 3.29 Γράφος της αρχιτεκτονικής βασισμένης στο <i>Lifting Scheme</i>	129
Σχήμα 3.30 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στο <i>lifting scheme</i>	130
Σχήμα 3.31 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στο <i>lifting scheme</i>	131

Πίνακας Πινάκων

Πίνακας 1.1 Σταδιακή αποδόμηση του σήματος <i>leleccum</i> με χρήση του μετασχηματισμού <i>wavelet CDF9/7</i> , τα πρώτα πέντε στάδια	37
Πίνακας 2.1 Σύμβολα κυκλωματικών διαγραμμάτων	43
Πίνακας 2.2 Βασικά αξιώματα κυκλωματικών διαγραμμάτων	44
Πίνακας 2.3 Σύγκριση αρχιτεκτονικών για περιττό φίλτρο k στοιχείων	70
Πίνακας 2.4 Σύγκριση αρχιτεκτονικών για ζευγάρι περιττών φίλτρων t και u σημείων	71
Πίνακας 2.5 Σύγκριση αρχιτεκτονικών βασισμένων στη συνέλιξη του <i>CDF 9/7</i>	73
Πίνακας 2.6 Σύγκριση αρχιτεκτονικών για συμμετρικό φίλτρο 3 σημείων	78
Πίνακας 2.7 Σύγκριση αρχιτεκτονικών για συμμετρικό φίλτρο 5 σημείων	78
Πίνακας 2.8 Σύγκριση αρχιτεκτονικών του κατανεμημένου τμήματος του <i>CDF 9/7</i>	78
Πίνακας 2.9 Σύγκριση <i>b-spline-based</i> αρχιτεκτονικών για το <i>CDF 9/7</i>	82
Πίνακας 2.10 Σύγκριση αρχιτεκτονικών που πραγματοποιούν τον ευθύ μετασχηματισμό <i>wavelet CDF 9/7</i>	89
Πίνακας 3.1 Πίνακας αληθείας πλήρη αθροιστή	95
Πίνακας 3.2 Πίνακας αληθείας αθροιστή δύο αρνητικών και ενός θετικού <i>bit</i>	97
Πίνακας 3.3 Σύγκριση απόλυτου σφάλματος σήματος προσέγγισης όλων των κυκλωμάτων	134
Πίνακας 3.4 Σύγκριση απόλυτου σφάλματος σήματος λεπτομερειών όλων των κυκλωμάτων	135
Πίνακας 3.5 Σύγκριση επιδόσεων σε υλικό, ταχύτητα και ακρίβεια	136

Κεφάλαιο 1

Εισαγωγή

A. Σκοπός και δομή

Σκοπός της διπλωματικής αυτής εργασίας είναι η μελέτη και η ανάπτυξη ως το επίπεδο πύλης και bit εναλλακτικών αριθμητικών κυκλωμάτων που πραγματοποιούν τον μονοδιάστατο ευθύ διακριτό μετασχηματισμό wavelet σχετιζόμενο με το wavelet Cohen-Daubechies-Feauveau (CDF) 9/7.

Η πορεία που θα ακολουθήσουμε είναι η εξής: στο εισαγωγικό αυτό κεφάλαιο θα κάνουμε μια σύντομη εισαγωγή στα wavelets και θα δείξουμε τις περιπτώσεις στις οποίες μας είναι απαραίτητα. Ταυτόχρονα, θα συνάγουμε (με μαθηματική χαλαρότητα) κάποιες σημαντικές έννοιες που τα διέπουν, και οι οποίες είναι αναγκαίες για τη συνέχεια της εργασίας. Επιπλέον, θα αναφερθούμε στις κυριότερες ως τώρα εφαρμογές τους και τη σημασία υλοποίησής τους σε υλικό.

Στο επόμενο κεφάλαιο θα μελετήσουμε σε *επίπεδο βασικών μονάδων* μια σειρά από εναλλακτικά αριθμητικά κυκλώματα που υλοποιούν το μετασχηματισμό του wavelet CDF 9/7. Επίπεδο βασικών μονάδων σημαίνει ότι θα αρκεστούμε στην αναπαράσταση των πολλαπλασιαστών, των αθροιστών, των καταχωρητών και των υπολοίπων στοιχείων των δικτυωμάτων ως «μαύρα κουτιά», και επιπλέον ασχολούμαστε με το ποιο εξάρτημα τροφοδοτεί (με δεδομένα) ποιο, και όχι τη μορφή αυτών των δεδομένων, αν είναι σε μορφή συμπληρώματος ως προς 2, carry-save κ.ο.κ. Με αυτήν την αφαίρεση γίνεται αρκετά ευκολότερο να επικεντρωθούμε σε τρόπους μείωσης του *αριθμού των βασικών μονάδων* και συνεπώς και του απαιτούμενου όγκου της υλοποίησης

Τέλος, θα επιλέξουμε (με κριτήρια που θα αναλυθούν στο επόμενο κεφάλαιο) μια σειρά από τις καλύτερες αρχιτεκτονικές που θα αναδειχθούν από το κεφάλαιο 2, και θα τις αναπτύξουμε στο τελευταίο κεφάλαιο της εργασίας ως το επίπεδο πύλης/bit. Αυτό σημαίνει ότι με δεδομένα την τοπολογία και το πλήθος των βασικών μονάδων, θα προσπαθήσουμε να ανακαλύψουμε τη *βέλτιστη* μορφή με την οποία αυτά υλοποιούνται. Δηλαδή, ενώ στο 2^ο κεφάλαιο η βελτιστοποίηση θα γίνει σε αρχιτεκτονικό επίπεδο, στο 3^ο κεφάλαιο η βελτιστοποίηση θα γίνει σε επίπεδο βασικών μονάδων. Ένα πλεονέκτημα της υλοποίησης σε επίπεδο πύλης είναι ότι με αυτόν τον τρόπο θα μπορούμε να συνάγουμε πολύ εύκολα συνθέσιμο κώδικα Verilog για το κάθε κύκλωμα, καθώς και άμεσα συγκρίσιμα αποτελέσματα για την ταχύτητα, τον όγκο και την ακρίβειά του. Έτσι, με σύγκριση των διαφόρων αρχιτεκτονικών που θα υλοποιήσουμε ως το επίπεδο bit, θα αποφασίσουμε για τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μίας και θα καταλήξουμε στα τελικά συμπεράσματα της εργασίας αυτής

B. Σήματα και αναπαραστάσεις

1. Σήματα και πληροφορία

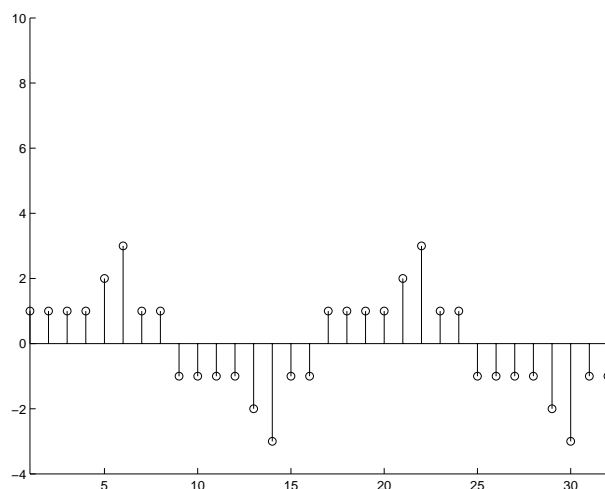
Στην ενότητα αυτή θα κάνουμε μια σύντομη, περισσότερο ποιοτική, εισαγωγή σε βασικές έννοιες σημάτων, με σκοπό κυρίως την αυτάρκεια του κειμένου της

εργασίας. Ο ενδιαφερόμενος αναγνώστης μπορεί να βρει περισσότερες, και πιο σωστά θεμελιωμένες, πληροφορίες σε ένα εισαγωγικό βιβλίο σημάτων και συστημάτων, όπως το [A1]

Τόσο ο φυσικός όσο και ο τεχνητός κόσμος αποτελούνται από συστήματα, τα οποία επικοινωνούν μεταξύ τους μέσω σημάτων. Ένα σήμα μπορεί να είναι συνεχούς (π.χ. ηλεκτρική τάση) ή διακριτού χρόνου (π.χ. η έξοδος ενός ψηφιακού κυκλώματος) και να λαμβάνει συνεχείς ή διακριτές τιμές. Επίσης, μπορεί να είναι μιας διάστασης (π.χ. ήχος, τάση ρευματοδότη, καρδιογράφημα), δύο διαστάσεων (π.χ. εικόνα), τριών (π.χ. βίντεο) ή και περισσότερων. Εμείς εδώ θα ασχοληθούμε με σήματα μιας διάστασης, καθώς τα συμπεράσματα (αλλά και οι μετασχηματισμοί) που προκύπτουν μπορούν εύκολα να μεταφερθούν σε περισσότερες διαστάσεις. Θα θεωρήσουμε ότι η μια αυτή διάσταση είναι ο **χρόνος**. Στην πραγματικότητα, αυτό γίνεται για λόγους διευκόλυνσης και μόνο, καθώς υπάρχει μια πληθώρα σημάτων με μονοδιάστατο πεδίο ορισμού (π.χ. η τάση κατά μήκος ενός αγωγού, μια χρονική στιγμή) που δεν ορίζονται με βάση το χρόνο.

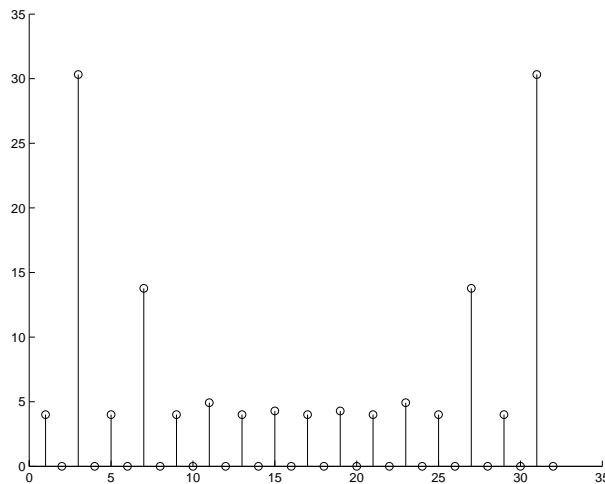
2. Αναπαράσταση στο χρόνο και τη συχνότητα

Η πιο συνηθισμένη αναπαράσταση ενός σήματος είναι η αναπαράστασή του στο χρόνο, γνωστή και ως αναπαράσταση πλάτους-χρόνου. Αυτή προκύπτει άμεσα από τις τιμές του σήματος και μας δίνει μια άμεση πληροφορία για τη χρονική του εξέλιξη.



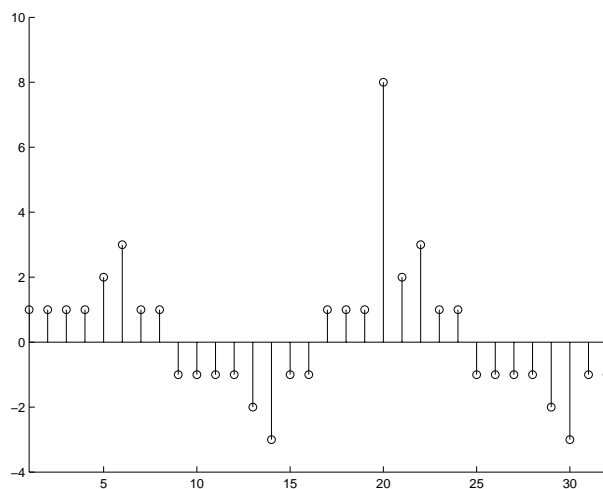
Σχήμα 1.1 Ένα απλό διακριτό σήμα διακριτού χρόνου, αναπαράσταση στο χρόνο

Ωστόσο, είναι αρκετές οι περιπτώσεις που η κύρια πληροφορία για το σύστημα που παράγει το σήμα, ή εν γένει οι κύριες πληροφορίες που μας ενδιαφέρουν, δεν προέρχονται από τη μορφή του σήματος στο χρόνο, αλλά από τη μορφή του σήματος στο χώρο της συχνότητας. Ένα πολύ διαδεδομένο εργαλείο που αναπαριστά σχεδόν οποιοδήποτε σήμα από το χώρο του χρόνου στο χώρο της συχνότητας είναι ο μετασχηματισμός Fourier και οι «συγγενείς» μετασχηματισμοί συνημιτόνου (DCT) ή τροποποιημένου συνημιτόνου (MDCT). Για παράδειγμα, το σήμα του σχ. 1.1 έχει την εξής μορφή στο χώρο της συχνότητας:



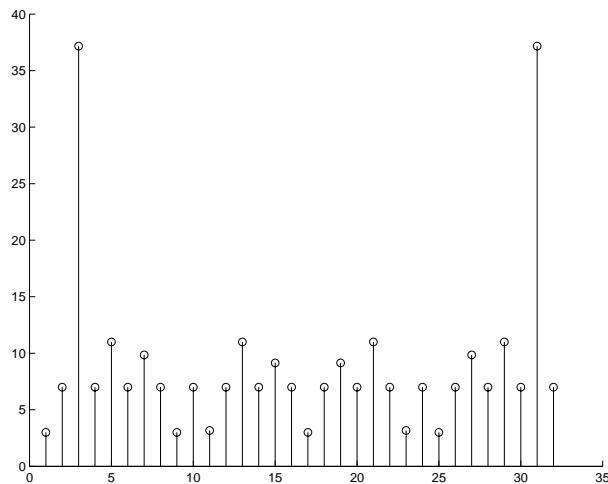
Σχήμα 1.2 Αναπαράσταση του σήματος του σχ. 1.1 στο χώρο της συχνότητας (με χρήση ταχύ μετασχηματισμού Fourier-FFT)

Παρατηρείστε ότι το σχ. 1.2 μας δείχνει *ποιές* συχνότητες απαντώνται στο αρχικό σήμα, αλλά δε μας δίνει καμία πληροφορία για το *που* στο σήμα μπορούμε να τις βρούμε. Για να κάνουμε τα πράγματα χειρότερα, θα προσθέσουμε ένα ισχυρό μοναδιαίο παλμό στο αρχικό μας σήμα:



Σχήμα 1.3 Το αρχικό σήμα συν ένας ισχυρός μοναδιαίος παλμός

Ο παλμός που προστέθηκε στη χρονική στιγμή $t=20$, προφανώς θα αλλάξει αρκετά το φασματικό περιεχόμενο του σήματος, καθώς λόγω της απότομης ανόδου/καθόδου που εισάγει, θα προκαλέσει μια σειρά από καινούριες αρμονικές. Πράγματι, ο γρήγορος μετασχηματισμός Fourier μας δίνει:



Σχήμα 1.4 Εφαρμογή FFT στο σχ. 1.3

Αν το σχ. 1.2 έδινε το περιεχόμενο συχνότητας ενός σήματος που παράγεται κατά την κανονική λειτουργία ενός μηχανήματος, και το σχ. 1.4 κατά την ανώμαλη λειτουργία, συγκρίνοντάς τα βλέπουμε ότι στη δεύτερη περίπτωση υπήρξε κάποιο απότομο μεταβατικό φαινόμενο. Δεν μπορούμε όμως να διακρίνουμε *πότε* αυτό έλαβε χώρα. Επίσης, παρατηρούμε ότι παρά την ένταση του παλμού στο σχ. 1.3, οι φασματικές συνιστώσες που αντιστοιχούν σε αυτόν είναι σχετικά μικρές, καθώς η ενέργειά του έχει διαμοιραστεί σε όλο το μήκος του σήματος.

Η ικανότητα συσχέτισης ενός φαινομένου με το χρόνο είναι πολύ σημαντική, συνεπώς πρέπει να βρεθεί κάποιος τρόπος να είναι δυνατό να γνωρίζουμε *ταυτόχρονα* το περιεχόμενο ενός σήματος στο χώρο της συχνότητας και τη χρονική στιγμή που αυτό μεταβάλλεται. Μια λύση στο πρόβλημα έχει δώσει ο μετασχηματισμός fourier βραχέως χρόνου: χωρίζουμε το αρχικό σήμα σε μια σειρά από μικρότερα σήματα ίσου μήκους, στο κάθε ένα από τα οποία κάνουμε μετασχηματισμό fourier. Το μέγεθος καθενός από αυτά τα μικρότερα σήματα λέγεται *παραθύρο ανάλυσης*.

Ωστόσο, αυτή η τεχνική του διαχωρισμού του αρχικού σήματος σε μικρότερα δημιουργεί τεχνητές *συνοριακές συνθήκες* στα άκρα κάθε ενός από τα παραθύρα ανάλυσης και επιπλέον, λόγω του ότι το καθένα από τα δύο στοιχεία εκατέρωθεν του συνόρου δέχεται ξεχωριστή επεξεργασία, είναι πολύ πιθανό, αν επιχειρηθεί ανασύνθεση

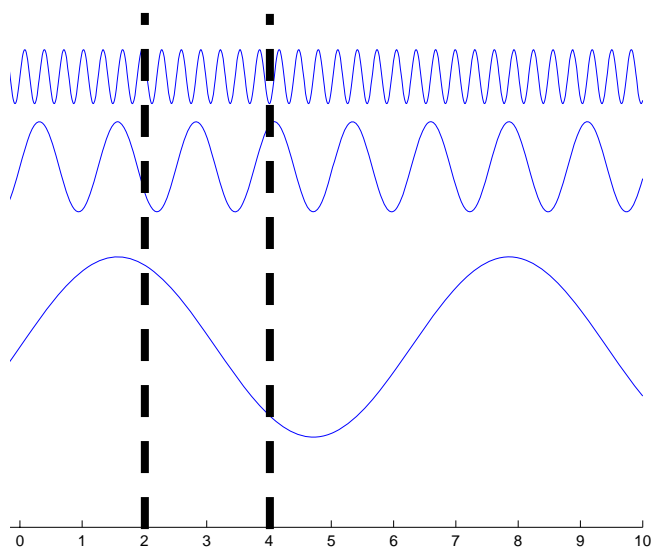


Σχήμα 1.5 Αρχική εικόνα (αριστερά) και εμφάνιση συνοριακών ασυνεχειών στα άκρα των παραθύρων ανάλυσης (δεξιά) σε υψηλή συμπίεση JPEG

του αρχικού σήματος μέσω των αποτελεσμάτων του μετασχηματισμού fourier βραχέως χρόνου να δημιουργηθούν τεχνητές ασυνέχειες στα σημεία τομής. Είναι για

παράδειγμα πολύ γνωστές οι ασυνέχειες που εμφανίζει σε μια εικόνα υπό υψηλή συμπίεση ο αλγόριθμος JPEG (ο οποίος χρησιμοποιεί το (διακριτό) μετασχηματισμό συνημιτόνου σε ομάδες 8x8 pixel, πολύ όμοιο με το μετασχηματισμό fourier).

Το πρόβλημα των τεχνητών συνόρων μπορεί να λυθεί με διάφορους τρόπους, για παράδειγμα με επιλογή παραθύρων που αλληλοεπικαλύπτονται και απόρριψη των οριακών σημείων κατά την ανασύνθεση. Αυτό το τέχνασμα χρησιμοποιείται από τον τροποποιημένο διακριτό μετασχηματισμό συνημιτόνου (MDCT-Modified Discrete Cosine Transform) που χρησιμοποιούν οι αλγόριθμοι συμπίεσης ήχου MP3, AAC και AC3. Ωστόσο, ένα πρόβλημα που παραμένει είναι ότι και πάλι, η επιλογή του μεγέθους του παραθύρου αποτελεί έναν συμβιβασμό.



Σχήμα 1.6 Επιλογή παραθύρου για το Μετασχηματισμό Fourier Βραχέως Χρόνου (STFT)

Για παράδειγμα, βλέπουμε στο σχ. 1.5 ότι το μέγεθος του παραθύρου που επιλέξαμε είναι αρκετά καλό για τη μεσαία ημιτονική συνάρτηση. Ωστόσο, το παράθυρο είναι πολύ μικρό σε μήκος για να μπορούμε να έχουμε πληροφορία για μικρότερες συχνότητες (αφού η περιόδός τους ολοκληρώνεται σε πολλά μήκη του) ενώ για τις υψηλότερες συχνότητες, εμφανίζεται πάλι το πρόβλημα της αδυναμίας εντοπισμού.

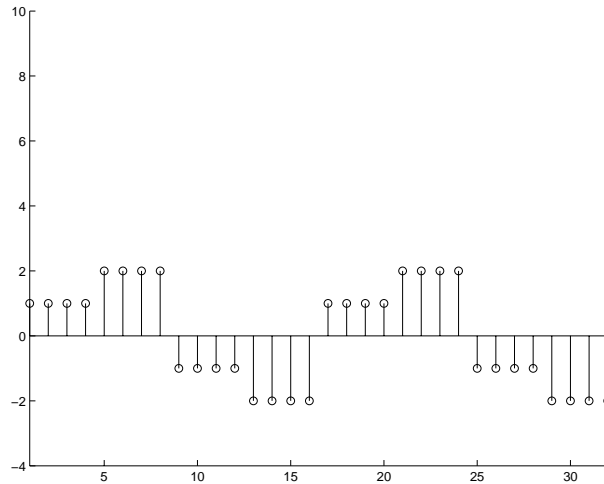
Το τελευταίο πρόβλημα είναι επακόλουθο μιας πολύ γενικότερης αρχής που ονομάζεται **αρχή της αβεβαιότητας**. Η αρχή αυτή μας λέει ότι είναι αδύνατο να έχουμε ταυτόχρονα πλήρη πληροφορία και στο πεδίο του χρόνου και στο πεδίο της συχνότητας. Αν έχουμε όλη την πληροφορία στο πεδίο του χρόνου (σχ. 1.1, 1.3) δεν έχουμε καμία πληροφορία για το πεδίο της συχνότητας. Αντίθετα, αν έχουμε ακριβείς πληροφορίες για το σύνολο του χώρου της συχνότητας (σχ. 1.2, 1.4) δεν έχουμε καμία πληροφορία για το πεδίο του χρόνου. Θα συναντήσουμε την αρχή της αβεβαιότητας και παρακάτω όταν θα αναφερθούμε στον εντοπισμό των συναρτήσεων βάσης.

3. Μια εναλλακτική αναπαράσταση

Είδαμε πριν ότι οι κλασσικοί μετασχηματισμοί σημάτων (fourier, συνημιτόνου) ενώ μας δείχνουν ποιες αρμονικές υπάρχουν σε ένα σήμα, δε μας δείχνουν πότε αυτές εμφανίζονται. Επομένως, είναι ακατάλληλοι για σήματα με

$$\phi_k(2n) = \phi(2n - 8 * k) = \phi(2(n - 4 * k)) \quad (1.5)$$

Όπως πριν, το $a(n)$ μπορεί να αναπαρασταθεί προσεγγιστικά, αλλά με περισσότερες λεπτομέρειες ως γραμμικός συνδυασμός των $\phi_{2k}(n)$:



Σχήμα 1.8 Προσέγγιση της $a(n)$ με την $\phi(2n)$

Οπότε, ισχύει για την $a(n)$ σε σχέση με την $\phi(2n)$:

$$a(n) \approx (1, 2, -1, -2, 1, 2, -1, -2) \quad (1.6)$$

Συνεχίζοντας παρόμοια, έχουμε για την $\phi(4n)$:

$$\phi(4n) = \begin{cases} 0, n < 0 \\ 1, 4n(\text{modulo})8 = 0 \Leftrightarrow n(\text{modulo})2 = 0 \\ 0, \text{αλλού} \end{cases} \quad (1.7)$$

και επίσης, από την (1.2):

$$\phi_k(4n) = \phi(4n - 8 * k) = \phi(4(n - 2 * k)) \quad (1.8)$$

Οπότε, και το $a(n)$ προσεγγίζεται ως εξής:

$$\psi(n) = \begin{cases} 0, n < 0 \\ 1, n \pmod{4} = 0 \\ -1, (n-4) \pmod{4} = 0 \\ 0, \text{αλλού} \end{cases} \quad (1.12)$$

και κατ' αναλογία με πριν, ορίζουμε ότι:

$$\psi_k(n) = \psi(n - 8 * k) \quad (1.13)$$

Τότε, μπορούμε εύκολα να δούμε ότι το $\{\psi_k(n)\} \cup \{\phi_k(n)\}$ έχουν το ίδιο σύνολο επέκτασης με το $\phi_k(2n)$. Για παράδειγμα, το σήμα του σχήματος 1.7 μπορεί να γραφεί ως:

$$a(n) = 1.5\phi_0(n) - 1.5\phi_1(n) + 1.5\phi_2(n) - 1.5\phi_3(n) - 0.5\psi_0(n) + 0.5\psi_1(n) - 0.5\psi_2(n) + 0.5\psi_3(n) \quad (1.14)$$

Κατά παρόμοιο τρόπο, μπορούμε να ορίσουμε τα $\psi_k(2n)$ για τα $\phi_k(2n)$ και τα $\psi_k(4n)$ για τα $\phi_k(4n)$. Συνεπώς, αφού το $a(n)$ μπορεί να αναπαρασταθεί επακριβώς από το $\{\phi_k(8n)\}$, θα μπορεί να αναπαρασταθεί από το $\{\psi_k(4n)\} \cup \{\phi_k(4n)\}$, αλλά και το $\{\phi_k(4n)\}$ έχει το ίδιο σύνολο επέκτασης με το $\{\psi_k(2n)\} \cup \{\phi_k(2n)\}$ κ.ο.κ, άρα το $a(n)$ μπορεί να αναπαρασταθεί επακριβώς και με μοναδικό τρόπο ως γραμμικός συνδυασμός των στοιχείων της βάσης

$$\{\psi_k(4n)\} \cup \{\psi_k(2n)\} \cup \{\psi_k(n)\} \cup \{\phi_k(n)\} \quad (1.15)$$

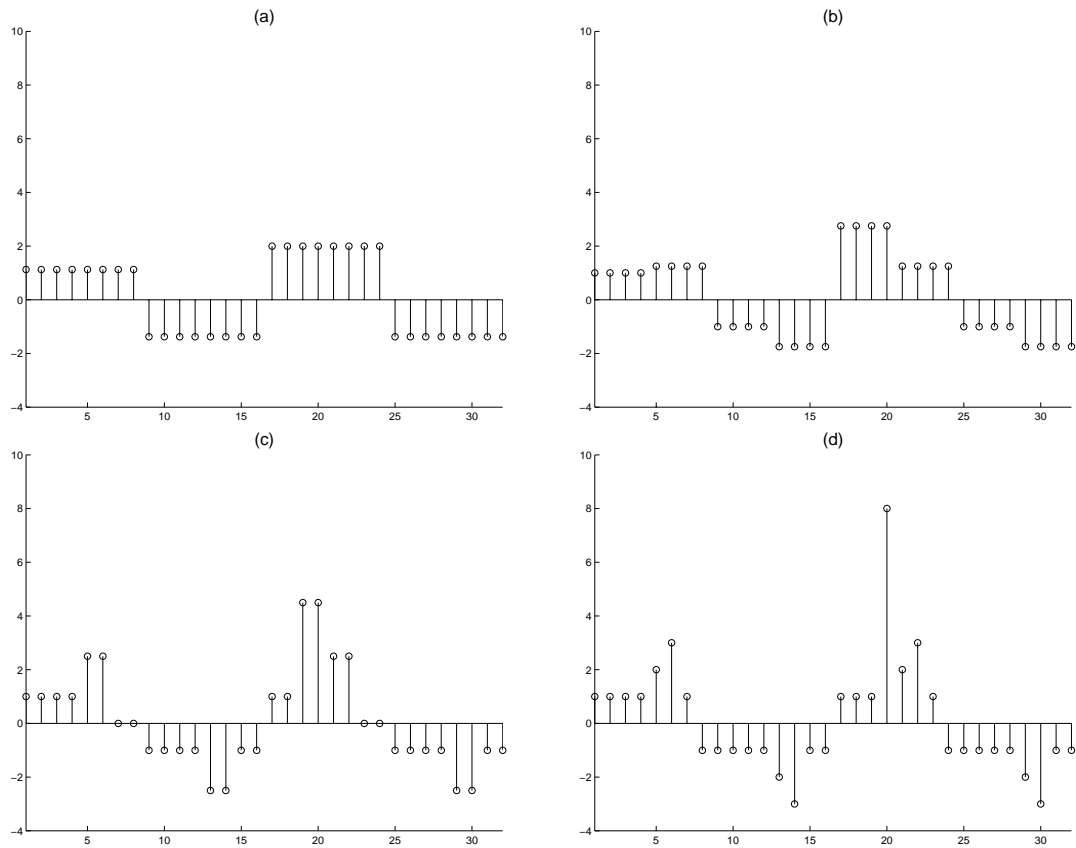
Κάθε ένα από τα σύνολα που αποτελούν τη βάση στην (1.15) αντιστοιχεί (και μόνο αυτό) σε ένα διαφορετικό επίπεδο ανάλυσης, και συνεπώς σε κάποιες υψηλότερες συχνότητες. Για παράδειγμα, το $a(n)$ μπορεί να γραφεί ως εξής:

- Συντελεστές για τα $\phi_k(n)$ (σχ. 1.9a): (1.125, -1.375, 2, -1.375) (1.16a)

- Συντελεστές για τα $\psi_k(n)$ (σχ. 1.9b): (-0.125, 0.375, 0.75, 0.375) (1.16b)

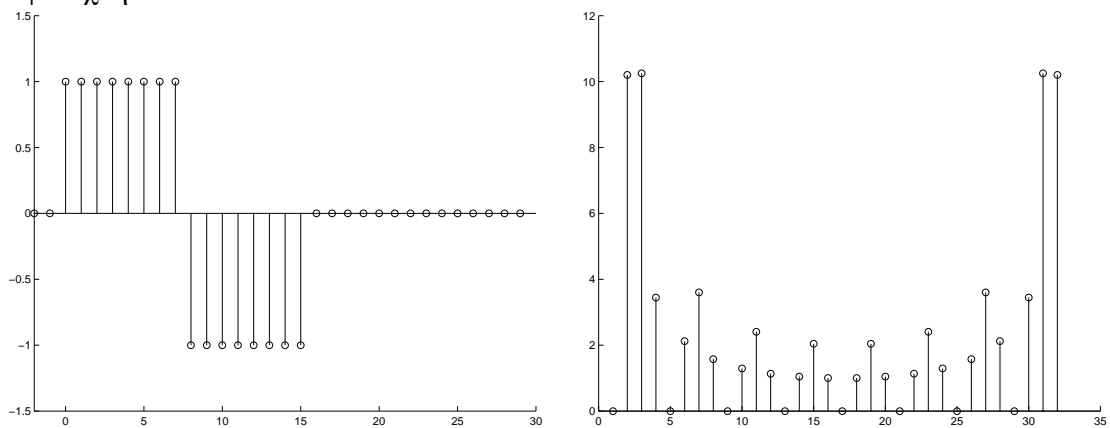
- Συντελεστές για τα $\psi_k(2n)$ (σχ. 1.9c): (0, 1.25, 0, -0.75, -1.75, 1.25, 0, -0.75) (1.16c)

- Συντελεστές για τα $\psi_k(4n)$ (σχ. 1.9d): (0, 0, -0.5, 1, 0, 0, 0.5, 0, 0, -3.5, -0.5, 1, 0, 0, 0.5, 0) (1.16d)



Σχήμα 1.10 Διαδοχικές προσεγγίσεις του $a(n)$ με συνεισφορές από τα $\psi_k(2^m n)$

Μια άμεση παρατήρηση είναι ότι λόγω όλων των προηγούμενων, τα διανύσματα στις (1.16) περιέχουν πληροφορίες για το περιεχόμενο συχνότητας του $a(n)$ ως το συγκεκριμένο επίπεδο λεπτομερειών: για παράδειγμα, στην (1.16a) μας δίνονται οι «θεμελιώδεις» ταλαντώσεις του σήματος, και **ταυτόχρονα** στην (1.16d) βλέπουμε ότι κάπου ανάμεσα στις χρονικές στιγμές 19 και 20 υπήρξε μια ισχυρή υψίσυχη συνιστώσα.



Σχήμα 1.11 Το $\psi(n)$ στο χρόνο και στη συχνότητα (με χρήση FFT)

Επίσης, ο συνολικός αριθμός των συντελεστών των (1.16) είναι 32, δηλαδή, όσο και ο αριθμός των δειγμάτων του σήματος, ή των δειγμάτων στο χώρο της συχνότητας στην αναπαράσταση που μας δίνει ο μετασχηματισμός fourier. Έχουμε, δηλαδή, μια **συμπαγή** αναπαράσταση του σήματος στην οποία έχουμε πληροφορία **και** για το

φασματικό του περιεχόμενο **και** για τις χρονικές στιγμές στις οποίες αυτό μεταβάλλεται. Επιτύχαμε, λοιπόν, μια αναπαράσταση **χρόνου-συχνότητας**.

Παρατηρήστε στο σχ. 1.10 ότι το φάσμα του $\psi(t)$ δεν είναι συγκεντρωμένο σε δύο σημεία, όπως του ημιτόνου, αλλά παρουσιάζει μεγαλύτερη διασπορά. Έτσι, η αναπαράσταση χρόνου-συχνότητας που πετύχαμε παρουσιάζει αβεβαιότητα στη συχνότητα (δηλαδή, δεν είναι δυνατό να αντιστοιχίσουμε π.χ. τους συντελεστές της (1.16d) σε μια συγκεκριμένη συχνότητα, όπως κάνει ο μετασχηματισμός fourier) αλλά και στο χρόνο (π.χ. στο 1.16d δεν μπορούμε να κρίνουμε αν η υψίσυχνη συνιστώσα παρουσιάζεται για $n=19$ ή για $n=20$). Αυτό δεν είναι παρά φυσική συνέπεια της αρχής της απροσδιοριστίας που αναφέραμε πριν. Ωστόσο, αυτή η αβεβαιότητα είναι πολύ μικρή για τη συγκεκριμένη εφαρμογή, και δε μας απασχολεί.

4. Ανάλυση

Είδαμε πριν πως μια αναπαράσταση του σήματος χρησιμοποιώντας ως ακολουθίες βάσης μια συγκεκριμένη ακολουθία και τις χρονικές της συμπίεσεις μπορεί και αποδίδει σωστά και με σαφήνεια τα χαρακτηριστικά σημάτων με απότομες μεταβολές ενώ παράλληλα δίδει και πληροφορία ταυτόχρονα στα πεδία χρόνου και συχνότητας, δύο ρόλοι στους οποίους ο μετασχηματισμός fourier αποτυγχάνει να ανταποκριθεί σωστά. Ο λόγος για τον οποίον αυτό συμβαίνει έχει να κάνει (και μόνο) με τις συναρτήσεις-ακολουθίες βάσης που χρησιμοποιεί ο κάθε μετασχηματισμός:

- Ο μετασχηματισμός fourier και οι «συγγενικοί» σε αυτόν μετασχηματισμοί (συνημίτονου, διακριτοί κλπ) χρησιμοποιεί ως συναρτήσεις βάσης τις τριγωνομετρικές συναρτήσεις. Οι συναρτήσεις αυτές είναι *εντοπισμένες στη συχνότητα* (δηλαδή είναι παντού μηδέν εκτός από ένα φραγμένο διάστημα) αλλά δεν είναι *εντοπισμένες στο χρόνο*. Έτσι, αν αναλύσουμε ένα σήμα με βάση αυτές παίρνουμε *ακριβή* πληροφορία στη συχνότητα και *καθόλου* πληροφορία στο χρόνο
- Από την άλλη, ο μετασχηματισμός που κάναμε στο $a(n)$ πριν έχει ως ακολουθίες βάσης την $\varphi(n)$ και τις $\psi_k(n)$ οι οποίες είναι *εντοπισμένες στο χρόνο* αλλά δεν είναι *εντοπισμένες στη συχνότητα*. Ωστόσο, σε αντίθεση με το ημίτονο που έχει την ίδια ενέργεια σε κάθε χρονική στιγμή, οι $\psi_k(n)$ έχουν την *περισσότερη* ενέργειά τους συγκεντρωμένη σε λίγες αρμονικές (βλ. 1.10). Έτσι, μπορούν να μας δώσουν *προσεγγιστική* πληροφορία **και** στο χρόνο **και** στη συχνότητα

Οι τριγωνομετρικές συναρτήσεις (ημίτονα, συνημίτονα) είναι συναρτήσεις των οποίων οι τιμές «ταλαντώνονται» επ'άπειρον και έχουν άπειρη ενέργεια. Είναι λοιπόν, **κύματα**. Από την άλλη, οι $\psi_k(n)$ είναι ακολουθίες που αν και ταλαντώνονται οι τιμές τους, είναι *εντοπισμένες χρονικά* και έχουν πεπερασμένη ενέργεια. Συνεπώς, θα μπορούσαμε να τις ονομάσουμε **κυματίδια** ή **wavelets**. Στην πραγματικότητα, οι ακολουθίες $\psi(n)$ και $\varphi(n)$ που παρουσιάσαμε πριν είναι οι «διακριτοποιημένες» εκδοχές της πρώτης συνάρτησης wavelet και της πρώτης συνάρτησης κλίμακας που ανακαλύφθηκαν ποτέ και σήμερα γνωστές με το όνομα “Haar Wavelet”. Περισσότερα γι' αυτές τις έννοιες, στην επόμενη ενότητα.

Γ. Εισαγωγή στα Wavelets

1. Σύντομη επισκόπηση βασικών εννοιών

Είδαμε πριν πως η αναπαράσταση ενός σήματος χρησιμοποιώντας κάποιες (μάλλον αυθαίρετες) συναρτήσεις βάσης οι οποίες είναι σχεδιασμένες έτσι ώστε η κάθε μια οικογένεια από αυτές να αντιστοιχεί σε συγκεκριμένο επίπεδο ανάλυσης/λεπτομέρειας, άρα και σε κάποιες συγκεκριμένες συχνότητες, μας δίνει πολύ καλύτερες πληροφορίες για ένα μη-περιοδικό σήμα με απότομες μεταβολές απ'ότι ο μετασχηματισμός Fourier. Είμαστε σε θέση πλέον, έχοντας ως πρακτικό παράδειγμα την ανάλυση της προηγούμενης ενότητας, να προχωρήσουμε στους ορισμούς των εννοιών που αφορούν τα wavelets.

Τα wavelets είναι ένας μάλλον καινούριος (οι σημαντικές ανακαλύψεις έγιναν μετά το 1985) και ραγδαία αναπτυσσόμενος κλάδος των μαθηματικών και της επεξεργασίας σήματος. Αυτό αποτελεί εν μέρει ένα μειονέκτημα καθώς πολλά βιβλία που θεωρούνταν κάποτε «κλασσικά» σήμερα στερούνται των τελευταίων εξελίξεων με πολύ σημαντικές εφαρμογές ή είναι γραμμένα σε γλώσσα για μαθηματικούς και όχι για μηχανικούς! Εμείς εδώ θα κάνουμε μια αρκετά χαλαρή μαθηματικά και «πρακτική» εισαγωγή σε όσες βασικές έννοιες χρειάζονται για την κατανόηση αυτών που θα ακολουθήσουν. Στη βιβλιογραφία παραθέτουμε πλήθος συγγραμάτων για τον ενδιαφερόμενο αναγνώστη. Ενδεικτικά, το [A2] αποτελεί μάλλον την καλύτερη πρακτική εισαγωγή στο αντικείμενο, προσφέροντας καλή ενημέρωση πάνω σε καινούριες σχετικά εξελίξεις όπως τα διορθογώνια wavelets. Το [A6] έχει το πλεονέκτημα της αυστηρής μαθηματικής θεμελίωσης που δεν χρειάζεται βαρύ μαθηματικό υπόβαθρο για να γίνει κατανοητό, όπως το [A7], το οποίο όμως είναι ένα αδιαμφισβήτητο κλασσικό σύγγραμμα από την πρωτοπόρο του κλάδου. Ωστόσο, το [A6] παρουσιάζει πολύ στοιχειώδεις έννοιες, με αποτέλεσμα να μην είναι επαρκές για την κατανόηση σύγχρονων εφαρμογών. Τέλος, το [A4] θεωρείται από πολλούς ως το καλύτερο και πιο περιεκτικό βιβλίο για τα wavelets και τις εφαρμογές τους, ωστόσο απαιτεί κάποιο παραπάνω μαθηματικό υπόβαθρο.

Ας θεωρήσουμε όλες τις (γενικευμένες) πραγματικές συναρτήσεις που ανήκουν στον απειροδιάστατο διανυσματικό χώρο $L^2(\mathfrak{R})$, δηλαδή, αν $f(t)$ μια από αυτές, ισχύει πάντοτε

$$\sqrt{\int_{-\infty}^{\infty} |f(t)|^2 dt} < \infty \quad (1.17)$$

Η συνθήκη αυτή σημαίνει πρακτικά ότι η αν η $f(t)$ αναπαριστά ένα σήμα, το σήμα αυτό έχει πεπερασμένη ενέργεια. Αυτό ισχύει για όλα σχεδόν τα σήματα που συναντώνται, και απαιτείται και από άλλα είδη μετασχηματισμών.

Ας θεωρήσουμε, λοιπόν, αυθαίρετα, μια κατάλληλη συνάρτηση $\phi(t)$ που να ανήκει στο $L^2(\mathfrak{R})$ έτσι ώστε όλες οι συναρτήσεις που ορίζονται ως

$$\phi_k(t) = \phi(t - k) \text{ όπου } k \text{ ακέραιος} \quad (1.18)$$

να είναι γραμμικά ανεξάρτητες. Στην πραγματικότητα, ο ορισμός δε χρειάζεται να είναι τόσο αυστηρός, ωστόσο κάτι τέτοιο ξεφεύγει πολύ από τα πλαίσια αυτής της

εργασίας. Το υποσύνολο του $L^2(\mathcal{R})$ που αποτελεί το σύνολο επέκτασης των $\{\phi_k(t)\}$ το ορίζουμε ως

$$V_0 = \overline{\text{Span}_k\{\phi_k(t)\}} \quad (1.19)$$

για όλους τους ακεραίους k από το μείον ως το συν άπειρο. Η μπάρα από πάνω δείχνει ότι πρόκειται για κλειστό σύνολο. Είναι φανερό ότι το V_0 δε θα είναι ποτέ κενό. Ας θεωρήσουμε μια $f(t) \in V_0$. Τότε, αυτή θα μπορεί να γραφεί πάντοτε ως:

$$f(t) = \sum_k a_k \phi_k(t) \quad (1.20)$$

Μια ακόμα προϋπόθεση για την επιλογή της $\phi(t)$ είναι να είναι τέτοια ώστε αυξάνοντας τη χρονική ακρίβεια της αρχικής συνάρτησης, το σύνολο επέκτασης των $\phi_k(mt)$ που προκύπτουν να είναι γνήσιο υπερσύνολο του V_0 . Έτσι, ορίζουμε μια οικογένεια από $\phi(t)$ με δύο παραμέτρους, τη χρονική μετατόπιση και τη χρονική συμπίεση. Ένας συχνά χρησιμοποιούμενος σύμβαση και συμβολισμός για μια τέτοια οικογένεια είναι ο

$$\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - k) \quad (1.21)$$

από τις οποίες, ο διανυσματικός χώρος που καλύπτει η κάθε ομάδα $\phi_j(t)$ είναι ο

$$V_j = \overline{\text{Span}_k\{\phi_k(2^j t)\}} = \overline{\text{Span}_k\{\phi_{j,k}(t)\}} \quad (1.22)$$

για όλους τους ακεραίους k . Αυτό σημαίνει ότι αν η $f(t) \in V_j$, τότε μπορεί να γραφεί ως:

$$f(t) = \sum_k a_k \phi_k(2^j t + k) \quad (1.23)$$

Λόγω των προηγουμένων, έχουμε ότι ισχύει για τα υποσύνολα V_j του $L^2(\mathcal{R})$ που προκύπτουν από κάθε $\{\phi_j(t)\}$:

$$\dots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \dots \subset L^2 \quad (1.24a)$$

ή, εναλλακτικά, $V_j \subset V_{j+1}$, με $V_{-\infty} = \{0\}$ και $V_{\infty} = L^2$ (1.24b)

Μια $\phi(t)$ που ικανοποιεί τις συνθήκες (1.20)...(1.24) ονομάζεται **συνάρτηση κλίμακας (scaling function)**. Μια «συνάρτηση» (ακολουθία) κλίμακας υπό τον ορισμό αυτό ήταν και η $\phi(t)$ που θέσαμε στην (1.1), αφού είδαμε πως οι επόμενες χρονικές συμπίεσεις της μπορούσαν να αναπαραστήσουν σήματα με περισσότερες λεπτομέρειες. Φυσικά, όπως και στο παράδειγμά μας πριν, ενώ οι $\phi(t)$ που ορίζονται από την (1.21) και έχουν το ίδιο j και διαφορετικά k είναι γραμμικά ανεξάρτητες, αυτές που έχουν διαφορετικά j δεν είναι κατ'ανάγκη (και ούτε θα πρέπει να είναι, όλες, για να ικανοποιείται η (1.24a)) γραμμικά ανεξάρτητες.

Ας θεωρήσουμε τώρα μια κατάλληλη $\phi(t)$ με σύνολο επέκτασης το V_0 . Τότε, προφανώς, η $\phi(t)$ περιέχεται στο V_0 αλλά και (λόγω της (1.24a)) σε κάθε άλλο

υπερσύνολό του, π.χ. το V_1 . Συνεπώς, θα μπορεί να γραφεί ως γραμμικός συνδυασμός των στοιχείων της βάσης του V_1 , δηλαδή, τα $\phi_k(2t)$ ως εξής:

$$\phi(t) = \sum_n h(n) \sqrt{2} \phi(2t - n), \text{ όπου } n \text{ ακέραιος} \quad (1.25)$$

Το $h(n)$ είναι μια ακολουθία (πραγματικών) συντελεστών που ονομάζονται **συντελεστές της συνάρτησης κλίμακας** (scaling function coefficients), **φίλτρο κλίμακας** (scaling filter) ή **διάνυσμα κλίμακας** (scaling vector). Η τετραγωνική ρίζα του 2 υπάρχει στο γινόμενο ως επακόλουθο της (1.23) για να είναι κανονικοποιημένα τα $h(n)$. Για παράδειγμα, στο παράδειγμά μας στην ενότητα B, από τις (1.1), (1.4), (1.7), (1.10) μπορούμε εύκολα να δούμε ότι $h(n) = (2^{-1}, 2^{-1})$.

Παρόμοια με την ενότητα B, η επιθυμία μας να έχουμε μια ενιαία βάση για το $L^2(\mathcal{R})$ της οποίας μερικά μέλη να ευθύνονται αποκλειστικά για κάποιο επίπεδο λεπτομερειών μας οδηγεί να ορίσουμε κάποιες νέες συναρτήσεις $\psi(n)$, το σύνολο επέκτασης των οποίων θα είναι η **διαφορά** μεταξύ των διαδοχικών V_j και θα διέπονται από αρχές παραπλήσιες της (1.21).

Δηλαδή, αν $\psi(t)$ μια κατάλληλη τέτοια συνάρτηση που αντιστοιχεί στην $\phi(t)$ που αναλύσαμε στις (1.20)-(1.24), τότε, θα πρέπει να μπορώ να ορίσω τις:

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k) \quad (1.26)$$

τέτοιες ώστε, κάθε υποσύνολο από αυτές ψ_j να έχει ως σύνολο επέκτασης το:

$$\overline{\text{Span}}_k \{ \psi_{j,k}(t) \} = W_j = V_{j+1} - V_j \quad (1.27a)$$

$$\text{ή, εναλλακτικά, } \overline{\text{Span}}_k \{ \{ \psi_{j,k}(t) \} \cup \{ \phi_{j,k}(t) \} \} = V_{j+1} \quad (1.27b)$$

Η (1.27) σημαίνει, αναδρομικά, ότι θα πρέπει όχι μόνο τα $\psi_{j,k}(t)$ με το ίδιο j να είναι γραμμικά ανεξάρτητα μεταξύ τους, αλλά όλα τα $\psi_{j,k}(t)$ να είναι γραμμικά ανεξάρτητα (πάλι, δε χρειάζεται τόση αυστηρότητα αλλά αυτό είναι πέραν των σκοπών αυτής της εργασίας). Οι συναρτήσεις $\psi_{j,k}(t)$ ονομάζονται **συναρτήσεις κυματιδίου** ή **συναρτήσεις wavelet**. Για παράδειγμα, η $\psi(n)$ που ορίσαμε στην (1.12) αποτελεί μια ακολουθία wavelet. Το όνομα “wavelet” («κυματίδιο», αν και εμείς εδώ θα επιμείνουμε στον αγγλικό όρο) προέρχεται από την μορφή μικρού, εντοπισμένου στο χρόνο κύματος (δηλαδή, λαμβάνουν τιμές και πάνω και κάτω από το μηδέν) που έχουν οι περισσότερες από αυτές.

Από τη στιγμή που τα $\{ \psi_j \}$ είναι βάση για το V_{j+1} , τότε ανήκουν σε αυτό. Αυτό σημαίνει, όπως πριν, ότι μπορούν να γραφούν ως γραμμικός συνδυασμός της βάσης του $\{ \phi_{j+1} \}$, οπότε, κατ’αναλογία με την (1.25) ισχύει για το $\psi(t)$:

$$\psi(t) = \sum_n h_1(n) \sqrt{2} \phi(2t - n), \text{ όπου } n \text{ ακέραιος} \quad (1.28)$$

Εδώ, το $h_1(n)$ είναι πάλι μια ακολουθία πραγματικών συντελεστών που ονομάζονται **συντελεστές της συνάρτησης wavelet** (wavelet function coefficients), **φίλτρο wavelet** (wavelet filter) ή **διάνυσμα κλίμακας** (wavelet vector).

Συνοψίζοντας, από τις (1.24) και (1.27) αν επιλέξουμε μια τυχαίας ακρίβειας συνάρτηση κλίμακας $\phi(t)$ και ορίσουμε την κατάλληλη γι' αυτή συνάρτηση wavelet $\psi(t)$, τότε, σύμφωνα με τα παραπάνω, κάθε πραγματικό σήμα που ανήκει στο $L^2(\mathcal{R})$ μπορεί να γραφεί ως

$$g(t) = \sum_{k=-\infty}^{\infty} c_{j_0}(k) * \phi_k(t) + \sum_{j=0}^{\infty} \sum_{k=-\infty}^{\infty} d_j(k) * \psi_{j,k}(t) \quad (1.29)$$

Η ανάλυση ενός σήματος στα $c_{j_0}(k)$ και $d_j(k)$ σύμφωνα με κάποια δεδομένα $\phi(t)$ και $\psi(t)$ λέγεται **διακριτός μετασχηματισμός wavelet (discrete wavelet transform)** του σήματος αυτού, και, όπως είδαμε στην ενότητα B, τα $d_j(k)$ μπορεί να έχουν αξία ανάλογη των συντελεστών που επιστέφει ο μετασχηματισμός fourier. Τα $c_{j_0}(k)$ (και εν γένει, οι συντελεστές της συνάρτησης κλίμακας) θα ονομάζονται από εδώ και εμπρός **σήμα προσέγγισης**, ή συντελεστές προσέγγισης, και τα $d_j(k)$ (για ένα συγκεκριμένο j) **σήμα λεπτομερειών**, ή συντελεστές wavelet.

Ουσιαστικά, έχουμε ορίσει όλες σχεδόν τις παραμέτρους που απαιτούνται για την επιλογή των συναρτήσεων $\psi(t)$, $\phi(t)$ (υπάρχουν και μερικές άλλες απαιτήσεις, μαθηματικής φύσεως που δε μας αφορούν εδώ). Πράγματι, σε αντίθεση με τους μετασχηματισμούς fourier, συνημιτόνου, czt κλπ. οι υποψήφιες συναρτήσεις κλίμακας και wavelet για το διακριτό μετασχηματισμό wavelet είναι *άπειρες*. Ωστόσο, υπάρχουν διάφορα κριτήρια που ευνοούν κάποιες συναρτήσεις έναντι κάποιων άλλων. Πριν αναφερθούμε όμως στα κριτήρια αυτά, θα προσπαθήσουμε να εξάγουμε τα $c_{j_0}(k)$, $d_j(k)$ από την (1.29)

Μέχρι τώρα, το μόνο κριτήριο που θέσαμε για τη σχέση μεταξύ των $\phi_k(t)$, $\psi_{j,k}(t)$ είναι να είναι γραμμικά ανεξάρτητα μεταξύ τους έτσι ώστε να μπορούν να αποτελούν μια βάση. Ένας τρόπος να εξάγουμε τα $c_{j_0}(k)$, $d_j(k)$ από την (1.29) θα ήταν να απαιτούσαμε τα $\phi_k(t)$, $\psi_{j,k}(t)$ να είναι εκτός από γραμμικά ανεξάρτητα να είναι και *ορθογώνια* μεταξύ τους, δηλαδή, να ισχύει πάντοτε:

$$\int_{-\infty}^{\infty} \psi_{j,k}(t) \psi_{l,m}(t) dt = \delta(j-l, m-k) \quad (1.30)$$

και παρόμοια και για τα $\phi_k(t)$. *Σημείωση: στην ουσία η 1.30 απαιτεί οι συναρτήσεις να είναι ορθοκανονικές και όχι απλά ορθογώνιες, ωστόσο, σε περίπτωση που είναι ορθογώνιες και όχι ορθοκανονικές, μπορούμε να τις μετατρέψουμε σε ορθοκανονικές με κατάλληλους συντελεστές*

Στην περίπτωση που οι συναρτήσεις είναι ορθογώνιες, τότε οι συντελεστές μπορούν να ληφθούν από τις σχέσεις:

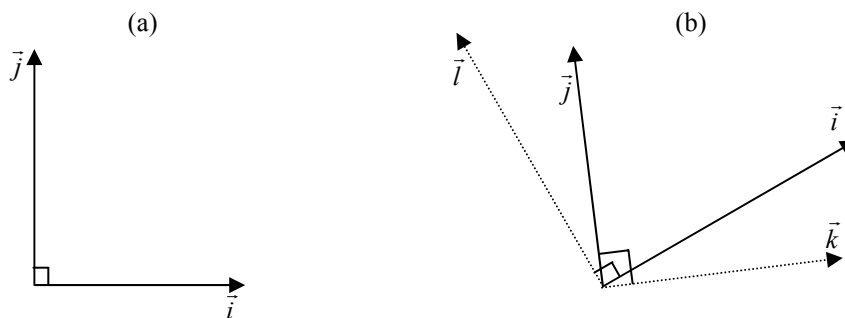
$$c_{j_0}(k) = \int_{-\infty}^{\infty} g(t) \phi_k(t) dt \quad (1.31a)$$

$$\text{και } d_j(k) = \int_{-\infty}^{\infty} g(t) * \psi_{j,k}(t) dt \quad (1.31b)$$

Ωστόσο, υπάρχουν περιπτώσεις που οι συναρτήσεις κλίμακας και wavelet δε θέλουμε να είναι ορθογώνιες ή δεν *πρέπει* να είναι ορθογώνιες: στην πράξη οι ορθογώνιες συναρτήσεις έχουν μια σειρά από περιορισμούς που τις κάνουν μη-

επιθυμητές για τις περισσότερες εφαρμογές. Τότε, μπορούμε να απαιτήσουμε τα $\phi_k(t)$, $\psi_{j,k}(t)$ να είναι *διορθογώνια*.

Τι σημαίνει διορθογώνια; Θα δώσουμε ένα γεωμετρικό ανάλογο. Στο σχ. 1.11(a), τα διανύσματα \vec{i} και \vec{j} είναι ορθογώνια. Αυτό σημαίνει ότι αν εκφράσουμε οποιοδήποτε διάνυσμα του επιπέδου ως γραμμικό συνδυασμό των \vec{i} , \vec{j} τότε ο πολλαπλασιασμός του (εσωτερικό γινόμενο) με π.χ. το \vec{i} θα μας δώσει τον συντελεστή του \vec{i} στο διάνυσμα αυτό.



Σχήμα 1.12 Γεωμετρικό ανάλογο μιας ορθογώνιας (a) και μιας διορθογώνιας (b) βάσης

Στο σχ. 1.11b αυτό δε μπορεί να γίνει με τα \vec{i}, \vec{j} , καθώς το εσωτερικό τους γινόμενο είναι διαφορετικό του μηδενός. Ωστόσο, μπορούμε να ορίσουμε δυο νέα διανύσματα, τα \vec{k} και \vec{l} , τα οποία είναι ορθογώνια ως προς το \vec{j} και το \vec{i} αντίστοιχα. Έτσι, αν έχουμε εκφράσει ένα οποιοδήποτε διάνυσμα του επιπέδου ως γραμμικό συνδυασμό των \vec{i} , \vec{j} τότε, το εσωτερικό γινόμενό του π.χ. με το \vec{k} θα μας δώσει το συντελεστή του \vec{j} . Έτσι, τα (\vec{i}, \vec{j}) σχηματίζουν μαζί με τα *δυναδικά* τους (\vec{k}, \vec{l}) μια *διορθογώνια* βάση στο επίπεδο.

Ας επιστρέψουμε, τώρα, στο διακριτό μετασχηματισμό wavelet. Αν λοιπόν απαιτήσουμε τα $\phi_k(t)$, $\psi_{j,k}(t)$ να είναι διορθογώνια, τότε θα πρέπει να είμαστε σε θέση να ορίσουμε και τα *δυναδικά* τους $\phi'_k(t)$, $\psi'_{j,k}(t)$, παρόμοια με το γεωμετρικό ανάλογο, έτσι ώστε, π.χ. για τα $\psi_{j,k}(t)$ να ισχύει

$$\int_{-\infty}^{\infty} \psi_{j,k}(t) \psi'_{l,m}(t) dt = \delta(j-l, m-k) \quad (1.32)$$

Από τη στιγμή που τα $\phi'_k(t)$, $\psi'_{j,k}(t)$ είναι *δυναδικά* των $\phi_k(t)$ και $\psi_{j,k}(t)$, μπορούν και αυτά να αποτελέσουν βάσεις των αντίστοιχων υποσυνόλων του $L^2(\mathcal{R})$ και επιπλέον γι'αυτά θα ισχύουν (αντίστοιχα) οι (1.21) και (1.26). Από το γεγονός ότι αποτελούν (και αυτά) βάση για τους αντίστοιχους διανυσματικούς υποχώρους, σημαίνει ότι θα πρέπει να ορίζονται τα $h'(t)$, $h_1'(t)$ έτσι ώστε και εδώ έχουμε:

$$\phi'(t) = \sum_n h'(n) \sqrt{2} \phi'(2t-n) \quad (1.33)$$

$$\text{και } \psi'(t) = \sum_n h_1'(n) \sqrt{2} \psi'(2t-n) \quad (1.34)$$

όπου n ακέραιος

Τότε, τα σήματα προσέγγισης και λεπτομερειών θα μπορούν να υπολογιστούν ως εξής:

$$c_{j0}(k) = \int_{-\infty}^{\infty} g(t)\phi_k'(t)dt \quad (1.35a)$$

$$\text{και } d_j(k) = \int_{-\infty}^{\infty} g(t) * \psi_{j,k}'(t)dt \quad (1.35b)$$

Οι διορθωόνιες συναρτήσεις κλίμακας και wavelet (που συχνά θα αποκαλούμε συλλογικά ως «διορθωόνια wavelets») έχουν πολύ περισσότερους βαθμούς ελευθερίας και συνεπώς πολύ μεγαλύτερη ευελιξία από τα ορθωόνια. Πράγματι, οι περισσότεροι μετασχηματισμοί wavelet που συμβαίνουν στην πράξη έχουν να κάνουν με διορθωόνια wavelets.

Ωστόσο, ούτε οι (1.31), ούτε οι (1.33) μας δίνουν κάποιον *αποδοτικό* και *αποτελεσματικό* τρόπο να υπολογίσουμε τα σήματα προσέγγισης και λεπτομερειών. Ένα ολοκλήρωμα είναι δύσκολο να υπολογιστεί γρήγορα με κάποιον αλγόριθμο και επιπλέον, πριν προχωρήσουμε στην ολοκλήρωση θα πρέπει αν ξέρουμε *ποια* σήματα ακριβώς θέλουμε, καθώς και τα $c_{j0}(k)$ και τα $d_j(k)$ είναι θεωρητικά άπειρα. Επιπλέον, οι (1.31), (1.33) αφορούν μόνο πραγματικά σήματα, και όχι σήματα ψηφιακά που έχουν προέλθει από δειγματοληψία, όπως αυτά που υπάρχουν μέσα σε ψηφιακά κυκλώματα. Έτσι, θα πρέπει κανονικά όλη η παραπάνω θεωρία να αναδιατυπωθεί για σήματα διακριτού χρόνου, κατά τρόπο ανάλογο με την εξαγωγή του μετασχηματισμού Fourier διακριτού χρόνου από το μετασχηματισμό Fourier.

Ευτυχώς, κάτι τέτοιο δε χρειάζεται να γίνει. Όπως θα δείξουμε στην αμέσως επόμενη υποενότητα, τα σήματα προσέγγισης και λεπτομερειών μπορούν να υπολογιστούν πολύ πιο εύκολα και αποδοτικά από μια διάταξη που λέγεται συστοιχία φίλτρων. Στην πραγματικότητα, στον υπολογισμό ενός μετασχηματισμού wavelet, δε χρειάζεται καν να γνωρίζουμε την ακριβή μορφή της συνάρτησης wavelet και της συνάρτησης κλιμάκωσης που σχετίζονται με αυτόν.

2. Wavelets και συστοιχίες φίλτρων

Θα δείξουμε εδώ συνοπτικά πως ο μετασχηματισμός wavelet μπορεί να πραγματοποιηθεί όχι από (1.31), (1.35) αλλά από μια πολύ πιο απλή διάταξη που ονομάζεται **συστοιχία φίλτρων (filter bank)**. Η «απόδειξη» που θα κάνουμε αφορά κυρίως ορθωόνιες συναρτήσεις κλίμακας και wavelet, αλλά είναι πολύ παρόμοια για την περίπτωση των διορθωόνιων συναρτήσεων βάσης.

Ας θεωρήσουμε μια ορθωόνια βάση από $\{\phi_k, \psi_{j,k}\}$, όπου η ϕ_k έχει επιλεγεί αυθαίρετα έτσι ώστε να μπορεί να αναπαριστά σήματα μέχρι ένα επίπεδο λεπτομερειών. Τότε, επεκτείνοντας την (1.25) μπορούμε να γράψουμε:

$$\phi(2^j t - k) = \sum_n h(n)\sqrt{2}\phi(2(2^j t - k) - n) = \sum_n h(n)\sqrt{2}\phi(2^{j+1}t - 2k - n) \quad (1.36)$$

Αν στην (1.36) αντικαταστήσουμε το $2k+n$ με m , παίρνουμε:

$$\phi(2^j t - k) = \sum_m h(m - 2k)\sqrt{2}\phi(2^{j+1}t - m) \quad (1.37)$$

Ας θεωρήσουμε τώρα, έναν από τους υποχώρους V_j που ορίσαμε στην (1.22), δηλαδή

$$V_j = \overline{\text{Span}}_k \{ \phi_{j,k}(t) \} = \overline{\text{Span}}_k \{ \phi(2^j t - k) \} \quad (1.38)$$

Τότε, αν μια $f(t)$ ανήκει στο V_{j+1} θα μπορεί να γραφεί, σύμφωνα με την (1.23) και χρησιμοποιώντας ορολογία της (1.29) ως

$$f(t) = \sum_k c_{j+1}(k) \sqrt{2} \phi_{j+1,k}(t) = \sum_k c_{j+1}(k) 2^{(j+1)/2} \phi(2^{j+1}t - k) \quad (1.39)$$

Αν ωστόσο θέλουμε να εκφράσουμε την $f(t)$ ως προς τις $\phi_{j,k}(t)$ θα πρέπει αναγκαστικά να χρησιμοποιήσουμε και τις συναρτήσεις wavelet $\psi_{j,k}(t)$, οπότε, από την (1.29), θα έχουμε:

$$\begin{aligned} f(t) &= \sum_k c_j(k) \sqrt{2} \phi_{j,k}(t) + \sum_k d_j(k) \psi_{j,k}(t) = \\ &= \sum_k c_j(k) 2^{j/2} \phi(2^j t - k) + \sum_k d_j(k) 2^{j/2} \phi(2^j t - k) \end{aligned} \quad (1.40)$$

Επειδή τα $\{\phi_k(t)\}$, $\{\psi_{j,k}(t)\}$ είναι ορθογώνια, τότε, από την (1.40) θα μπορούμε να λάβουμε το σήμα λεπτομερειών $c_j(k)$ από το ολοκλήρωμα

$$c_j(k) = \int_{-\infty}^{\infty} f(t) \phi_{j,k}(t) dt = \int_{-\infty}^{\infty} f(t) 2^{j/2} \phi(2^j t - k) dt \quad (1.41)$$

Από την (1.37) ισχύει όμως $\phi(2^j t - k) = \sum_m h(m - 2k) \sqrt{2} \phi(2^{j+1}t - m)$, άρα, αντικαθιστώντας στην (1.41) παίρνουμε:

$$\begin{aligned} c_j(k) &= \int_{-\infty}^{\infty} f(t) 2^{j/2} \phi(2^j t - k) dt = \\ &= \int_{-\infty}^{\infty} f(t) 2^{j/2} \left(\sum_m h(m - 2k) 2^{1/2} \phi(2^{j+1}t - m) \right) dt = \\ &= \sum_m h(m - 2k) \int_{-\infty}^{\infty} f(t) 2^{(j+1)/2} \phi(2^{j+1}t - m) dt \end{aligned} \quad (1.42)$$

αλλά από την (1.41) έχουμε ότι

$$\int_{-\infty}^{\infty} f(t) 2^{(j+1)/2} \phi(2^{j+1}t - m) dt = c_{j+1}(m) \quad (1.43)$$

άρα, η (1.42) μετατρέπεται στην:

$$c_j(k) = \sum_m h(m - 2k) c_{j+1}(m) \quad (1.44a)$$

Η (1.44) μας δείχνει τον τρόπο να λάβουμε το σήμα προσέγγισης ενός μετασχηματισμού wavelet από το σήμα προσέγγισης της αμέσως μεγαλύτερης

ακρίβειας μέσω ενός απλού γραμμικού φίλτρου με υποδειγματοληψία ανά δύο στην έξοδο. Παρόμοια, για το σήμα λεπτομερειών (συντελεστές wavelet) ισχύει

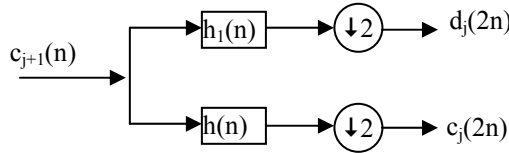
$$d_j(k) = \sum_m h_1(m - 2k) c_{j+1}(m) \quad (1.44b)$$

Ενώ με τον ίδιο τρόπο αποδεικνύεται ότι αν τα $\{\phi_k(n)\}$, $\{\psi_{j,k}(n)\}$ διορθογώνια, οι (1.44) έχουν τη μορφή:

$$c_j(k) = \sum_m h'(m - 2k) c_{j+1}(m) \quad (1.45a)$$

$$d_j(k) = \sum_m h_1'(m - 2k) c_{j+1}(m) \quad (1.45b)$$

Η σημασία των σχέσεων (1.44) και (1.45) είναι θεμελιώδης, αφού μας επιτρέπει να εκτελέσουμε το μετασχηματισμό wavelet ενός σήματος με τη χρήση ενός ζεύγους φίλτρων (εξ' ου και ο όρος filter bank) χωρίς να χρειάζεται καν να ξέρουμε τη μορφή των συναρτήσεων wavelet και κλίμακας που ορίζουν το μετασχηματισμό. Υπάρχει ένα



Σχήμα 1.13 Υπολογισμός των σημάτων προσέγγισης και λεπτομερειών μέσω μιας συστοιχίας φίλτρων

πρόβλημα, ωστόσο. Πως θα υπολογίσουμε τα c_{j+1} για κάποιο j ; Και επιπλέον, αφού τα j είναι άπειρα για ένα σήμα για το οποίο δε γνωρίζουμε σε ποιο υποσύνολο V_j του $L^2(\mathcal{R})$ ανήκει, δε θα πρέπει αυτό να γίνεται συνέχεια; Η απάντηση είναι όχι, και μας δίνει τη λαβή για την εφαρμογή των παραπάνω στα σήματα διακριτού χρόνου. Έχουμε ότι

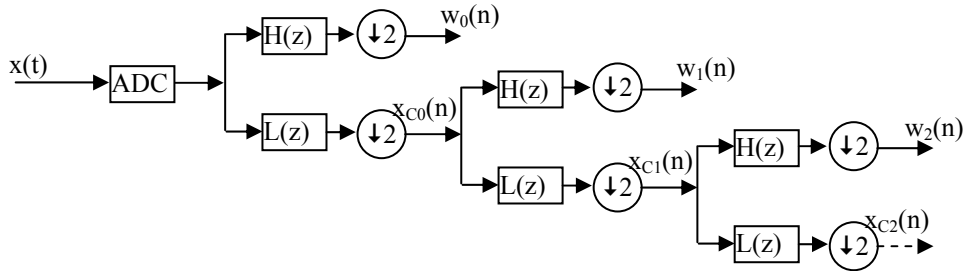
$$c_j(k) = \int_{-\infty}^{\infty} f(t) \phi_{j,k}(t) dt . \text{ Αν η } \phi(t) \text{ είναι σωστά επιλεγμένη (κάτι που ισχύει για όλες}$$

τις συναρτήσεις κλίμακας που χρησιμοποιούνται στην πράξη), τότε για μεγάλα j η $\phi_{j,k}(t)$ δρα σαν τη συνάρτηση dirac, η οποία είναι γνωστή και ως συνάρτηση δειγματοληψίας (ως ανάλογο παράδειγμα, στην (1.11) η $\phi(n)$ τελικά έγινε ο μοναδιαίος παλμός). Συνεπώς, με σεβασμό του θεωρήματος δειγματοληψίας του Shannon (και σε περίπτωση μη ικανοποίησής του, φιλτράρισμα του αρχικού αναλογικού σήματος έτσι ώστε να το ικανοποιεί), τα $c_{j+1}(n)$ είναι τα διακριτού χρόνου δείγματα του αρχικού σήματος.

Έτσι, μπορούμε άμεσα να εφαρμόσουμε τη διάταξη του σχ. 1.12 στην έξοδο ενός μετατροπέα αναλογικού-σε-ψηφιακό (ADC converter) και έτσι να λάβουμε τα σήματα προσέγγισης και λεπτομερειών του $1^{ου}$ επιπέδου. Στη συνέχεια, αν θέλουμε, μπορούμε να περάσουμε και το σήμα $c_j(2n)$ από μια ίδια διάταξη έτσι ώστε να λάβουμε τα σήματα προσέγγισης και λεπτομερειών για τα επόμενα επίπεδα (χαμηλότερες συχνότητες) και ούτω καθ' εξής.

Η διάταξη αυτή είναι γνωστή και ως *Αλγόριθμος Πυραμίδας* (Pyramid Algorithm) και ανακαλύφθηκε από τον Mallat στα τέλη της δεκαετίας του 1980. Αυτή η ανακάλυψη ήταν που έκανε εφικτό υπολογιστικά τον Διακριτό Μετασχηματισμό wavelet και άνοιξε το δρόμο για τις μετέπειτα εφαρμογές του. Ο αλγόριθμος

πυραμίδας ονομάζεται σε ορισμένα βιβλία και ως «Ταχύς Μετασχηματισμός Wavelet» ή «Fast Wavelet Transform». Στην πραγματικότητα, είναι ο μόνος εφαρμόσιμος αλγόριθμος για το μετασχηματισμό wavelet, συνεπώς στην εργασία αυτή όταν θα αναφερόμαστε στον Διακριτό Μετασχηματισμό Wavelet, θα εννοούμε τον αλγόριθμο πυραμίδας του Mallat. Έτσι, μπορούμε άμεσα να υλοποιήσουμε τον διακριτό μετασχηματισμό wavelet σε υλικό ως ένα ζευγάρι (ψηφιακών) φίλτρων με υποδειγματοληψία στην έξοδο.



Σχήμα 1.14 Μετασχηματισμός wavelet τριών σταδίων

Στις εφαρμογές της επεξεργασίας σήματος, το φίλτρο κλίμακας ονομάζεται *βαθυπερατό φίλτρο* και συμβολίζεται με $L(z)$ στο χώρο κατάστασης, ενώ το φίλτρο wavelet *υπιπερατό φίλτρο* και συμβολίζεται με $H(z)$. Επιπλέον, τα σήματα προσέγγισης και λεπτομερειών που προκύπτουν από ένα ζεύγος φίλτρων όπως στο σχ. 1.12 συμβολίζονται με $x_{cm}(n)$, $w_m(n)$ αντίστοιχα (αντί για $c_j(k)$, $d_j(k)$) όπου $m=0,1,2,\dots$ ένας ακέραιος που δείχνει πόσες συστοιχίες φίλτρων έχουν προηγηθεί από το αρχικό κβαντισμένο σήμα (βλ. σχ. 1.13). Το m μπορεί να απουσιάζει όταν μελετούμε απλώς τη συστοιχία φίλτρων και δε μας ενδιαφέρει το ποιο στάδιο του μετασχηματισμού θα εκτελέσει. Οι συμβολισμοί αυτοί θα υιοθετηθούν και για το υπόλοιπο της εργασίας.

3. Επιλογή wavelet

Δείξαμε πριν πως ο (διακριτός) μετασχηματισμός wavelet ενός σήματος μπορεί αν υπολογιστεί εύκολα με τη χρήση ενός ζεύγους φίλτρων. Δεν αναφερθήκαμε όμως εν τέλει στο ποιες θα είναι οι συναρτήσεις κλίμακας και wavelet που ορίζουν το μετασχηματισμό.

Υπάρχει μια πληθώρα από μαθηματικά κριτήρια για να μπορεί μια συνάρτηση να αποτελεί συνάρτηση κλίμακας. Για παράδειγμα, θα πρέπει οπωσδήποτε να ικανοποιούνται οι σχέσεις που παραθέσαμε πριν. Ούτως ή άλλως, όπως αναφέραμε, ο αριθμός των υποψηφίων συναρτήσεων βάσης και wavelet είναι *άπειρος*. Ωστόσο, πάντα γίνεται κάποια επιλογή με *πρακτικά* κυρίως κριτήρια.

Για παράδειγμα, μας ενδιαφέρει άμεσα για μια οικογένεια σημάτων να βρούμε τέτοιες συναρτήσεις κλίμακας και wavelet έτσι ώστε τα σήματα αυτά να περιέχουν ένα μεγάλο ποσοστό της ενέργειάς τους σε λίγους μόνο συντελεστές του μετασχηματισμού, να μπορεί δηλαδή να γίνει μια *αραιή αναπαράσταση* του αρχικού σήματος με χρήση του Διακριτού Μετασχηματισμού wavelet. Αυτό είναι πολύ σημαντικό για τεχνικές συμπίεσης και αφαίρεσης θορύβου.

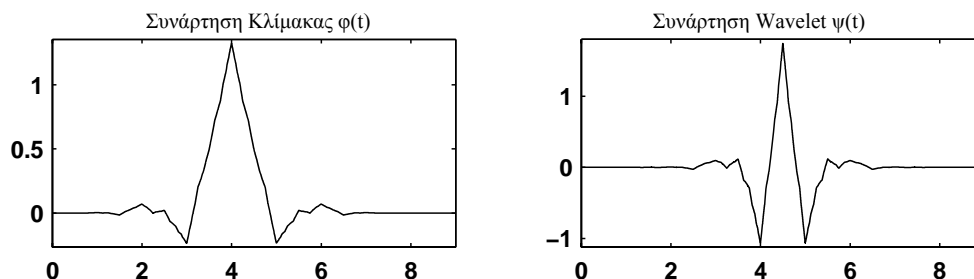
Ένα άλλο πολύ σημαντικό κριτήριο, που θα εξασφαλίζει και τον εύκολο υπολογισμό του μετασχηματισμού είναι τα φίλτρα κλίμακας και wavelet, $h(n)$ και $h_1(n)$ για τα ορθογώνια wavelet και $h'(n)$ και $h_1'(n)$ για τα διορθογώνια, να είναι

πεπερασμένου μήκους (δηλαδή, FIR φίλτρα) και να έχουν λίγα σημεία. Επιπλέον ιδιότητες των φίλτρων που διευκολύνουν τους υπολογισμούς, π.χ. συμμετρικότητα των συντελεστών, είναι φυσικά επιθυμητές.

Εν γένει, η θεωρία της δημιουργίας μιας κατάλληλης συνάρτησης κλίμακας και wavelet με βάση κάποιες προδιαγραφές και απαιτήσεις είναι ένας ολόκληρος κλάδος των μαθηματικών από μόνος του και έχουν γραφεί πολλά ειδικευμένα βιβλία πάνω σε αυτό το ζήτημα (στη βιβλιογραφία παραθέτουμε μερικά). Αυτό είναι και ένα πολύ ισχυρό πλεονέκτημα του μετασχηματισμού wavelet, καθώς υπάρχει η ευελιξία της επιλογής των καταλληλότερων συναρτήσεων wavelet και βάσης για μια οικογένεια σημάτων ή για ένα συγκεκριμένο σκοπό επεξεργασίας. Έτσι, έχουν αναπτυχθεί wavelets (δηλαδή, συναρτήσεις wavelet και βάσης) για μια πληθώρα εφαρμογών, όπως γεωφυσική, συμπίεση εικόνας και ήχου, γραφικά, οικονομικές προβλέψεις κλπ. που σε κάθε περίπτωση δίνουν μετασχηματισμούς πολύ πιο «πληροφοριακούς» απ' ό,τι οι βάσεις τριγωνομετρικών συναρτήσεων.

Μια πολύ διαδεδομένη οικογένεια wavelet, με το μετασχηματισμό ενός μέλους της οποίας θα ασχοληθούμε και εμείς στη συνέχεια, είναι τα wavelets Cohen-Daubechies-Feauveau, η εν συντομία CDF. Πρόκειται διορθογώνια wavelets τα οποία έχουν συμμετρικά, FIR φίλτρα και των οποίων τα μήκη είναι κατά το δυνατόν παρόμοια. Τα CDF έχουν και άλλες επιθυμητές και σημαντικές ιδιότητες, βέβαια, αλλά αυτές είναι πέραν των σκοπών της εργασίας.

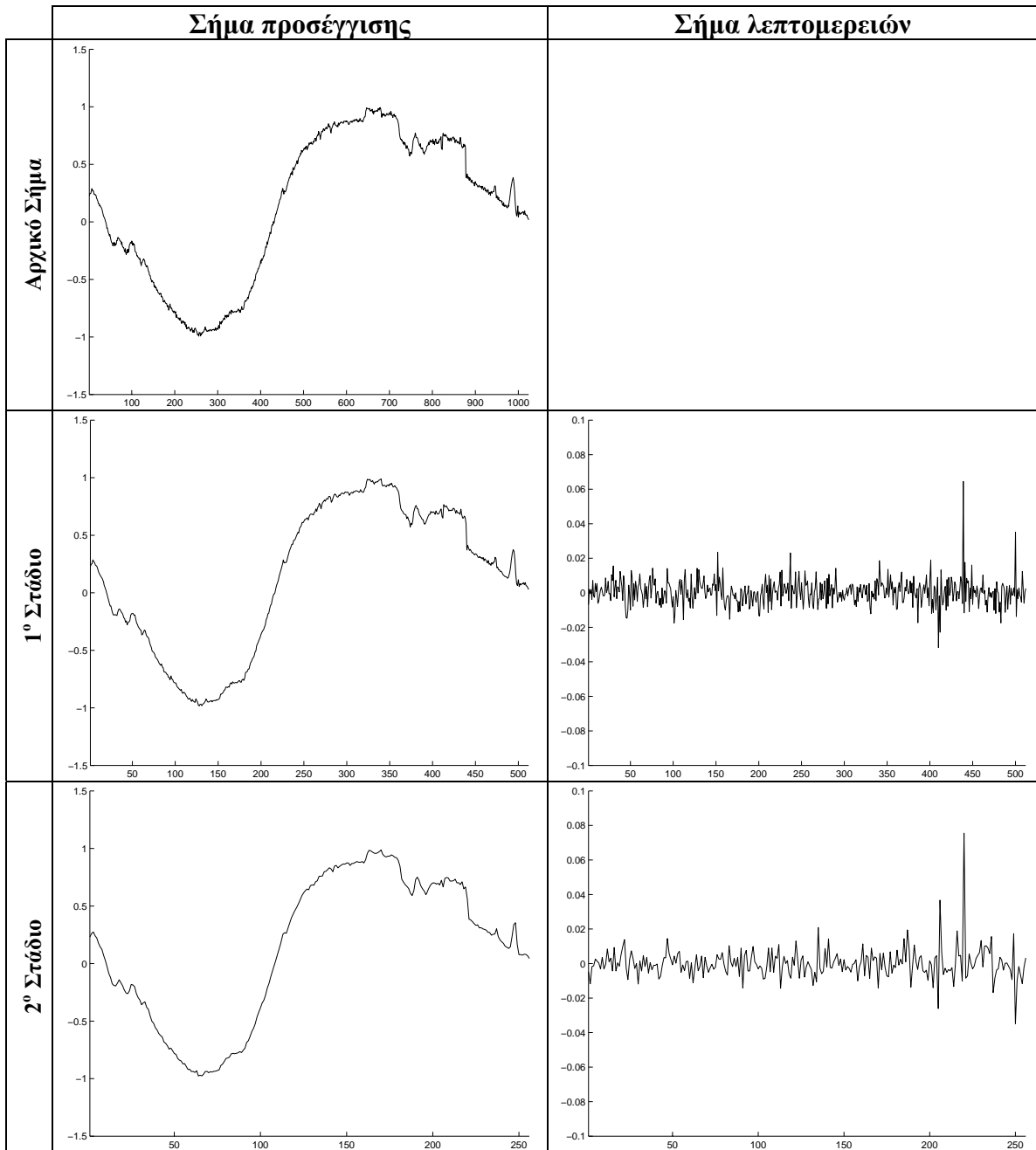
Ένα μέλος της οικογένειας CDF χαρακτηρίζεται από το μήκος (σε σημεία) του βαθυπερατού και του υψιπερατού φίλτρου που εκτελούν τον αντίστοιχο διακριτό μετασχηματισμό. Το πιο διαδεδομένο και γνωστό μέλος αυτής της οικογένειας είναι το CDF 9/7, στον ευθύ διακριτό μετασχηματισμό του οποίου το βαθυπερατό φίλτρο έχει μήκος 9 σημείων και το υψιπερατό 7.



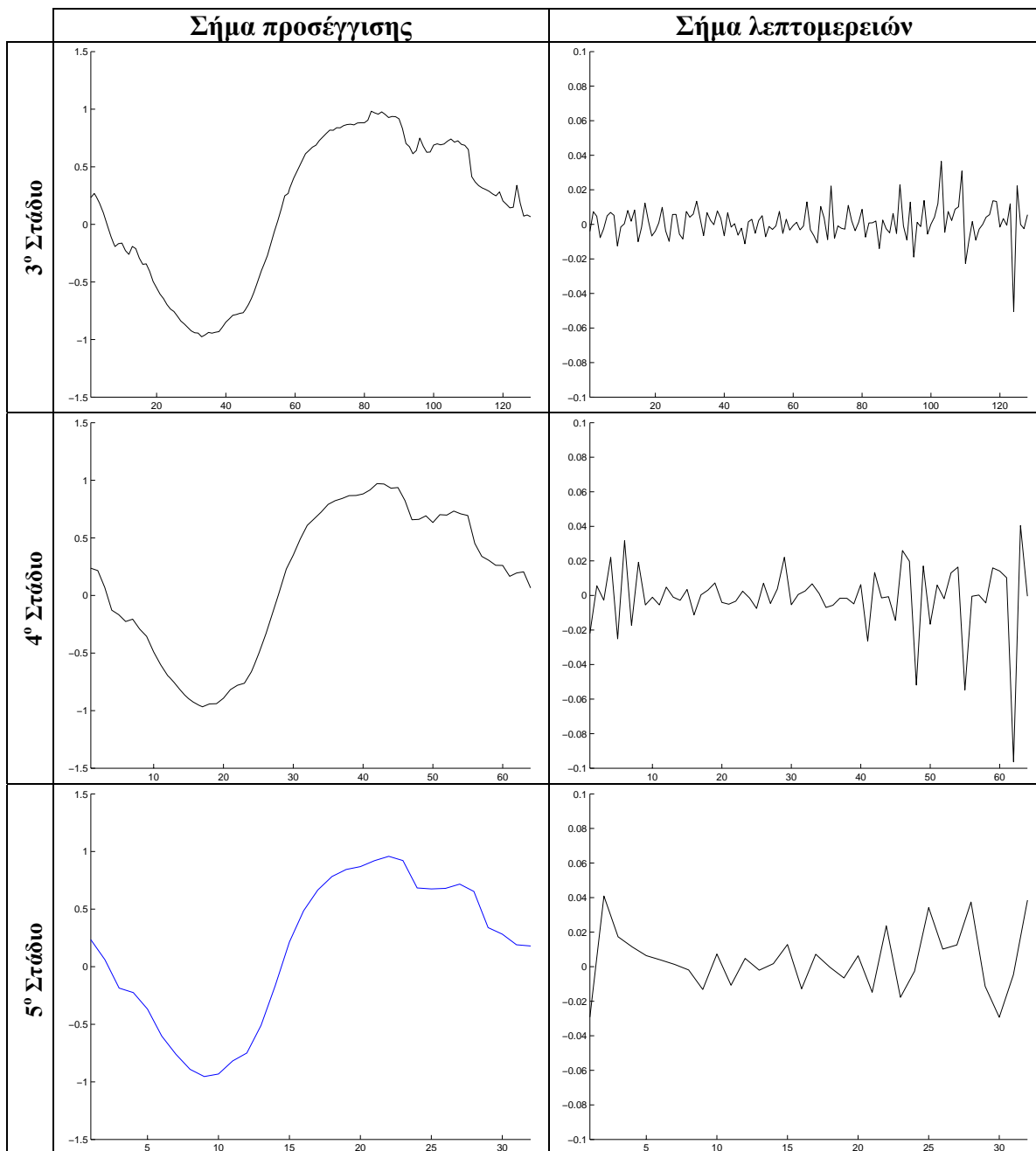
Σχήμα 1.15 Συνάρτηση κλίμακας και wavelet Cohen-Daubechies-Feauveau 9/7

Στην εργασία αυτή θα ασχοληθούμε με αρχιτεκτονικές αριθμητικών κυκλωμάτων που εκτελούν το διακριτό μετασχηματισμό wavelet CDF 9/7. Αν και όπως δείξαμε πριν, για την εκτέλεση του μετασχηματισμού δε χρειάζεται να ξέρουμε τη μορφή των συναρτήσεων κλίμακας και wavelet παρά μόνο τους συντελεστές των $H(z)$ και $L(z)$, παραθέτουμε εδώ τη μορφή τους για λόγους πληρότητας, καθώς και τα αποτελέσματα του διακριτού μετασχηματισμού τους (5 σταδίων) στα 1024 πρώτα δείγματα του σήματος `leleccum` από τις βιβλιοθήκες του Matlab.

Πίνακας 1.1 Σταδιακή αποδόμηση του σήματος leccum με χρήση του μετασχηματισμού wavelet CDF9/7, τα πρώτα πέντε στάδια



Πίνακας 1.1 (συνέχεια) Σταδιακή αποδόμηση του σήματος leleccum με χρήση του μετασχηματισμού wavelet CDF9/7, τα πρώτα πέντε στάδια



Δ. Εφαρμογές

Από τα παραπάνω είναι εμφανές ότι οι μετασχηματισμοί wavelet έχουν μια πληθώρα εφαρμογών σε θέματα επεξεργασίας σημάτων. Είκοσι σχεδόν χρόνια μετά την ανακάλυψη των πρώτων σημαντικών wavelets από την Ingrid Daubechies, και παρά τα διάφορα προβλήματα νομικής φύσεως που έχουν προκύψει λόγω ευρεσιτεχνιών στα μαθηματικά, χρησιμοποιούνται συνεχώς σε τομείς όπου

συναντάμε μη-εργοδικά, μη-περιοδικά σήματα και οι παραδοσιακοί μετασχηματισμοί Fourier αδυνατούν να δώσουν χρήσιμα αποτελέσματα.

Στη βιβλιογραφία που παραθέτουμε στο τέλος αναφέρονται πλήθος από τέτοιες εφαρμογές. Οι κυριότερες είναι στην Ιατρική και τη Μηχανολογία, για εντοπισμό απότομων ασυνεχειών που υποδηλώνουν προβληματική λειτουργία και οι οποίες θα χάνονταν μέσα στο θόρυβο αν χρησιμοποιούσαμε τεχνικές Fourier, στην αστρονομία για τον διαχωρισμό σημάτων από το θόρυβο, ακόμα και στα μακροοικονομικά για μακροπρόθεσμες προβλέψεις της πορείας των μετοχών. Μια από τις πιο γνωστές εφαρμογές, στις οποίες χρησιμοποιείται το wavelet CDF 9/7 είναι ο αλγόριθμος συμπίεσης εικόνας JPEG2000, ο οποίος δημιουργήθηκε για να αντικαταστήσει τον πολύ γνωστό JPEG. Αν και οι πατέντες λογισμικού έχουν αποτρέψει την ευρείας κλίμακας χρησιμοποίησή του ως τώρα, πετυχαίνει πολύ ανώτερα αποτελέσματα από τον παραδοσιακό JPEG τόσο σε επίπεδο συμπίεσης όσο και επίπεδο ταχύτητας. Επιπλέον, η φύση των wavelets να μπορούν να δίνουν επιλεκτικά μέχρι ένα επίπεδο λεπτομερειών σε ένα σήμα, έχει βρει εφαρμογή σε περιπτώσεις που έχουμε μεγάλες εικόνες τις οποίες όταν τις βλέπουμε στο σύνολό τους δε χρειαζόμαστε μεγάλη λεπτομέρεια, αλλά όταν εστιάζουμε σε μια συγκεκριμένη περιοχή περιμένουμε να δούμε και λεπτομέρειες. Τέτοιες είναι για παράδειγμα τοπογραφικές εικόνες που χρησιμοποιούνται από συστήματα GIS, για τις οποίες έχει αναπτυχθεί το πρότυπο συμπίεσης με wavelets ECW

Είναι φανερό ότι λόγω της ωρίμανσης των σχετικών αλγορίθμων αλλά και της επίλυσης των νομικών ζητημάτων στο προσεχές μέλλον, οι εφαρμογές των μετασχηματισμών wavelet θα πολλαπλασιαστούν και ενδεχομένως θα εκτοπίσουν από μερικούς τομείς τους παραδοσιακούς μετασχηματισμούς αναπαράστασης στη συχνότητα. Έτσι, κρίνεται επιτακτική η ανάπτυξη αποδοτικών και εξειδικευμένων αριθμητικών κυκλωμάτων που θα είναι σε θέση να εκτελούν γρήγορα και οικονομικά τους μετασχηματισμούς αυτούς σε ενσωματωμένα συστήματα, όπως φορητά ιατρικά όργανα, κινητά τηλέφωνα, «mp3» players και ψηφιακές φωτογραφικές μηχανές. Αυτό είναι και το ζήτημα με το οποίο πραγματεύεται η εργασία αυτή

Κεφάλαιο 2

Αρχιτεκτονικές σε επίπεδο βασικών μονάδων

A. Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε τις αρχιτεκτονικές για τον ευθύ διακριτό μετασχηματισμό wavelet για το wavelet Cohen-Daubechies-Faveau 9/7 σε επίπεδο βασικών μονάδων.

Πιο συγκεκριμένα, τα κυκλώματα που θα παρουσιάσουμε δε θα αναλυθούν ως το επίπεδο πύλης, αλλά ως το επίπεδο πολλαπλασιαστών, αθροιστών/αφαιρετών, πολυπλεκτών και καταχωρητών. Δε θα ασχοληθούμε ούτε με την ενδιάμεση μορφή των δεδομένων, δηλαδή αν αυτά είναι σε κανονική μορφή, συμπληρώματος ως προς 2, carry-save κοκ, ούτε φυσικά με το πως υλοποιούνται π.χ. οι πολλαπλασιαστές, δηλαδή αν εφαρμόζουν τεχνικές διάδοσης ή σώσιμου κρατουμένου. Η ανάλυση σε αυτό το επίπεδο θα γίνει στο επόμενο κεφάλαιο.

Θεωρώντας ότι οι «μακροσκοπικές» αρχιτεκτονικές θα υλοποιηθούν σε παρόμοιας τεχνολογίας υλικό, η ανάλυση της καθεμίας θα επικεντρωθεί σε τρεις περιοχές:

- Πλήθος βασικών μονάδων και συνεπώς έκταση πάνω στο chip.
- Κρίσιμο μονοπάτι, συνεπώς το throughput της αρχιτεκτονικής.
- Κατανάλωση ισχύος. Από τη θεωρία γνωρίζουμε ότι η (δυναμική) κατανάλωση ισχύος είναι ανάλογη του γινομένου $C \cdot a \cdot f$, όπου C το πλήθος των στοιχείων του κυκλώματος, f η συχνότητα του ρολογιού και a το activity factor, δηλαδή το πόσα, κατά μέσο όρο, σήματα μεταβαίνουν από 0 σε 1 ή το αντίστροφο σε κάθε χρονική στιγμή

Υπάρχουν συνολικά τρεις αλγόριθμοι με τους οποίους πραγματοποιείται ο εν λόγω μετασχηματισμός, τα κύρια γνωρίσματα των οποίων αναφέρουμε συνοπτικά εδώ (περισσότερη ανάλυση στις αντίστοιχες ενότητες παρακάτω):

- Πέρασμα του σήματος προς ανάλυση από ένα ζευγάρι φίλτρων με υποδειγματοληγία ανά δύο στοιχεία στην έξοδο, όπως αναφέραμε και στη θεωρία. Το ζευγάρι αυτό αποτελείται από δύο συμμετρικά FIR φίλτρα τα οποία είναι συγκεκριμένα για κάθε είδος wavelet και οι συντελεστές των οποίων μας δίδονται από τη θεωρία. Επειδή τα φίλτρα υλοποιούν ουσιαστικά την πράξη της συνέλιξης, οι αρχιτεκτονικές που τα υλοποιούν ονομάζονται **αρχιτεκτονικές βασισμένες στη συνέλιξη (convolution-based)**
- Πέρασμα του σήματος προς ανάλυση από ένα ζευγάρι φίλτρων, το οποίο έχει προέλθει από το αντίστοιχο της προηγούμενης μεθόδου. Κάθε ένα από τα νέα δύο φίλτρα αποτελείται από δύο (πάλι συμμετρικά) φίλτρα σε σειρά, εκ των οποίων το πρώτο έχει δε χρειάζεται πολλαπλασιαστές, το δε δεύτερο είναι σημαντικά μικρότερο σε μήκος από το αρχικό. Η τεχνική αυτή βασίζεται στον αλγόριθμο b-spline παραγοντοποίησης, γι' αυτό και οι αρχιτεκτονικές που την υλοποιούν ονομάζονται **αρχιτεκτονικές βασισμένες στη b-spline παραγοντοποίηση**

- Τέλος, μπορούμε να εκμεταλλευτούμε κάποιες ιδιότητες που έχει αυτό το ζευγάρι φίλτρων και να τα ενοποιήσουμε σε ένα ενιαίο, μη διαχωρίσιμο δικτύωμα το οποίο επίσης πραγματοποιεί το μετασχηματισμό. Επειδή αυτό το δίκτυωμα ονομάζεται στη διεθνή βιβλιογραφία *lifting scheme*, οι αρχιτεκτονικές που βασίζονται σε αυτό καλούνται **αρχιτεκτονικές βασισμένες στο lifting scheme**

Στη συνέχεια αυτού του κεφαλαίου αναπτύσσουμε, για κάθε μία από τις τρεις κατηγορίες, εναλλακτικές τοπολογίες που υλοποιούν το μετασχηματισμό, στη σειρά που δοθήκαν παραπάνω. Επιπλέον, επειδή βασικό στοιχείο τόσο των αρχιτεκτονικών που βασίζονται στη συνέλιξη, όσο και αυτών που βασίζονται στη b-spline παραγοντοποίηση, είναι τα συμμετρικά FIR φίλτρα με υποδειγματοληψία ανά δύο στοιχεία στην έξοδο, πριν από όλα αυτά διενεργούμε μια αρκετά θεωρητική ανάλυση των εναλλακτικών αρχιτεκτονικών αυτών των φίλτρων, τα αποτελέσματα της οποίας αξιοποιούμε άμεσα (και εν μέρει εν ίδει παραδείγματος) στη συνέχεια.

B. Συμμετρικά FIR φίλτρα με υποδειγματοληψία στην έξοδο

1. Εισαγωγή

α. Γενικά

Αναφέραμε και πριν, ο υπολογισμός του μετασχηματισμού wavelet ενός σήματος επιτυγχάνεται με τη χρήση ενός ζεύγους καταλλήλων FIR φίλτρων, των οποίων η έξοδος υποδειγματοληπτείται ανά δύο δείγματα. Ειδικότερα, δε, αν το wavelet είναι διορθογώνιο (όπως το CDF 9/7 με το οποίο ασχολούμαστε σε αυτή τη διπλωματική εργασία) το φίλτρο είναι συμμετρικό (αν περιέχει περιττό αριθμό σημείων) ή αντισυμμετρικό (αν περιέχει άρτιο αριθμό σημείων).

Καθώς σκοπός της εργασίας αυτής είναι η ανάδειξη κατάλληλων αρχιτεκτονικών για υλοποίηση του μετασχηματισμού wavelet CDF 9/7, θα ασχοληθούμε μόνο με συμμετρικά φίλτρα, άρτια ή περιττά. Ο λόγος που θα προσπαθήσουμε να διατυπώσουμε γενικότερες σχέσεις για συμμετρικά FIR φίλτρα και όχι για τα δύο συγκεκριμένα φίλτρα 9 και 7 σημείων που χρειαζόμαστε, είναι διότι, όπως θα δούμε και παρακάτω, πολλές φορές είναι σκόπιμο να «σπάσουμε» το αρχικό φίλτρο σε δύο τμήματα, ο αριθμός των σημείων των οποίων μπορεί να είναι οποιοσδήποτε. Επιπλέον, σε πιο προχωρημένες αρχιτεκτονικές (b-spline παραγοντοποίηση) το ζεύγος των φίλτρων 9/7 σημείων ανάγεται σε (συμμετρικά) φίλτρα με μικρότερο αριθμό σημείων, οπότε είναι σκόπιμο να γνωρίζουμε ήδη ποια είναι η βέλτιστη μορφή για ένα γενικό συμμετρικό φίλτρο και απλά να εξειδικεύουμε στον συγκεκριμένο αριθμό σημείων.

Έστω λοιπόν ένα FIR φίλτρο k σημείων. Η έξοδος του φίλτρου αυτού $y(n)$ περιγράφεται συναρτήσσει της εισόδου του $x(n)$ από τη σχέση:

$$y(n) = \sum_{i=0}^k x(n-i) * h_i \quad (2.1)$$

Βλέπουμε λοιπόν ότι για τον υπολογισμό της εξόδου απαιτούνται n πολλαπλασιασμοί, $n-1$ προσθέσεις και η απομνημόνευση των $n-1$ προηγούμενων τιμών του x . Στο χώρο κατάστασης, εξίσωση του φίλτρου γράφεται ως:

$$y(z) = \sum_{i=0}^k h_i * z^{-i} \quad (2.2)$$

Έχουμε ως δεδομένο ότι το φίλτρο μας είναι συμμετρικό. Συνεπώς, ισχύει για κάθε ακέραιο $i \in \left[0, \left\lfloor \frac{k}{2} \right\rfloor\right]$:

$$h_i = h_{k-1-i} \quad (2.3)$$

Παρατηρούμε ότι έχουμε μόλις $\left\lfloor \frac{k}{2} \right\rfloor$ ξεχωριστούς συντελεστές, συνεπώς, με εκμετάλλευση της συμμετρίας, η μέγιστη εξοικονόμηση υλικού που μπορώ να επιτύχω είναι να μειώσω τον αριθμό των πολλαπλασιασμών σε $\left\lfloor \frac{k}{2} \right\rfloor$

β. Η έννοια της υποδειματοληψίας ανά δύο στοιχεία

Όπως προαναφέρθηκε, ο μετασχηματισμός wavelet ενός σήματος δίδεται μέσα από ένα ζευγάρι FIR φίλτρων, των οποίων η έξοδος υποδειματοληπτείται ανά δύο δείγματα. Τα φίλτρα αυτά μερικές φορές ονομάζονται και multirate, αν και ο όρος αυτός είναι πολύ ευρύτερος και περιλαμβάνει όλα τα συστήματα των οποίων διαφορετικά τμήματα λειτουργούν σε διαφορετικές ταχύτητες. Στην περίπτωση μας, αν $y(n)$ η έξοδος ενός FIR φίλτρου που χρησιμοποιείται για το σκοπό αυτό, τότε μας ενδιαφέρουν μόνο οι έξοδοί του για $n=2k$. Αυτό μπορεί να χρησιμοποιηθεί με πολλούς τρόπους για τη βελτιστοποίηση του φίλτρου και τη μείωση των απαιτήσεων σε υλικό.

Παλαιότερα (π.χ. στα [B1], [B2]) η τάση ήταν να χρησιμοποιούνται κανονικά FIR φίλτρα για το μετασχηματισμό, «διπλώνοντας» στο αρχικό ζευγάρι τα επόμενα στάδια. Αυτό, ωστόσο, είναι οριακά μη αποδοτικό: εφ'όσον, όπως αναφέρθηκε στη θεωρία, τα σήματα λεπτομερειών και προσέγγισης κάθε σταδίου παράγονται με το μισό ρυθμό απ'ότι του προηγούμενου σταδίου, αν κάθε χρονική στιγμή που η έξοδος του φίλτρου απορρίπτεται υπολογίζουμε τους συντελεστές του επόμενου σταδίου, η συνολική εκμετάλλευση του υλικού είναι $\sum_{i=1}^n 2^{-i} < 1$, όπου n ο αριθμός των σταδίων

(τα οποία, σε πολλές πρακτικές εφαρμογές, δεν ξεπερνούν τα τρία, π.χ. [B4]). Επιπλέον, η πραγματοποίηση αυτής της τεχνικής απαιτεί πολύπλοκες αρχιτεκτονικές, με σημαντικές απαιτήσεις σε ενδιάμεση μνήμη.

Έτσι, τα τελευταία χρόνια η τάση στην υλοποίηση των μετασχηματισμών wavelet είναι ο σχεδιασμός εξειδικευμένων φίλτρων για τον σκοπό αυτό (δηλαδή, φίλτρα των οποίων τα δείγματα εξόδου παράγονται σε μισό ρυθμό απ'ότι εισέρχονται τα δείγματα εισόδου) και τα επόμενα στάδια του μετασχηματισμού υπολογίζονται από ένα ίδιο ζευγάρι φίλτρων, ή, ακόμα, και από το ίδιο ζευγάρι, αφού ολοκληρωθεί η παραγωγή των σημάτων λεπτομερειών και προσέγγισης του σταδίου αυτού (εν ιδει batch process). Με τον τρόπο αυτόν, επιτυγχάνονται αξιοποίηση του υλικού κατά 100% αλλά και σημαντικά απλούστερες αρχιτεκτονικές.

Ας αναλύσουμε, λοιπόν, ένα τέτοιο φίλτρο. Σύμφωνα με την εξίσωση (2.2.1), το φίλτρο αυτό χρειάζεται k πολλαπλασιαστές, $k-1$ αθροιστές και $k-1$ καταχωρητές. Τα στοιχεία αυτά δουλεύουν σε πλήρη ρυθμό (δηλαδή, όσο είναι ο ρυθμός εισόδου των δειγμάτων του αρχικού σήματος) και παράγουν σε κάθε κύκλο του ρολογιού k

γινόμενα και εκτελούν $k-1$ προσθέσεις. Ωστόσο, λόγω της υποδειγματοληψίας, εμείς χρειαζόμαστε κάθε δύο κύκλους του ρολογιού τέτοιο αριθμό πράξεων, επομένως, υπάρχουν δύο λύσεις για καλύτερη αξιοποίηση του υλικού:

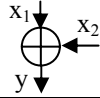
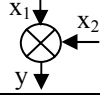
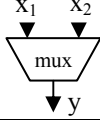
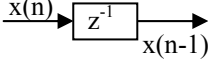
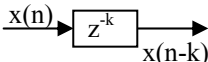
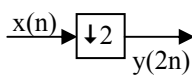
- 1^η Λύση: Κάνουμε τους k πολλαπλασιαστές και τους $k-1$ αθροιστές (το τι γίνεται με τα στοιχεία καθυστέρησης/καταχωρητές θα μελετηθεί παρακάτω) να δουλεύουν στον μισό ρυθμό, έτσι ώστε κάθε δύο κύκλους του ρολογιού να παράγονται και να αθροίζονται k γινόμενα. Η λύση αυτή, αν και δεν προσφέρει εξοικονόμηση στο υλικό, προσφέρει σημαντική εξοικονόμηση ενέργειας στο κύκλωμα, καθώς η κατανάλωση είναι ανάλογη του ρυθμού λειτουργίας
- 2^η Λύση: Μειώνουμε τον αριθμό των πολλαπλασιαστών σε $\left\lceil \frac{k}{2} \right\rceil$ και των αθροιστών σε $\left\lceil \frac{k-1}{2} \right\rceil$ και τους αφήνουμε να δουλεύουν στον ίδιο ρυθμό απ'ότι αρχικά, δηλαδή, να εκτελούν μια πράξη ανά κύκλο ρολογιού. Για να επιτευχθεί αυτό, συχνά χρειάζεται «πολύπλεξη» των διαδοχικών πράξεων στο ίδιο στοιχείο, μια τεχνική γνωστή και ως folding.

Προτού προχωρήσουμε παρακάτω και μελετήσουμε τρόπους πρακτικής εφαρμογής των παραπάνω στις διάφορες μορφές των φίλτρων, θα παρουσιάσουμε τα σύμβολα που θα χρησιμοποιηθούν στα επόμενα σχήματα, καθώς και ορισμένα σημαντικά αξιώματα των κυκλωματικών διαγραμμάτων.

γ. Σύμβολα και αξιώματα κυκλωματικών διαγραμμάτων

Στη διπλωματική αυτή εργασία, χρησιμοποιούμε τα εξής σύμβολα:

Πίνακας 2.1 Σύμβολα κυκλωματικών διαγραμμάτων

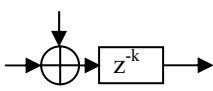
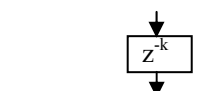
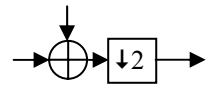
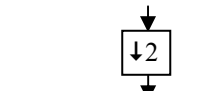
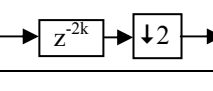
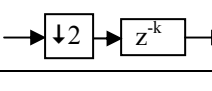
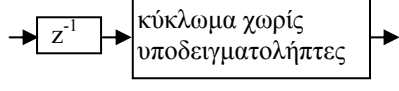

Επεξήγηση	Σύμβολο
αθροιστής με εισόδους x_1 και x_2 και έξοδο y	
πολλαπλασιαστής με εισόδους x_1 και x_2 και έξοδο y	
πολυπλέκτης δύο εισόδων	
καταχωρητής (στοιχείο καθυστέρησης)	
k διαδοχικοί καταχωρητές	
Υποδειγματολήπτης ανά δύο δείγματα	

Επιπλέον, σε όλα τα σχήματα ακολουθούμε τις παρακάτω συμβάσεις, όπου n η σειρά άφιξης των δειγμάτων εισόδου:

1. Οι υποδειματολήπτες ανά δύο δείγματα αλλάζουν την τιμή της εξόδου τους μόνο όταν το n (χρονική στιγμή άφιξης δείγματος) είναι άρτιο: οι τιμές στην είσοδό τους που έρχονται για περιττά n απορρίπτονται
2. Στους πολυπλέκτες θεωρούμε ότι η είσοδος που βρίσκεται **αριστερά** ή **πάνω** επιλέγεται για n άρτιο, αντίθετα ή είσοδος που βρίσκεται **δεξιά** ή **κάτω** επιλέγεται για n περιττό
3. Ο ρυθμός λειτουργίας ενός στοιχείου (δηλαδή, κάθε πότε αλλάζει την έξοδό του αν είναι καταχωρητής ή κάθε πόσους κύκλους ρολογιού εκτελεί μια πράξη) εξαρτάται **μόνο** από το ρυθμό άφιξης των δεδομένων εισόδου σε αυτό. Συνεπώς, στοιχεία που βρίσκονται μετά από κάποιον υποδειματολήπτη στο γράφο σηματοροής, θεωρείται ότι δουλεύουν στο μισό ρυθμό (και αυτό δε θα επισημαίνεται με άλλον τρόπο)

Επιπλέον, στα κυκλωματικά διαγράμματα ισχύουν και τα ακόλουθα αξιώματα:

Πίνακας 2.2 Βασικά αξιώματα κυκλωματικών διαγραμμάτων

ισοδύναμες μορφές		
		(2.4)
		(2.4b)
		(2.4c)
		(2.4d)

Ορισμένες παρατηρήσεις επί των παραπάνω αξιωμάτων:

- Τα αξιώματα (2.4a) και (2.4b) ισχύουν και για την περίπτωση πολλαπλασιαστή στη θέση του αθροιστή.
- Εάν ο πολλαπλασιαστής έχει σταθερό συντελεστή, τότε προφανώς ο υποδειματολήπτης/καταχωρητής δε χρειάζεται να μπει πριν τον συντελεστή αυτόν (αφού είναι σταθερός)
- Τα αξιώματα (2.4b) και (2.4c) μας δίνουν τους μόνους επιτρεπτούς τρόπους που υπάρχουν για να μειώσουμε τον ρυθμό λειτουργίας ενός στοιχείου του κυκλώματος στο μισό. Συνεπώς, εάν θέλουμε να αυξήσουμε την αποδοτικότητα του κυκλώματος κρατώντας τον ίδιο αριθμό στοιχείων, αλλά μειώνοντας το ρυθμό λειτουργίας τους, θα πρέπει απαραίτητα να χρησιμοποιήσουμε κάποια από αυτές τις ισοδυναμίες

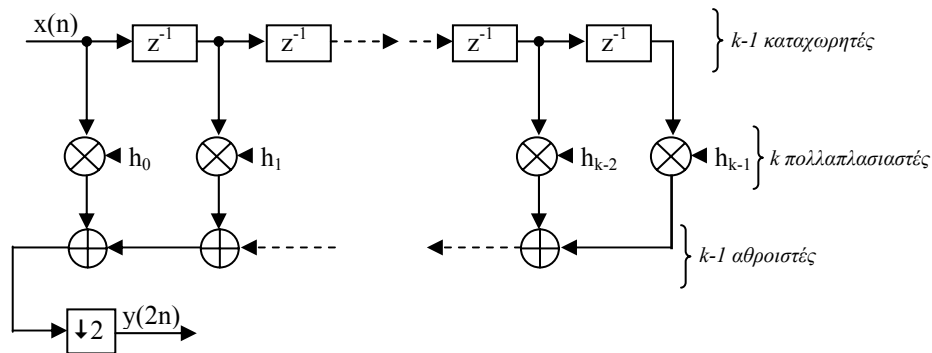
- Στο αξίωμα (2.4d) το γενικό κύκλωμα δεν πρέπει να περιέχει υποδειματολήπτες, έτσι ώστε όλα τα στοιχεία του να δουλεύουν στην ίδια ταχύτητα.

Στη συνέχεια του κεφαλαίου αυτού, βάσει των παραπάνω απαιτήσεων και αρχών, θα αναζητήσουμε τις αποδοτικότερες μορφές για συμμετρικά FIR φίλτρα με υποδειματοληψία ανά δύο στην έξοδο

2. Κανονική μορφή

α. Γενικά

Η κανονική (direct) μορφή ενός γενικού FIR φίλτρου k σημείων με υποδειματοληψία ανά δύο δείγματα στην έξοδο προκύπτει ως φυσικό επακόλουθο της σχέσης (2.2.2) όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 2.1 Direct μορφή γενικού φίλτρου

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.5)

- Πλήθος στοιχείων: $k-1$ καταχωρητές, $k-1$ αθροιστές, k πολλαπλασιαστές, 1 υποδειματολήπτης. **Ο υποδειματολήπτης δε θα λαμβάνεται υπ' όψιν στις αναλύσεις ισχύος και κρίσιμο μονοπατιού.**
- Αν T_A η καθυστέρηση που εισάγει ένας αθροιστής, T_M ένας πολλαπλασιαστής, τότε, το κρίσιμο μονοπάτι ισούται με $T_M + (k-1)T_A$
- Αν P_M η (μέση) ισχύς που καταναλώνει ένας πολλαπλασιαστής, P_A ένας αθροιστής και P_R ένας καταχωρητής **όταν δουλεύουν σε ρυθμό ίσο με το ρυθμό άφιξης των $x(n)$** , η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται ισούται με $kP_M + (k-1)(P_A + P_R)$

Εφ' όσον στην ασχολούμαστε με συμμετρικά φίλτρα, μπορούμε, ομαδοποιώντας τα γινόμενα που έχουν τον ίδιο συντελεστή, να μειώσουμε τον αριθμό των πολλαπλασιαστών σε $\left\lfloor \frac{k}{2} \right\rfloor$. Πράγματι, με χρήση της (2.3), η (2.1) γράφεται, ανάλογα με το αν το k είναι περιττό ή άρτιο, ως:

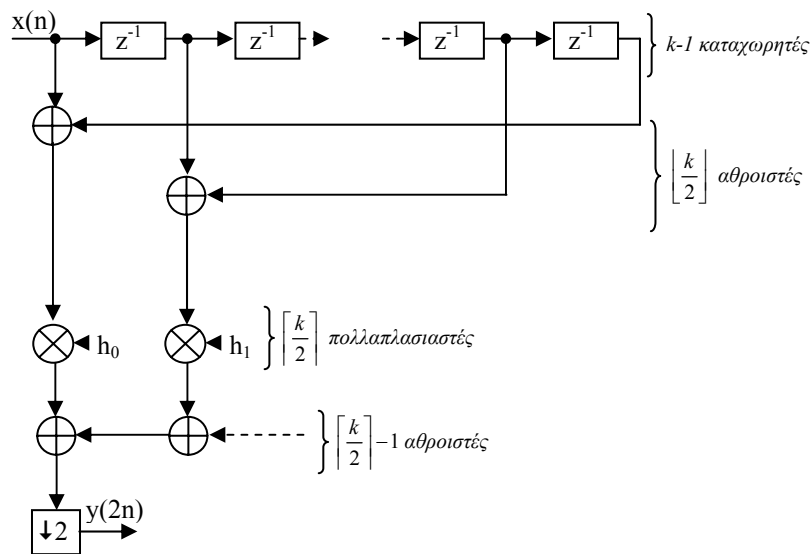
$$\text{για άρτιο: } y(n) = \sum_{i=0}^{\frac{k}{2}-1} h_i * [x(n-i) + x(n-k+1+i)] \quad (2.6a)$$

$$\text{για περιττό: } y(n) = \sum_{i=0}^{\left\lfloor \frac{k}{2} \right\rfloor - 1} h_i * [x(n-i) + x(n-k+1+i)] + h_{\left\lfloor \frac{k}{2} \right\rfloor} * x\left(n - \left\lfloor \frac{k}{2} \right\rfloor\right) \quad (2.6b)$$

Το πλήθος των πράξεων σε κάθε περίπτωση ισούται με:

- Στην (2.6a) έχω $\frac{k}{2} + \left(\frac{k}{2} - 1\right) = k - 1$ προσθέσεις, $\frac{k}{2}$ πολλαπλασιασμούς και χρειάζεται απομνημόνευση των $(k-1)$ προηγούμενων στοιχείων της εισόδου
- Στην (2.6b) έχω $\left\lfloor \frac{k}{2} \right\rfloor + \left(\left\lfloor \frac{k}{2} \right\rfloor - 1 \right) + 1 = 2 \left\lfloor \frac{k}{2} \right\rfloor = k - 1$ προσθέσεις, $\left\lfloor \frac{k}{2} \right\rfloor + 1 = \left\lfloor \frac{k}{2} \right\rfloor$ πολλαπλασιασμούς και χρειάζεται απομνημόνευση των $(k-1)$ προηγούμενων στοιχείων της εισόδου

Άρα, σε κάθε περίπτωση, με ομαδοποίηση των γινομένων με τον ίδιο συντελεστή, μειώνω τους απαιτούμενους πολλαπλασιασμούς σε $\left\lfloor \frac{k}{2} \right\rfloor$. Συνεπώς, με άμεση εφαρμογή των παραπάνω, έχω το παρακάτω σχήμα:



Σχήμα 2.2 Εκμετάλλευση της συμμετρικότητας των συντελεστών

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.7)

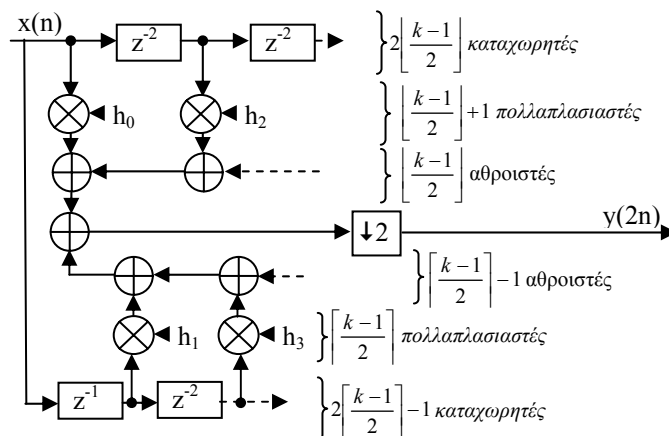
- Πλήθος στοιχείων: $k-1$ καταχωρητές, $k-1$ αθροιστές, $\left\lfloor \frac{k}{2} \right\rfloor$ πολλαπλασιαστές, 1 υποδειγματολήπτης.
- Αν T_A η καθυστέρηση που εισάγει ένας αθροιστής, T_M ένας πολλαπλασιαστής, τότε, το κρίσιμο μονοπάτι ισούται με $T_A + T_M + \left(\left\lfloor \frac{k}{2} \right\rfloor - 1 \right) T_A = T_M + \left\lfloor \frac{k}{2} \right\rfloor T_A$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται ισούται με $\left\lfloor \frac{k}{2} \right\rfloor P_M + (k-1)(P_A + P_R)$

Παρά τη σχετική βελτίωση, υπενθυμίζουμε ότι λόγω της υποδειγματοληψίας ανά δύο στην έξοδο του φίλτρου, ο βαθμός χρήσης του υλικού ισούται (ακόμα) με 50%. Συνεπώς, θα πρέπει να εφαρμόσω μεθόδους αξιοποίησης του χαρακτηριστικού αυτού.

β. Μείωση του ρυθμού λειτουργίας των στοιχείων του φίλτρου

Όπως αναφέρθηκε και στην εισαγωγή του κεφαλαίου αυτού, ο ένας τρόπος να αποφευχθεί η κατασπατάληση πόρων που δημιουργεί ο υποδειματολήπτης στην έξοδο του κυκλώματος είναι να αναδιατάξω το κύκλωμα, έτσι ώστε τα εσωτερικά του εξαρτήματα να δουλεύουν πλέον στον μισό ρυθμό. Κατ' ουσίαν, αυτό που σκοπεύουμε να κάνουμε είναι να μετακινήσουμε τον υποδειματολήπτη στην αρχή του κυκλώματος, από το τέλος στο οποίο βρίσκεται.

Παρατηρούμε ότι το φίλτρο αποτελείται από αθροιστές, πολλαπλασιαστές και στοιχεία καθυστέρησης. Από τις (2.4b) και (2.4c) βλέπω ότι ενώ ο υποδειματολήπτης μπορεί ελεύθερα να μετακινηθεί από την έξοδο στις εισόδους των πολλαπλασιαστών και των αθροιστών, μπορεί να μετακινηθεί μόνο από την έξοδο άρτιου αριθμού καταχωρητών στην είσοδό τους, μειώνοντας τον αριθμό τους στο μισό. Επομένως, θα πρέπει το φίλτρο του σχήματος (2.1) να αναδιαταχθεί με τέτοιο τρόπο έτσι ώστε μεταξύ των διαφόρων σημάτων να υπάρχουν μόνο ζευγάρια καταχωρητών. Είναι φανερό ότι ο **μόνος** τρόπος να πραγματοποιηθεί αυτό είναι να χωρίσω τα σημεία του φίλτρου σε αυτά που είναι άρτιου και αυτά που είναι περιττού βαθμού, όπως φαίνεται και στο παρακάτω σχήμα:

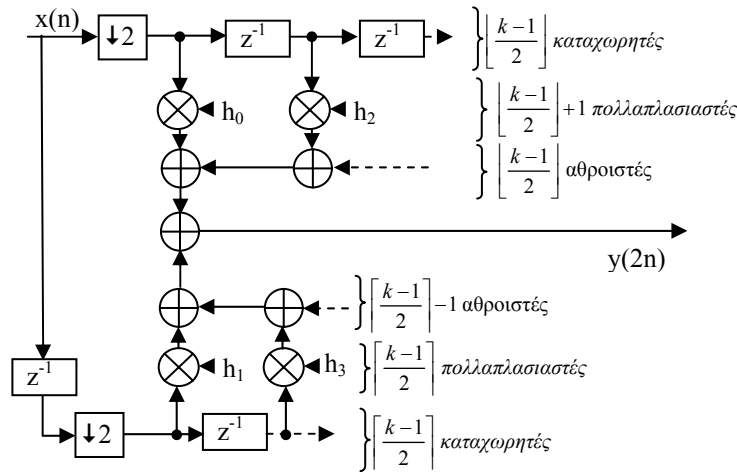


Σχήμα 2.3α Διαχωρισμός σε άρτια-περιττά

Απόδειξη αριθμού καταχωρητών στο σχ. 2.3α: σύμφωνα με το σχ. 2.2, ο αριθμός των καταχωρητών είναι συνολικά $m=k-1$. Για τον διαχωρισμό σε άρτια-περιττά σημεία, ψάχνω τους μέγιστους δυνατούς f και g (άρτιος και περιττός, αντίστοιχα), τέτοιους ώστε να είναι μικρότεροι ή ίσοι του m . Ισχύει:

- f άρτιος, άρα, $f = 2 \lfloor \frac{m}{2} \rfloor = 2 \lfloor \frac{k-1}{2} \rfloor$, άρα, $2 \lfloor \frac{k-1}{2} \rfloor$ καταχωρητές στην «άρτια γραμμή»
- g περιττός, άρα, $g = 2 \lfloor \frac{m}{2} \rfloor - 1 = 2 \lfloor \frac{k-1}{2} \rfloor - 1$, άρα, $2 \lfloor \frac{k-1}{2} \rfloor - 1$ καταχωρητές στην «περιττή» γραμμή
- Ο αριθμός των πολλαπλασιαστών κα κάθε «γραμμή» προκύπτει άμεσα από την θέση των συντελεστών στο σχ. 2.2.2

Οπότε, με μετακίνηση των υποδειματοληπτών σύμφωνα με τις (2.4b), (2.4c) παίρνουμε:



Σχήμα 2.3b Μετακίνηση υποδειματοληπτών

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής (2.8)

- Πλήθος στοιχείων: $\left\lfloor \frac{k-1}{2} \right\rfloor + \left\lfloor \frac{k-1}{2} \right\rfloor = k-1$ καταχωρητές, $\left\lfloor \frac{k-1}{2} \right\rfloor + \left\lfloor \frac{k-1}{2} \right\rfloor - 1 + 1 = k-1$

αθροιστές, $\left\lfloor \frac{k-1}{2} \right\rfloor + 1 + \left\lfloor \frac{k-1}{2} \right\rfloor = k$ πολλαπλασιαστές, 2 υποδειματολήπτες.

- Το κρίσιμο μονοπάτι ισούται με $T_M + \left[\max \left(\left\lfloor \frac{k-1}{2} \right\rfloor - 1, \left\lfloor \frac{k-1}{2} \right\rfloor \right) + 1 \right] T_A = T_M + \left(\left\lfloor \frac{k-1}{2} \right\rfloor + 1 \right) T_A$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται να ισούται με $k \frac{P_M}{2} + (k-2) \frac{P_R}{2} + P_R + (k-1) \frac{P_A}{2} = k \left(\frac{P_M}{2} + \frac{P_R}{2} \right) + (k-1) \frac{P_A}{2}$, αφού η κατανάλωση ισχύος ενός κυκλώματος είναι ανάλογη της συχνότητας λειτουργίας του.

Παρατηρούμε στο σχ. 2.3b ότι στην ουσία χωρίσαμε το αρχικό (direct μορφής) FIR φίλτρο σε δύο μικρότερα (direct μορφής) FIR φίλτρα μικρότερου μεγέθους, ένα με τους όρους άρτιου βαθμού και ένα με τους όρους περιττού, τα οποία λειτουργούν στη μισή συχνότητα από το αρχικό. Στη διεθνή βιβλιογραφία η τεχνική αυτή ονομάζεται **polyphase decomposition** (πολυφασικός διαχωρισμός) και χρησιμοποιείται ευρύτατα σε μετασχηματισμούς wavelet.

Ωστόσο, ο χωρισμός κατά αυτόν τον τρόπο δεν είναι αναγκαστικός. Στην ουσία, στο σχ. 2.3b έχουμε δύο ξεχωριστές γραμμής παραγωγής γινομένων τα αποτελέσματα των οποίων είμαστε ελεύθεροι να τα αθροίσουμε ή/και ομαδοποιήσουμε με όποιον τρόπο/σειρά θέλουμε λόγω των ιδιοτήτων της πράξης της πρόσθεσης (2.5). Προφανώς, αφού οι συντελεστές του αρχικού φίλτρου είναι συμμετρικοί, έχει νόημα να επιδιώξουμε τέτοια οργάνωση των γινομένων έτσι ώστε να γίνεται εκμετάλλευση της συμμετρίας και στη μορφή αυτή (αφού η συμμετρία στην direct μορφή γίνεται εκμεταλλεύσιμη με πρόσθεση των ιδίων όρων πριν τον πολλαπλασιασμό, όπως δείξαμε στις (2.6a) και (2.6b)

Ας δούμε, λοιπόν, τι γίνεται με το θέμα της συμμετρίας, ανάλογα με το αν το k είναι περιττό ή άρτιο.

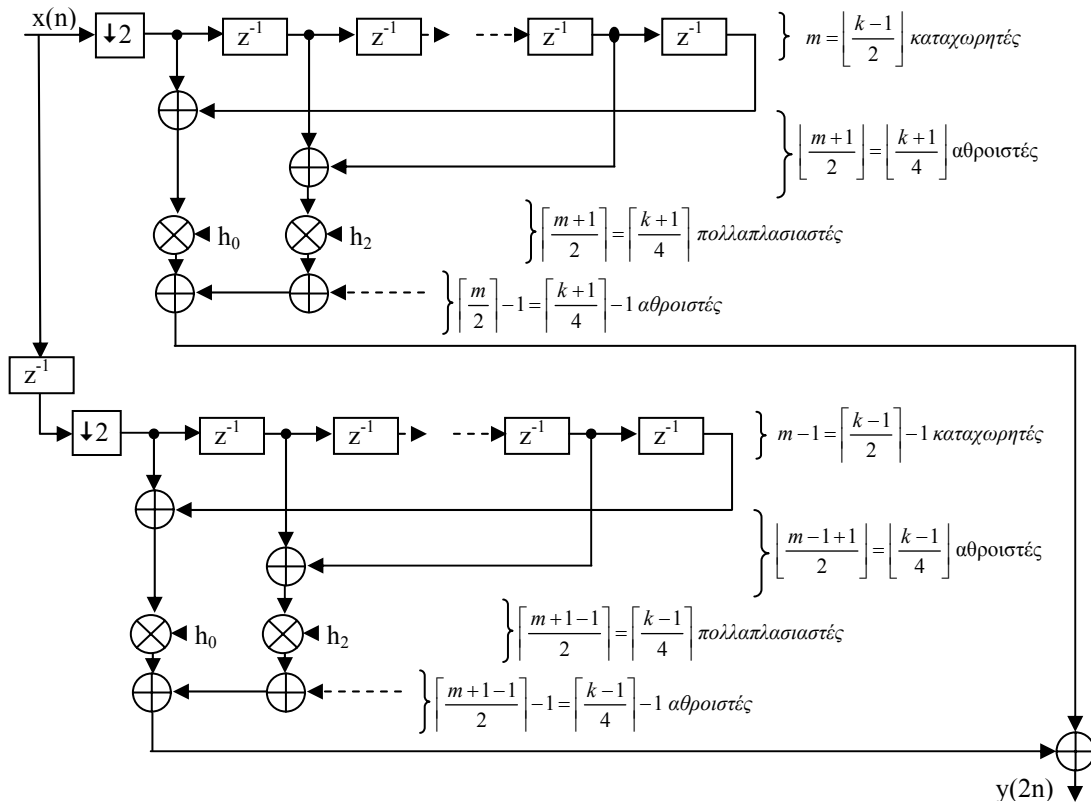
- Αν k είναι περιττός, ισχύει $k=2m+1$. Τότε, η 2.3 ξαναγράφεται ως:

$$\forall i \in [0, m]: h_i = h_{2m-i} \quad (2.9a)$$

Άρα, οι περιττού βαθμού συντελεστές είναι ίσοι με περιττού βαθμού συντελεστές και οι άρτιου βαθμού με άρτιου βαθμού. Αυτό σημαίνει ότι το «περιττό» και το «άρτιο» υποφίλτρο στο σχ. 2.3b είναι και αυτά συμμετρικά, συνεπώς, μπορούμε άμεσα σε αυτά να εφαρμόσουμε τις σχέσεις (2.6a) και (2.6b) προκειμένου, μέσω της πρόσθεσης των εκδοχών του x που θα πολλαπλασιαστούν με τον ίδιο συντελεστή, να επιτύχουμε μείωση των πολλαπλασιαστών. Από τη στιγμή που σύμφωνα με τις (2.5), (2.7) σε φίλτρο k σημείων ο αριθμός των πολλαπλασιαστών μειώνεται από k σε $\left\lfloor \frac{k}{2} \right\rfloor$ και το

κρίσιμο μονοπάτι από

$T_M + (k-1)T_A$ σε $T_M + \left\lfloor \frac{k}{2} \right\rfloor T_A$, έχω την παρακάτω αρχιτεκτονική:



Σχήμα 2.4a Συμμετρία και μείωση ταχύτητας στοιχείων (polyphase decomposition) σε περιττά φίλτρα

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής (2.10)

- Πλήθος στοιχείων: $\left\lfloor \frac{k-1}{2} \right\rfloor + \left\lfloor \frac{k-1}{2} \right\rfloor - 1 + 1 = k - 1$ καταχωρητές,

$$\left\lfloor \frac{k+1}{4} \right\rfloor + \left\lfloor \frac{k+1}{4} \right\rfloor - 1 + \left\lfloor \frac{k-1}{4} \right\rfloor + \left\lfloor \frac{k-1}{4} \right\rfloor - 1 + 1 = \frac{k+1}{2} + \frac{k-1}{2} - 1 = k - 1 \text{ αθροιστές,}$$

$$\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m+1}{2} \right\rfloor = \left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{2} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor + 1 = m + 1 = \left\lfloor \frac{k}{2} \right\rfloor \text{ πολλαπλασιαστές, 2}$$

υποδειγματολήπτες.

- Το κρίσιμο μονοπάτι ισούται με $T_A + T_M + \left(\left\lfloor \frac{m+1}{2} \right\rfloor - 1\right)T_A + T_A = T_M + \left(\left\lfloor \frac{k+1}{4} \right\rfloor + 1\right)T_A$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται να ισούται με $\left\lfloor \frac{k}{2} \right\rfloor \frac{P_M}{2} + (k-2) \frac{P_R}{2} + P_R + (k-1) \frac{P_A}{2} = \left\lfloor \frac{k}{2} \right\rfloor \frac{P_M}{2} + k \frac{P_R}{2} + (k-1) \frac{P_A}{2}$

Βλέπουμε, λοιπόν, ότι σε περίπτωση περιττού φίλτρου, επιτυγχάνουμε σημαντική βελτίωση όλων των παραμέτρων με εκμετάλλευση της υποδειγματοληψίας στην έξοδο και της συμμετρικότητας των συντελεστών. Ας δούμε τώρα τι συμβαίνει στην περίπτωση του αρτίου φίλτρου

- **Αν k άρτιος**, τότε, $k=2m$, και η 2.3 ξαναγράφεται ως:

$$\forall i \in [0, m]: h_i = h_{2m-1-i} \quad (2.9b)$$

Αυτό σημαίνει ότι περιττής τάξης συντελεστές είναι ίσοι με άρτιας τάξης συντελεστές και αντίστροφα, δηλαδή, τα δύο φίλτρα του σχ. (2.3b) δεν είναι συμμετρικά. Αυτό, βέβαια, δε θα μας εμποδίσει να εκμεταλλευτούμε τη συμμετρία: όπως προαναφέρθηκε, η οργάνωση των αθροισμάτων των γινομένων σε δύο φίλτρα δεν είναι ο μοναδικός τρόπος που μπορούμε να ομαδοποιήσουμε τις προσθέσεις.

Συμμετρία σημαίνει (2.9b) ότι ο πρώτος συντελεστής (άρτια τάξη) είναι συμμετρικός με τον τελευταίο (περιττή τάξη), ο δεύτερος (περιττή τάξη) με τον προτελευταίο (άρτια τάξη) κλπ. Επιπλέον, βλέπουμε ότι η εκμετάλλευση της συμμετρίας της direct μορφής στο σχ. 2.2 θα είναι η ίδια στην περίπτωση που οι ίδιοι συντελεστές τροφοδοτούνται από διαφορετικά σήματα, π.χ. αν ισχύει:

$$y(n) = \sum_{i=0}^{l-1} x_1(n-i) * h_i + \sum_{i=0}^{l-1} x_2(n-i) * h_{l-1-i} \quad (2.11)$$

μπορούμε και πάλι με τον ίδιο τρόπο να μειώσουμε τους πολλαπλασιασμούς από 2l σε l.

Αυτό ακριβώς θα εφαρμόσουμε και στην περίπτωσή μας: από τα γινόμενα το σχ. 2.3b θα δημιουργήσουμε δύο φίλτρα «δύο εισόδων» που ορίζονται από τις παρακάτω σχέσεις (υπενθυμίζουμε το αρχικό φίλτρο είναι $k=2m$ σημείων και γι' αυτό ισχύουν οι (2.1), (2.9b)):

$$\begin{aligned} y_1(n) &= \sum_{i=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_1(n-i) * h_{2i} + \sum_{j=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_2(n-j) * h_{2j+2m-1-2\left(\left\lfloor \frac{m}{2} \right\rfloor - 1\right)} = \\ &= \sum_{i=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_1(n-i) * h_{2i} + \sum_{j=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_2(n-j) * h_{2\left\lfloor \frac{m}{2} \right\rfloor + 1 + 2j} \end{aligned} \quad (2.12a)$$

$$\begin{aligned} y_2(n) &= \sum_{i=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_3(n-i) * h_{2i+1} + \sum_{j=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_4(n-j) * h_{2m-2-2\left(\left\lfloor \frac{m}{2} \right\rfloor - 1\right) + 2j} = \\ &= \sum_{i=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_3(n-i) * h_{2i+1} + \sum_{i=0}^{\left\lfloor \frac{m}{2} \right\rfloor - 1} x_4(n-i) * h_{2\left\lfloor \frac{m}{2} \right\rfloor + 2j} \end{aligned} \quad (2.12b)$$

όπου,

$$x_2 = x_3 \left(n - \left\lfloor \frac{m}{2} \right\rfloor \right) \quad (2.12c)$$

$$x_4 = x_1 \left(n - \left\lfloor \frac{m}{2} \right\rfloor \right) \quad (2.12d)$$

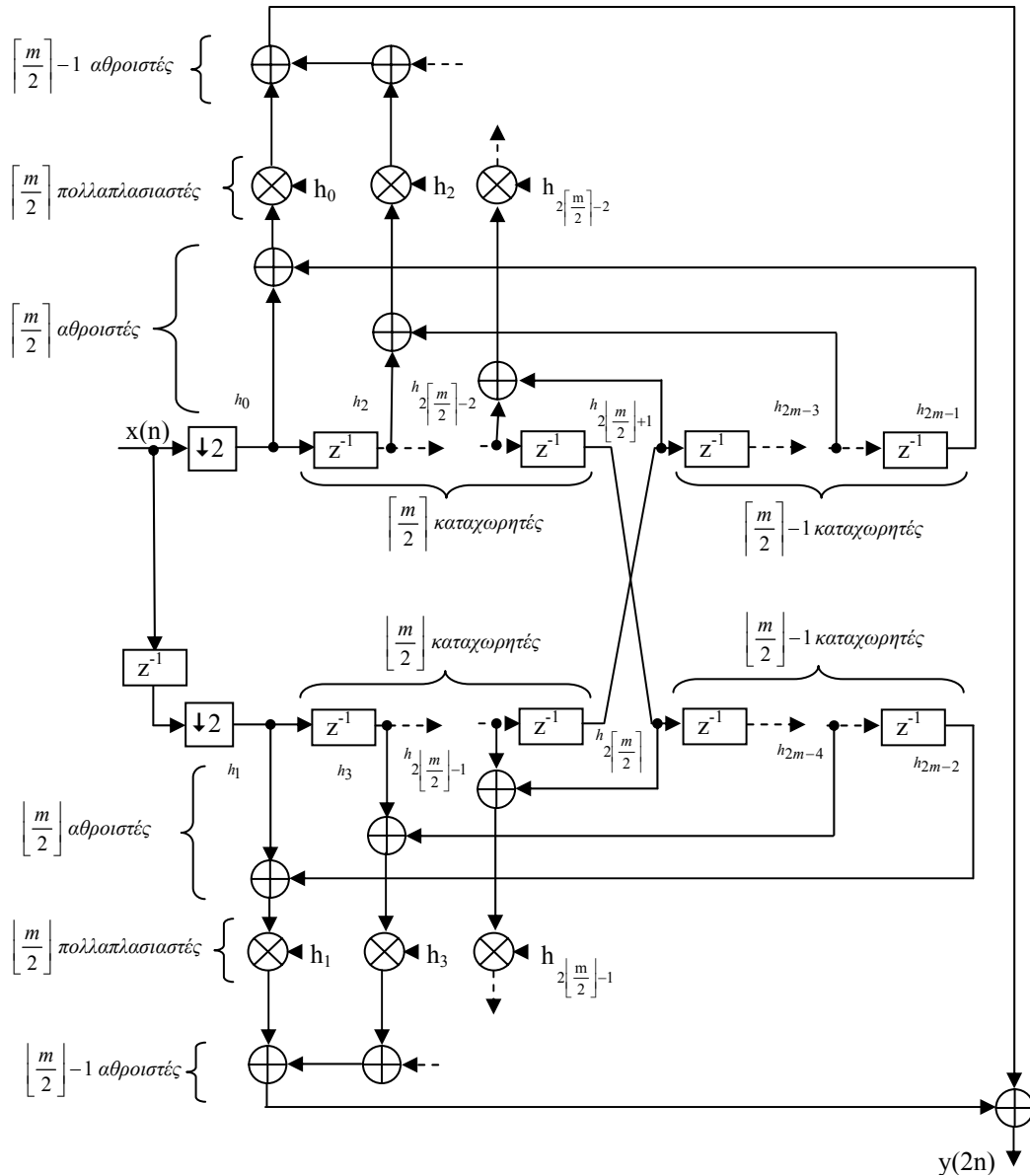
Εάν στο x_1 θέσω τις άρτιες (σε χρόνο άφιξης) εισόδους x και στο x_3 τις περιττές, παρατηρώ ότι το άθροισμα y_1+y_2 περιέχει όλα τα γινόμενα του σχ. 2.3b. Συνεπώς, τα δύο «φίλτρα δύο εισόδων» παράγουν την έξοδο του αρχικού φίλτρου. Επιπλέον, στις (2.12a), (2.12b) παρατηρώ ότι σε κάθε σχέση για ζευγάρια (i, j) της μορφής $\left(u, \left\lfloor \frac{m}{2} \right\rfloor - u - 1 \right)$ και $\left(u, \left\lfloor \frac{m}{2} \right\rfloor - u - 1 \right)$, αντίστοιχα, (τα οποία κάθε u που παίρνει τις τιμές του i ορίζει μονοσήμαντα) οι αντίστοιχοι συντελεστές εντός των αθροισμάτων είναι ίδιοι. Πράγματι, στην (2.12a) έχω ότι:

$$h_{2i} = h_{2u} \text{ και } h_{2\left\lfloor \frac{m}{2} \right\rfloor + 1 + 2j} = h_{2\left\lfloor \frac{m}{2} \right\rfloor + 1 + 2\left\lfloor \frac{m}{2} \right\rfloor - 2u - 2} = h_{2m-1-2u} = h_{2u}$$

(από την (2.9b) και στην (2.12b) έχω ότι:

$$h_{2i+1} = h_{2u+1} \text{ και } h_{2\left\lfloor \frac{m}{2} \right\rfloor + 2j} = h_{2\left\lfloor \frac{m}{2} \right\rfloor + 2\left\lfloor \frac{m}{2} \right\rfloor - 2u - 2} = h_{2m-1-(2u+1)} = h_{2u+1}.$$

Δηλαδή, τα y_1, y_2 ικανοποιούν την (2.11), οπότε, μπορούμε άμεσα να εκμεταλλευτούμε τη συμμετρικότητα των συντελεστών του αρχικού φίλτρου, όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 2.4b Συμμετρία και polyphase decomposition σε άρτια φίλτρα

Σημείωση: στο παραπάνω σχήμα, για λόγους διευκόλυνσης, παρατίθενται στα αριστερά των χρονικών μετατοπίσεων του $x(n)$ ο συντελεστής με τον οποίο το κάθε ένα θα πολλαπλασιαστεί.

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής (2.13)

- Πλήθος στοιχείων: $\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor - 1 + \left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor - 1 + 1 = 2m - 1 = k - 1$ καταχωρητές,

$$\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor - 1 + \left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor - 1 + 1 = 2m - 1 = k - 1 \text{ αθροιστές,}$$

$$\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor = m = \frac{k}{2} = \left\lfloor \frac{k}{2} \right\rfloor \text{ πολλαπλασιαστές, } 2 \text{ υποδειγματολήπτες.}$$

- Το κρίσιμο μονοπάτι ισούται με $T_A + T_M + \max\left(\left\lfloor \frac{m}{2} \right\rfloor - 1, \left\lfloor \frac{m}{2} \right\rfloor - 1\right) T_A + T_A =$
 $= T_M + \left(\left\lfloor \frac{k}{4} \right\rfloor + 1\right) T_A$

- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται να ισούται με
$$\left\lceil \frac{k}{2} \right\rceil \frac{P_M}{2} + (k-2) \frac{P_R}{2} + P_R + (k-1) \frac{P_A}{2} = \left\lceil \frac{k}{2} \right\rceil \frac{P_M}{2} + k \frac{P_R}{2} + (k-1) \frac{P_A}{2}$$

Συμπερασματικά, βλέπουμε, λοιπόν, ότι μπορούμε να εκμεταλλευτούμε αποδοτικά τη συμμετρία και την υποδειγματοληψία και σε άρτια φίλτρα, μόνο που εδώ δεν διαχωρίζουμε απλώς το αρχικό φίλτρο σε δύο μικρότερα. Επιπλέον, βλέπουμε ότι συναρτήσει του k , τα αποτελέσματα για άρτια και περιττά φίλτρα διαφέρουν μόνο στο θέμα του κρίσιμου μονοπατιού

Συνολικά, από τις (2.12) και (2.13), οι επιδόσεις ενός συμμετρικού FIR φίλτρου με υποδειγματοληψία στην έξοδο, αν επιχειρηθεί η ως άνω μείωση του ρυθμού λειτουργίας των στοιχείων του, είναι οι εξής (2.14)

-Στοιχεία: $k-1$ καταχωρητές, $k-1$ αθροιστές, $\left\lceil \frac{k}{2} \right\rceil$ πολλαπλασιαστές, 2

υποδειγματολήπτες

-Κρίσιμο μονοπάτι: $T_M + \left(\left\lceil \frac{k-1}{4} \right\rceil + 1 \right) T_A$, αν k περιττό, αλλιώς

$T_M + \left(\left\lceil \frac{k}{4} \right\rceil + 1 \right) T_A$ αν k άρτιο

-Αναμενόμενη κατανάλωση ισχύος: $\left\lceil \frac{k}{2} \right\rceil \frac{P_M}{2} + k \frac{P_R}{2} + (k-1) \frac{P_A}{2}$

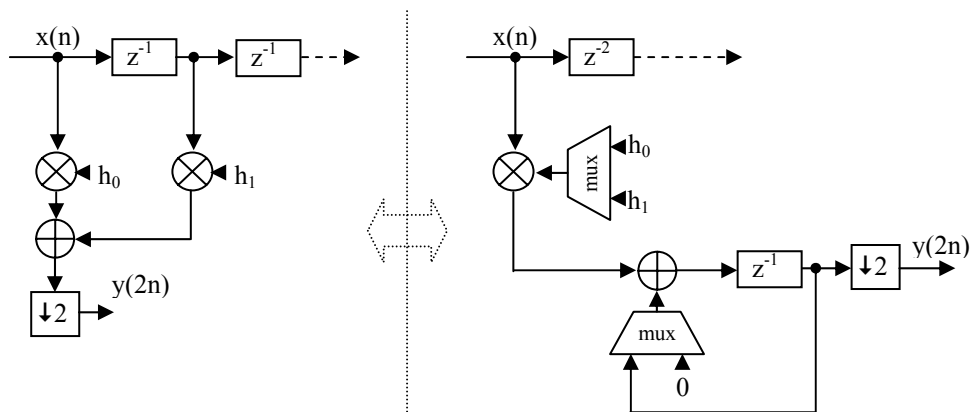
γ. Folding

Όπως αναφέρθηκε και πιο πριν, ο άλλος τρόπος εκμετάλλευσης της υποδειγματοληψίας στην έξοδο (δηλαδή, του γεγονότος ότι εμείς θέλουμε να δημιουργηθούν και να προστεθούν k γινόμενα σε 2 κύκλους ρολογιού και όχι μόνο σε έναν, για το οποίο είναι ικανό οποιοδήποτε γενικό FIR φίλτρο) είναι να μην μειώσουμε την ταχύτητα λειτουργίας των στοιχείων του, αλλά να μειώσουμε τον αριθμό τους (π.χ. των πολλαπλασιαστών από k σε $\left\lceil \frac{k}{2} \right\rceil$) έτσι ώστε σε κάθε κύκλο του

ρολογιού να δημιουργούνται και να προστίθενται το πολύ $\left\lceil \frac{k}{2} \right\rceil$ γινόμενα.

Ουσιαστικά, η ιδέα αυτή δεν είναι τίποτα άλλο από τη δίπλωση στο χρόνο των ίδιων τάξης συντελεστών των δύο «υποφίλτρων» του σχ. 2.4a ή 2.4b, και διατυπώθηκε για πρώτη φορά το 2004 από τους McCanny et. al. στο [B4]. Εδώ, θα ακολουθήσουμε έναν διαφορετικό, πιο γενικό τρόπο για την ανάλυση αυτής της μεθόδου και παράλληλα θα προχωρήσουμε στην απόδειξη των αντίστοιχων σχέσεων για το πλήθος βασικών μονάδων/κρίσιμο μονοπάτι/κατανάλωση ισχύος όπως και στην περίπτωση της μείωσης του ρυθμού λειτουργίας. Επιπλέον, θα ξεκινήσουμε από το φίλτρο του σχ. 2.2 που ήδη κάνει εκμετάλλευση της συμμετρίας, παρά από τη γενικότερη μορφή όπως πριν.

Ας δούμε, πρώτα, τον τρόπο με τον οποίον μπορούμε να πολυπλέξουμε στο χρόνο δύο γειτονικά σημεία σε ένα (multirate FIR) φίλτρο:

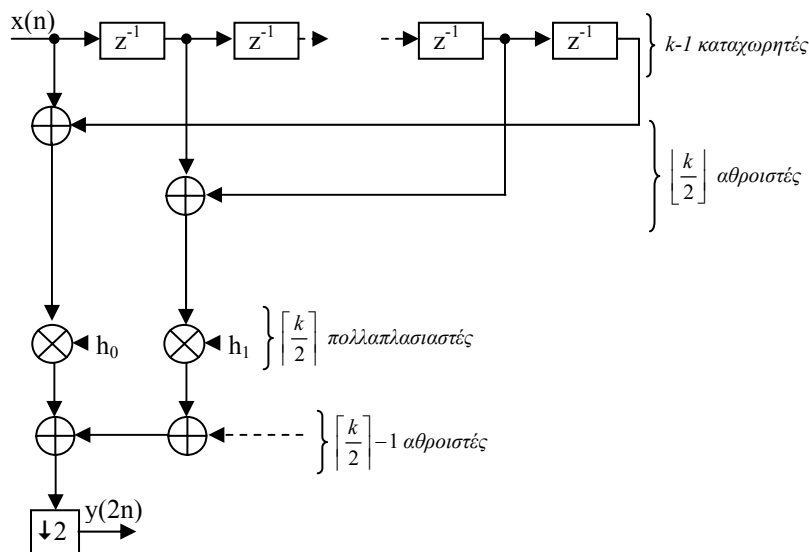


Σημείωση: στο δεξί σχήμα:

- Πρώτα (χρονικά) υπολογίζεται το γινόμενο του h_1 με το $x(n-1)$ και μετά του h_0 με το $x(n)$
- Ο υποδειγματολήπτης αλλάζει την έξοδο του στην είσοδο μόνο στις περιττές χρονικές στιγμές (περιττή φάση). Όπως έχει προαναφερθεί, συνήθως (και όπου δεν αναφέρεται) οι υποδειγματολήπτες είναι άρτιας φάσης

Σχήμα 2.5 Βασική αρχή δίπλωσης multirate φίλτρου

Είναι προφανές ότι η παραπάνω τεχνική μπορεί να επεκταθεί και σε περισσότερα σημεία ή/και διαφορετικές τοπολογίες. Ας θεωρήσουμε ξανά το γενικό συμμετρικό FIR φίλτρο k σημείων:



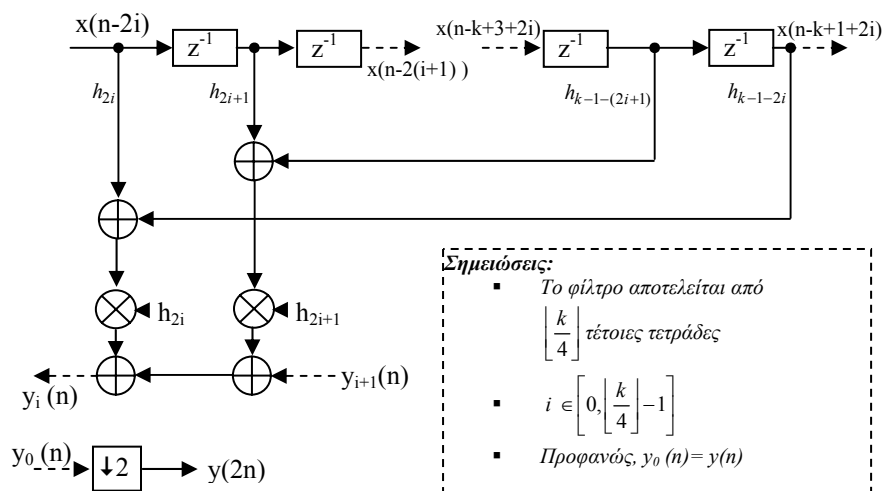
Σχήμα 2.2 Εκμετάλλευση της συμμετρικότητας των συντελεστών

Το παραπάνω φίλτρο μπορεί να θεωρηθεί ότι αποτελείται από δύο συνιστώσες:

- Ένα άθροισμα από $\left\lfloor \frac{k}{4} \right\rfloor$ τετράδες σημείων (το «ομαλό» μέρος), ανά δύο διαδοχικά και ανά δύο συμμετρικά, των οποίων οι (δύο) πολλαπλασιαστές και αθροιστές συμμετρίας μπορούν να «δίπλωθούν» σε έναν σύμφωνα με το σχ. 2.5.
- Ένα «υπόλοιπο» ($k \bmod 4$) από μηδέν ως τρία σημεία (τουλάχιστο τα δύο να είναι συμμετρικά και όλα διαδοχικά) και τα οποία, πάλι παρόμοια με το σχ.

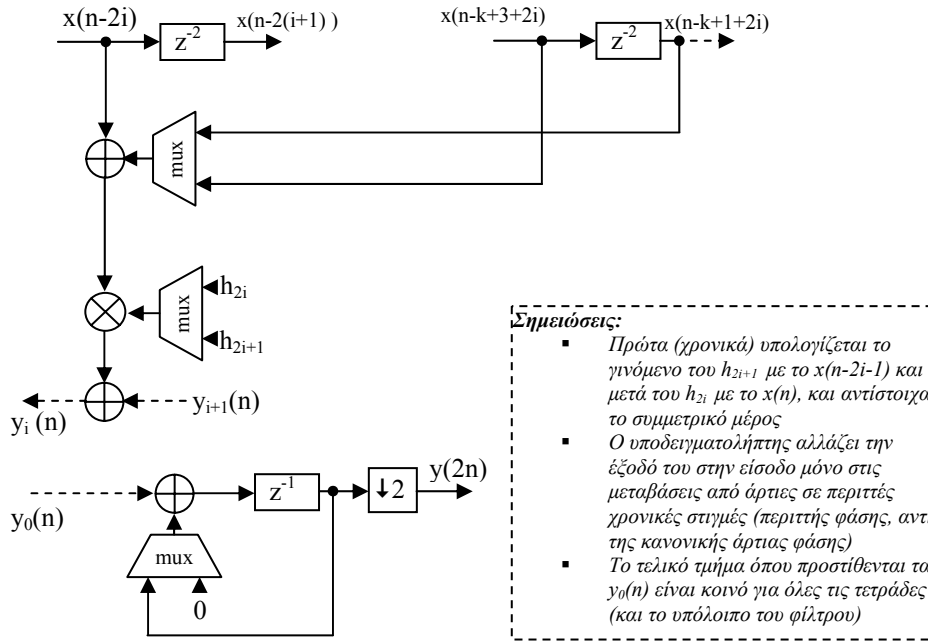
2.5 «διπλώνονται» γύρω από έναν πολλαπλασιαστή και έναν αθροιστή. Η ανάλυση των επιδόσεων του συνολικού «διπλωμένου» φίλτρου σε κάθε περίπτωση απαιτεί και την ανάλυση αυτού του υπολοίπου.

Κανονικό Μέρος Παρακάτω παραθέτουμε το σχήμα της i -οστής τετράδας (i ακέραιος μεταξύ 0 και $\left\lfloor \frac{k}{4} \right\rfloor - 1$) που αποτελεί το ομαλό μέρος. Είναι φανερό ότι το παρακάτω σχήμα αποτελεί απλώς ένα κομμάτι του σχ. 2.2, και επιλέχθηκε έτσι ώστε οι συντελεστές να είναι ανά δύο διαδοχικοί και ανά δύο συμμετρικοί. Οι επιδόσεις της αρχιτεκτονικής αυτής έχουν επισημανθεί στην (2.7)



Σχήμα 2.6a Η i -οστή τετράδα του ομαλού μέρους

Με βάση, λοιπόν, τη δίπλωση στο σχήμα 2.5, καταλήγω στην παρακάτω folded μορφή για την i -οστή τετράδα του φίλτρου του παραπάνω σχήματος:



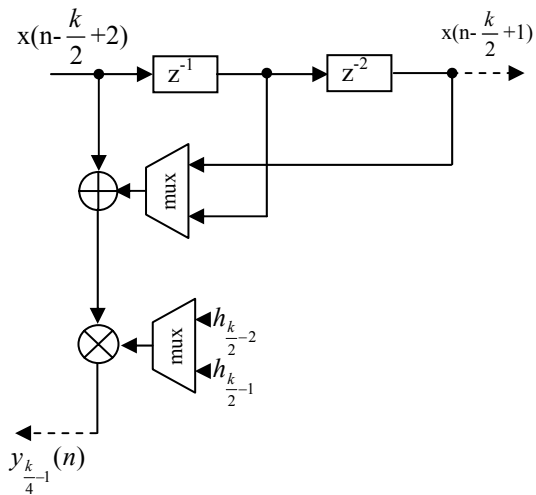
Σχήμα 2.6b Folding της i-οστής τετράδας του ομαλού μέρους

Κάθε τετράδα συνεισφέρει στο συνολικό κύκλωμα (2.14):

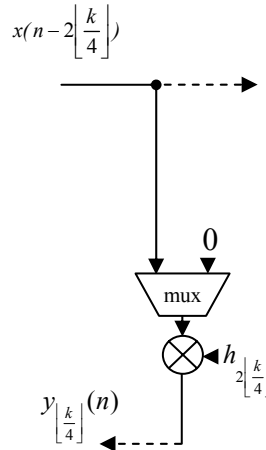
- Υλικό: 2 πολυπλέκτες, 1 πολλαπλασιαστή, 2 αθροιστές, 4 καταχωρητές
- Προστίθεται ένα T_A στο κρίσιμο μονοπάτι (δεν μπορούμε να το ποσοτικοποιήσουμε αν δεν αναλύσουμε το τμήμα υπολοίπου απ' όπου ξεκινάει)
- Αναμενόμενη κατανάλωση ισχύος: $4P_R+2P_A+P_M+2P_{mux}$, όπου P_{mux} η ισχύς που καταναλώνει ένας πολυπλέκτης σε πλήρη ρυθμό λειτουργίας

Μέρος Υπολοίπου Για να μπορούμε να έχουμε μια πλήρη ποσοτική περιγραφή της folded αρχιτεκτονικής, θα πρέπει να μελετήσουμε και το «υπόλοιπο» του κυκλώματος, δηλαδή τα $k-4 \left\lfloor \frac{k}{4} \right\rfloor$ σημεία που δεν μπορούν να ομαδοποιηθούν όπως στο ομαλό μέρος.

- **(k modulo 4)=0** Στην περίπτωση αυτή, μέρος υπολοίπου δεν υφίσταται ουσιαστικά, για δε $i = \left\lfloor \frac{k}{4} \right\rfloor - 1 = \frac{k}{4} - 1$, στο σχ. 2.26b έχω $x(n-2i) = x(n - \frac{k}{2} + 2)$, $x(n-2(i+1)) = x(n - \frac{k}{2})$, $x(n-k+3+2i) = x(n - \frac{k}{2} + 1)$, $x(n-k+1+2i) = x(n - \frac{k}{2} - 1)$, οπότε, το μέρος υπολοίπου (υπενθυμίζουμε ότι εδώ το ομαλό μέρος έχει μόνο $\left\lfloor \frac{k}{4} \right\rfloor - 1$ τετράδες) έχει τη μορφή που φαίνεται στο σχ. (2.7a):



Σχήμα 2.7a Υπόλοιπο για (k modulo 4)=0



Σχήμα 2.7b Υπόλοιπο για (k modulo 4)=1

Συνολικά, και από την (2.14), ένα folded φίλτρο με (k modulo 4)=0 έχει τα εξής χαρακτηριστικά (2.15):

-Υλικό: $\frac{k}{4} - 1 + 1 = \frac{k}{4}$ πολλαπλασιαστές, $2 * \left(\frac{k}{4} - 1\right) + 2 + 1 = \frac{k}{2} + 1$ πολυπλέκτες,

$2 * \left(\frac{k}{4} - 1\right) + 1 + 1 = \frac{k}{2}$ αθροιστές, $4 * \left(\frac{k}{4} - 1\right) + 3 + 1 = k$ καταχωρητές και έναν υποδειγματολήπτη

-Κρίσιμο μονοπάτι: $T_{mux} + T_A + T_M + \left(\frac{k}{4} - 1\right)T_A + T_A = T_{mux} + T_M + \left(\frac{k}{4} + 1\right)T_A$, όπου

T_{mux} η καθυστέρηση που εισάγει ένας πολυπλέκτης

-Κατανάλωση ισχύος: $\frac{k}{4} P_M + \left(\frac{k}{2} + 1\right) P_{mux} + \frac{k}{2} P_A + k P_R$

- **(k modulo 4)=1** Στην περίπτωση αυτή, το τμήμα υπολοίπου αποτελείται από ένα μόνο στοιχείο. Στο σχ. 2.6b, έχω για $i = \left\lfloor \frac{k}{4} \right\rfloor - 1$ ότι $x(n - 2(i + 1)) = x(n - 2 \left\lfloor \frac{k}{4} \right\rfloor)$ και

$$x(n - k + 3 + 2i) = x(n - 4 \left\lfloor \frac{k}{4} \right\rfloor - 1 + 3 + 2 \left\lfloor \frac{k}{4} \right\rfloor - 2) = x(n - 2 \left\lfloor \frac{k}{4} \right\rfloor),$$

συνεπώς, το μέρος

υπολοίπου έχει τη μορφή του σχ. (2.7b)

Συνολικά, και από την (2.14), ένα folded φίλτρο με (k modulo 4)=1 έχει τα εξής χαρακτηριστικά (2.16):

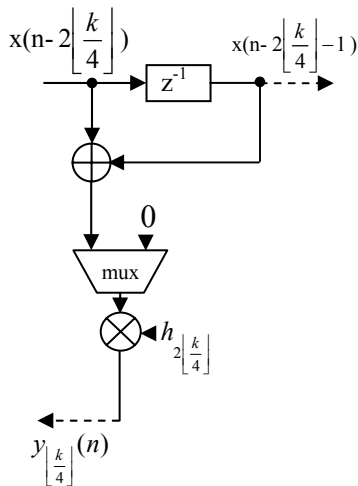
-Υλικό: $\left\lfloor \frac{k}{4} \right\rfloor + 1$ πολλαπλασιαστές, $2 * \left\lfloor \frac{k}{4} \right\rfloor + 1 + 1 = 2 * \left\lfloor \frac{k}{4} \right\rfloor + 2$ πολυπλέκτες,

$2 * \left\lfloor \frac{k}{4} \right\rfloor + 1$ αθροιστές, $4 * \left\lfloor \frac{k}{4} \right\rfloor + 1 = k$ καταχωρητές και έναν υποδειγματολήπτη

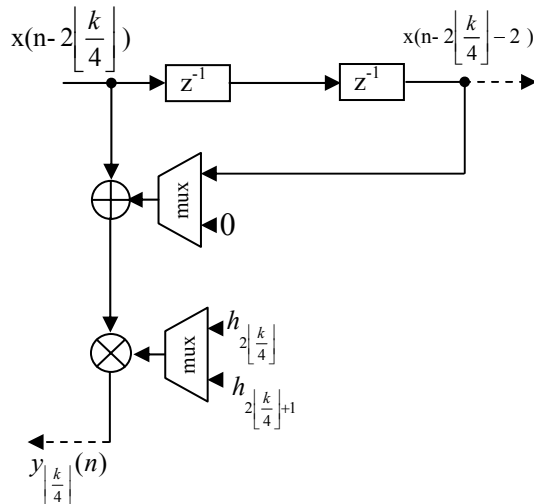
-Κρίσιμο μονοπάτι: $T_{mux} + T_A + T_M + \left\lfloor \frac{k}{4} \right\rfloor T_A + T_A = T_{mux} + T_M + \left(\left\lfloor \frac{k}{4} \right\rfloor + 2\right) T_A$

-Κατανάλωση ισχύος: $\left(\left\lfloor \frac{k}{4} \right\rfloor + 1\right) P_M + \left(2 \left\lfloor \frac{k}{4} \right\rfloor + 2\right) P_{mux} + \left(2 \left\lfloor \frac{k}{4} \right\rfloor + 1\right) P_A + k P_R$

- **(k modulo 4)=2** Στην περίπτωση αυτή, το τμήμα υπολοίπου αποτελείται από δύο στοιχεία. Στο σχ. 2.6b, έχω για $i = \lfloor \frac{k}{4} \rfloor - 1$ ότι $x(n-2(i+1)) = x(n-2\lfloor \frac{k}{4} \rfloor)$ και $x(n-k+3+2i) = x(n-4\lfloor \frac{k}{4} \rfloor - 2 + 3 + 2\lfloor \frac{k}{4} \rfloor - 2) = x(n-2\lfloor \frac{k}{4} \rfloor - 1)$, συνεπώς, το μέρος υπολοίπου έχει τη μορφή του σχ. 2.7c



Σχήμα 2.7c Υπόλοιπο για (k modulo 4)=2



Σχήμα 2.7d Υπόλοιπο για (k modulo 4)=3

Συνολικά, και από την (2.14), ένα folded φίλτρο με $(k \text{ modulo } 4)=2$ έχει τα εξής χαρακτηριστικά (2.17):

-Υλικό: $\lfloor \frac{k}{4} \rfloor + 1$ πολλαπλασιαστές, $2 * \lfloor \frac{k}{4} \rfloor + 1 + 1 = 2 * \lfloor \frac{k}{4} \rfloor + 2$ πολυπλέκτες,

$2 * \lfloor \frac{k}{4} \rfloor + 2$ αθροιστές, $4 * \lfloor \frac{k}{4} \rfloor + 2 = k$ καταχωρητές και έναν υποδειγματολήπτη

-Κρίσιμο μονοπάτι: $T_{mux} + T_A + T_M + \lfloor \frac{k}{4} \rfloor T_A + T_A = T_{mux} + T_M + \left(\lfloor \frac{k}{4} \rfloor + 2 \right) T_A$

-Κατανάλωση ισχύος: $\left(\lfloor \frac{k}{4} \rfloor + 1 \right) P_M + \left(2 \lfloor \frac{k}{4} \rfloor + 2 \right) P_{mux} + \left(2 \lfloor \frac{k}{4} \rfloor + 2 \right) P_A + k P_R$

- **(k modulo 4)=3** Στην περίπτωση αυτή, το τμήμα υπολοίπου αποτελείται από τρία στοιχεία. Στο σχ. 2.6b, έχω για $i = \lfloor \frac{k}{4} \rfloor - 1$ ότι $x(n-2(i+1)) = x(n-2\lfloor \frac{k}{4} \rfloor)$ και $x(n-k+3+2i) = x(n-4\lfloor \frac{k}{4} \rfloor - 3 + 3 + 2\lfloor \frac{k}{4} \rfloor - 2) = x(n-2\lfloor \frac{k}{4} \rfloor - 2)$, συνεπώς, το μέρος υπολοίπου έχει τη μορφή του σχ. 2.7d

Συνολικά, και από την (2.14), ένα folded φίλτρο με $(k \text{ modulo } 4)=3$ έχει τα εξής χαρακτηριστικά (2.18):

-Υλικό: $\lfloor \frac{k}{4} \rfloor + 1$ πολλαπλασιαστές, $2 * \lfloor \frac{k}{4} \rfloor + 2 + 1 = 2 * \lfloor \frac{k}{4} \rfloor + 3$ πολυπλέκτες,

$2 * \lfloor \frac{k}{4} \rfloor + 2$ αθροιστές, $4 * \lfloor \frac{k}{4} \rfloor + 3 = k$ καταχωρητές και έναν υποδειγματολήπτη

-Κρίσιμο μονοπάτι: $T_{mix} + T_A + T_M + \left\lfloor \frac{k}{4} \right\rfloor T_A + T_A = T_{mix} + T_M + \left(\left\lfloor \frac{k}{4} \right\rfloor + 2 \right) T_A$

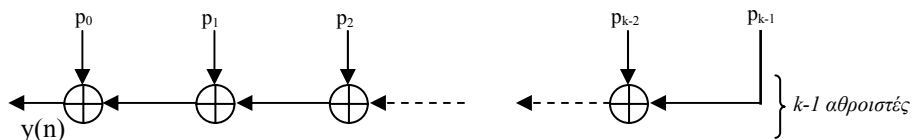
-Κατανάλωση ισχύος: $\left(\left\lfloor \frac{k}{4} \right\rfloor + 1 \right) P_M + \left(2 \left\lfloor \frac{k}{4} \right\rfloor + 3 \right) P_{mix} + \left(2 \left\lfloor \frac{k}{4} \right\rfloor + 2 \right) P_A + k P_R$

δ. Τεχνικές για περαιτέρω βελτίωση των επιδόσεων

Τέλος, πριν προχωρήσουμε στην διερεύνηση των φίλτρων σε transpose μορφή, θα δούμε δύο τεχνικές που μπορούν να εφαρμοστούν στα φίλτρα σε κανονική μορφή και οι οποίες οδηγούν σε βελτίωση των χαρακτηριστικών του κυκλώματος

Οργάνωση της αλυσίδας των αθροιστών σε δένδρο

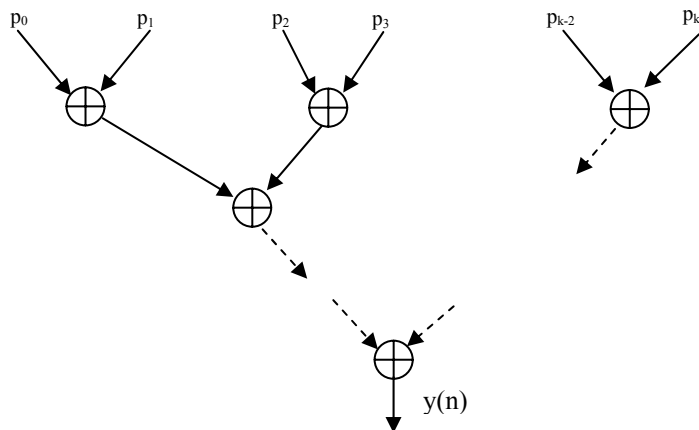
Ένα σημαντικό μειονέκτημα των φίλτρων σε direct μορφή είναι το μεγάλο μήκος του κρίσιμου μονοπατιού, το οποίο οφείλεται στους αθροιστές των γινομένων και το οποίο αυξάνεται γραμμικά με την αύξηση του αριθμού των σημείων του φίλτρου. Έστω, λοιπόν, όπως φαίνεται στο σχ. 2.8a, μια τυπική αλυσίδα άθροισης γινομένων ενός FIR φίλτρου k σημείων:



Σχήμα 2.8a Τυπική αλυσίδα άθροισης γινομένων

Είναι φανερό ότι η αλυσίδα αυτή των αθροιστών επιβαρύνει το κρίσιμο μονοπάτι με $(k-1)T_A$. Μια γνωστή μεθοδολογία μείωσης αυτής της επιβάρυνσης είναι να αθροίσουμε τα γινόμενα (υπενθυμίζουμε εδώ ότι λόγω της ιδιότητας προσεταιρισμού της πρόσθεσης, μπορούμε να τα αθροίσουμε με όποια σειρά θέλουμε, πράγμα που το εκμεταλλευτήκαμε πολύ στην εύρεση αποδοτικών αρχιτεκτονικών πριν) σε μορφή δένδρου.

Η οργάνωση των αθροιστών σε (δυναμικό) δένδρο έγκειται στο να αθροίσουμε τα «γειτονικά» γινόμενα μεταξύ τους ανά δύο, και στη συνέχεια, να προστεθούν κατά παρόμοιο τρόπο μεταξύ τους τα αποτελέσματα των προσθέσεων, όπως φαίνεται στο επόμενο σχήμα:



Σχήμα 2.8b Οργάνωση των αθροιστών σε δένδρο

Θα μελετήσουμε τώρα το μέγεθος του κρίσιμου μονοπατιού που προκύπτει και τον αριθμό των αθροιστών που απαιτούνται στο παραπάνω σχήμα:

Όσον αφορά το κρίσιμο μονοπάτι, ο αριθμός των αποτελεσμάτων της πρώτης σειράς προσθέσεων είναι $\left\lceil \frac{k}{2} \right\rceil$ (επειδή πιθανώς το k να είναι περιττό, οπότε το τελευταίο γινόμενο διοχετεύεται κατ'ευθείαν στην επόμενη σειρά). Αυτά, όταν προστεθούν μεταξύ τους, θα δώσουν $\left\lceil \frac{\left\lceil \frac{k}{2} \right\rceil}{2} \right\rceil = \left\lceil \frac{k}{4} \right\rceil$ αποτελέσματα, αυτά στη συνέχεια

δίνουν $\left\lceil \frac{k}{8} \right\rceil$ αθροίσματα. Οι προσθέσεις σταματάνε μετά από m τέτοια βήματα, όταν ισχύει για πρώτη φορά $\left\lceil \frac{k}{2^m} \right\rceil = 1$. Άρα, $m = \lceil \log_2 k \rceil$. Συνεπώς, το μήκος του κρίσιμου μονοπατιού μειώνεται από $k-1$ σε $\lceil \log_2 k \rceil$ (2.19)

Επίσης, αριθμός των αθροιστών δίδεται από το γεγονός ότι όλα τα δυαδικά δένδρα που έχουν αριθμό φύλλων k έχουν $k-1$ κόμβους, συνεπώς, δεν έχω κάποια επιβάρυνση σε υλικό.

Με τη χρήση της (2.19), έχω για το κρίσιμο μονοπάτι σε όλες τις αρχιτεκτονικές που προηγήθηκαν:

- Στην αρχιτεκτονική με απλή εκμετάλλευση της συμμετρίας (σχ. 2.2) έχω $\left\lceil \frac{k}{2} \right\rceil$ γινόμενα, άρα, το κρίσιμο μονοπάτι γίνεται $T_A + T_M + \left\lceil \log_2 \left\lceil \frac{k}{2} \right\rceil \right\rceil T_A = T_M + \left(\left\lceil \log_2 \left\lceil \frac{k}{2} \right\rceil \right\rceil + 1 \right) T_A$ (2.20a), από $T_M + \left\lceil \frac{k}{2} \right\rceil T_A$ που ήταν αρχικά
- Στην αρχιτεκτονική με εκμετάλλευση της συμμετρίας και polyphase decomposition, έχω

- Για k περιττό (σχ. 2.4a) έχω δύο σύνολα $\left\lceil \frac{k+1}{4} \right\rceil$ και $\left\lceil \frac{k-1}{4} \right\rceil$ γινομένων αντίστοιχα, οπότε το κρίσιμο μονοπάτι δίδεται από τη σχέση: $T_A + T_M + \left\lceil \log_2 \left\lceil \frac{k+1}{4} \right\rceil \right\rceil T_A + T_A = T_M + \left(\left\lceil \log_2 \left\lceil \frac{k+1}{4} \right\rceil \right\rceil + 2 \right) T_A$ (2.20b), από $T_M + \left(\left\lceil \frac{k-1}{4} \right\rceil + 1 \right) T_A$ που ήταν αρχικά

- Για k άρτιο (σχ. 2.4b) έχω δύο σύνολα $\left\lceil \frac{k}{4} \right\rceil$ και $\left\lfloor \frac{k}{4} \right\rfloor$ γινομένων αντίστοιχα, οπότε το κρίσιμο μονοπάτι δίδεται από τη σχέση:

$$T_A + T_M + \left\lceil \log_2 \left\lceil \frac{k}{4} \right\rceil \right\rceil T_A + T_A = T_M + \left(\left\lceil \log_2 \left\lceil \frac{k}{4} \right\rceil \right\rceil + 2 \right) T_A \quad (2.20c)$$

από $T_M + \left(\left\lceil \frac{k}{4} \right\rceil + 1 \right) T_A$ που ήταν αρχικά

- Στην αρχιτεκτονική με εκμετάλλευση της συμμετρίας και folding (σχ. 2.6 και 2.7) έχω ένα σύνολο $\left\lceil \frac{k}{4} \right\rceil$ γινομένων προς πρόσθεση, οπότε, το κρίσιμο μονοπάτι γίνεται

$$T_{max} + T_A + T_M + \left\lceil \log_2 \left\lceil \frac{k}{4} \right\rceil \right\rceil T_A + T_A = T_{max} + T_M + \left(\left\lceil \log_2 \left\lceil \frac{k}{4} \right\rceil \right\rceil + 2 \right) T_A \quad (2.20d)$$

από $T_{max} + T_M + \left(\left\lceil \frac{k}{4} \right\rceil + 1 \right) T_A$ ή $T_{max} + T_M + \left(\left\lfloor \frac{k}{4} \right\rfloor + 2 \right) T_A$ που ήταν πριν, για (k modulo 4) ίσο με {0,1} ή {2,3} αντίστοιχα

Από τα παραπάνω γίνεται φανερό ότι η οργάνωση των αθροιστών σε δένδρο μπορεί να μην μειώνει πάντοτε το κρίσιμο μονοπάτι σε σχέση με την κλασσική «αλυσίδα» αθροιστών, ειδικά για μικρά k. Η αξιοποίηση, συνεπώς, αυτής της μεθόδου ή όχι έγκειται στην εφαρμογή

Χρήση κοινής αλυσίδας καταχωρητών

Κατά τα γνωστά, ο μετασχηματισμός wavelet ενός σήματος γίνεται από ένα ζευγάρι φίλτρων, ένα βαθυπερατό και ένα υψυπερατό, το κάθε ένα από τα οποία έχει – σε direct μορφή- τη γενική αρχιτεκτονική που φαίνεται στο σχ. 2.1. Αν το ένα φίλτρο είναι s και το άλλο t σημείων, υλοποιώντας το καθένα ξεχωριστά θα χρειαστούμε ο(s+t) καταχωρητές, όπως δείξαμε πριν. Ωστόσο, και τα δύο φίλτρα επεξεργάζονται το ίδιο σήμα, συνεπώς, θα μπορούσαμε να χρησιμοποιήσουμε μόνο μια σειρά καταχωρητών (προφανώς, του μεγαλύτερου φίλτρου) και για τα δύο φίλτρα (μιας και η αλυσίδα των καταχωρητών απλά παρέχει μετατοπισμένες στο χρόνο εκδοχές του x και συνεπώς μπορεί να συμμετάσχει σε απεριόριστο αριθμό φίλτρων ταυτόχρονα) και να μειώσουμε τον αριθμό των καταχωρητών σε ο(max (s,t)). Πιο συγκεκριμένα, έστω ότι θέλουμε να υλοποιήσουμε δύο συμμετρικά FIR φίλτρα με υποδειγματοληψία στην έξοδο, μήκους s και t αντίστοιχα. Τότε:

- Στην αρχιτεκτονική με απλή εκμετάλλευση της συμμετρίας (σχ. 2.2) χρειάζονται k-1 καταχωρητές για την αλυσίδα, οπότε, κανονικά θέλω s+t-2. Αν όμως, χρησιμοποιήσω κοινή αλυσίδα ο αριθμός μειώνεται σε max (s,t)-1 (2.21a)
- Στην αρχιτεκτονική με εκμετάλλευση της συμμετρίας και polyphase decomposition (σχ. 2.4a/b) χρειάζονται k-1 καταχωρητές για την αλυσίδα, οπότε, πάλι, μπορώ να μειώσω τον αριθμό τους από s+t-2 σε max (s,t)-1 (2.21b). Εδώ χρειάζεται η επισήμανση ότι με αυτόν τον τρόπο, το τελικό σχήμα της αρχιτεκτονικής του σχ. 2.4b θα γίνει υπερβολικά περίπλοκο!
- Στην αρχιτεκτονική με εκμετάλλευση της συμμετρίας και folding (σχ. 2.6/2.7) χρειάζομαι σε κάθε (ξεχωριστό) φίλτρο k-1 καταχωρητές για την αλυσίδα και

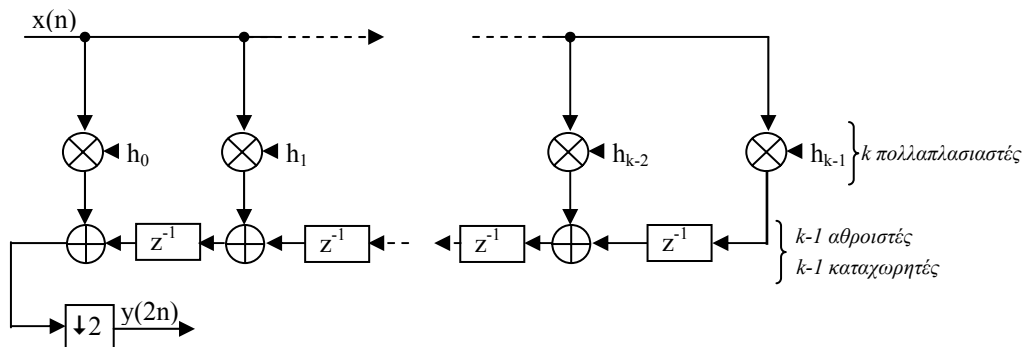
έναν για την πρόσθεση των διαφορετικών φάσεων, άρα, με χρήση κοινής αλυσίδας ο αριθμός τους μειώνεται από $s+t$ σε $\max(s,t)+1$ (2.21c)

Είδαμε, λοιπόν, ως τώρα τους τρόπους εξαγωγής αποδοτικών αρχιτεκτονικών από συμμετρικά FIR φίλτρα με υποδειγματοληψία ανά δύο στην έξοδο, όταν αυτά είναι σε direct μορφή. Στη συνέχεια, θα μελετήσουμε τα ίδια φίλτρα ξεκινώντας από την transpose μορφή τους η οποία, αν και δεν είναι τόσο ευέλικτη όπως θα αποδειχθεί πολύ σύντομα, έχει το πλεονέκτημα του πολύ μικρότερου κρίσιμου μονοπατιού.

3. Transpose μορφή

α. Γενικά

Η transpose μορφή ενός γενικού FIR φίλτρου k σημείων με υποδειγματοληψία ανά δύο δείγματα στην έξοδο προκύπτει με διαδοχική εφαρμογή του αξιώματος (2.4d) στο σχήμα 2.1 και έχει τη μορφή που φαίνεται στο παρακάτω σχήμα:



Σχήμα 2.9 Transpose μορφή γενικού φίλτρου

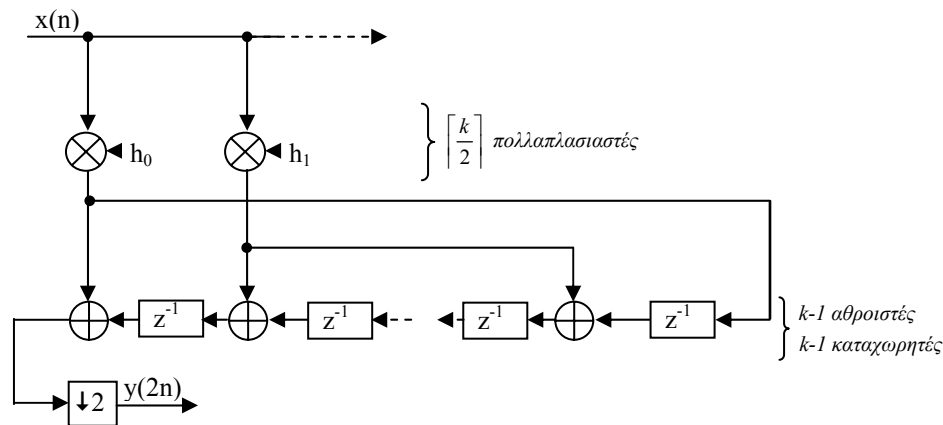
Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.22)

- Πλήθος στοιχείων: $k-1$ καταχωρητές, $k-1$ αθροιστές, k πολλαπλασιαστές, 1 υποδειγματολήπτης.
- Το κρίσιμο μονοπάτι ισούται με T_M+T_A
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται ισούται με $kP_M + (k-1)(P_A + P_R)$

Βλέπουμε, λοιπόν, παραπάνω ότι η transpose μορφή του κυκλώματος, ενώ έχει ίδιες απαιτήσεις σε υλικό και ισχύ απ'ότι η κανονική μορφή, εξασφαλίζει κρίσιμο μονοπάτι πολύ μικρότερο και, το σημαντικότερο, σταθερό και ασύνδετο με το πλήθος των σημείων του φίλτρου. Αυτό είναι ένα σημαντικότατο πλεονέκτημα της transpose μορφής η οποία, παρά το ότι δεν είναι τόσο ευέλικτη όσο η κανονική, βρίσκει σημαντικές εφαρμογές στα αριθμητικά κυκλώματα.

Φυσικά και εδώ θα προσπαθήσουμε να εκμεταλλευτούμε τη συμμετρία των συντελεστών του φίλτρου, όπως και την υποδειγματοληψία στην έξοδο. Για να κάνουμε αντιπαραβολή με την κανονική μορφή, στο σχ. 2.9 παρατηρούμε ότι τα γινόμενα πρέπει να προστεθούν με συγκεκριμένη σειρά και επιπλέον, είναι αδύνατο να χρησιμοποιήσουμε το τέχνασμα της κοινής γραμμής καταχωρητών για ζεύγη φίλτρων, όπως πριν (η οργάνωση των αθροιστών σε δένδρο είναι όχι μόνο αδύνατη, αλλά χωρίς νόημα)

Παρατηρούμε στο σχ. 2.9 ότι σε όλους τους πολλαπλασιαστές διοχετεύεται το ίδιο σήμα, συνεπώς, αφού στην περίπτωσή μας έχουμε συμμετρία των συντελεστών, δηλαδή ισχύει η (2.3) μπορούμε να συγχωνεύσουμε άμεσα τους $2 \left\lfloor \frac{k}{2} \right\rfloor$ από τους πολλαπλασιαστές σε $\left\lfloor \frac{k}{2} \right\rfloor$, μειώνοντας τον τελικό τους αριθμό σε $\left\lfloor \frac{k}{2} \right\rfloor$, όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 2.10 Εκμετάλλευση συμμετρίας στην transpose μορφή

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.23)

- Πλήθος στοιχείων: $k-1$ καταχωρητές, $k-1$ αθροιστές, $\left\lfloor \frac{k}{2} \right\rfloor$ πολλαπλασιαστές, 1 υποδειγματολήπτης.
- Το κρίσιμο μονοπάτι ισούται με $T_M + T_A$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται ισούται με $\left\lfloor \frac{k}{2} \right\rfloor P_M + (k-1)(P_A + P_R)$

Σε αυτό το σημείο θα πρέπει να επισημάνουμε μια σημαντικότερη διαφορά μεταξύ της εκμετάλλευσης της συμμετρίας στην direct και την εκμετάλλευση της συμμετρίας στην transpose μορφή. **Ενώ στην κανονική μορφή η εκμετάλλευση της συμμετρίας έγκειται στην πρόσθεση-ομαδοποίηση των σημάτων που πολλαπλασιάζονται με τον ίδιο συντελεστή, στην transpose μορφή αυτό ανάγεται στην υπόθεση ότι το συγκεκριμένο γινόμενο θα χρειαστεί και σε μια δεύτερη χρονική στιγμή, αργότερα.**

Είναι όμως αυτή η υπόθεση πάντα βάσιμη σε φίλτρα των οποίων η έξοδος υποδειγματοληπτείται ανά δύο σημεία; Έστω ένα φίλτρο k σημείων την χρονική στιγμή n , και ας υποθέσουμε ότι τη χρονική στιγμή αυτή το γινόμενο του $x(n)$ με το h_i , $i \in \left[0, \left\lfloor \frac{k}{2} \right\rfloor \right]$ θα χρησιμοποιηθεί αργότερα σε κάποιο δείγμα εξόδου (αυτή η υπόθεση δεν είναι αυθαίρετη: όπως δείξαμε πιο πάνω, αυτό ισχύει ότι και το n και το i είναι και τα δύο άρτια ή και τα δύο περιττά, συνεπώς, μπορούμε πάντα να διαλέξουμε μια κατάλληλη χρονική στιγμή n)

Από τη στιγμή που το φίλτρο είναι συμμετρικό, χρησιμοποιώντας το μηχανισμό του σχ. 2.10 υποθέτουμε ότι και το γινόμενο του h_{k-1-i} με το $x(n)$ θα χρησιμοποιηθεί σε κάποιο δείγμα εξόδου, και, μάλιστα, λόγω των $(k-1-i-i=)$ $k-1-2i$ καταχωρητών μεταξύ των δύο γινομένων, στο δείγμα εξόδου που θα εμφανιστεί $k-1-2i$ κύκλους του ρολογιού αργότερα από αυτό που θα περιέχει το h_i .

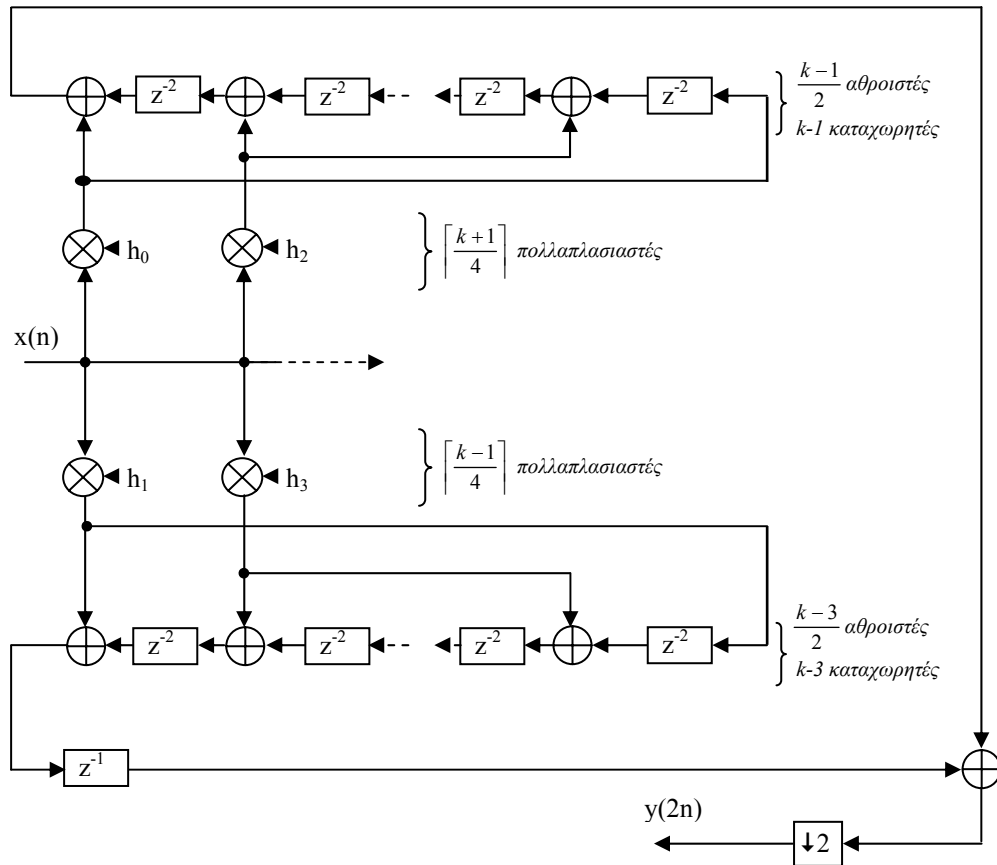
Είναι προφανές ότι αν το $k-1-2i$ είναι περιττός, δηλαδή, αν το k είναι άρτιος, το δείγμα εξόδου που περιέχει το γινόμενο του h_{k-1-i} με το $x(n)$ θα απορριφθεί από τον δειγματολήπτη. Συνεπώς, η εκμετάλλευση της συμμετρίας δεν έχει κάποιο ρόλο βελτιστοποίησης του κυκλώματος, αφού «συγκρούεται» με την υποδειγματοληψία. **Άρα, λοιπόν, τα συμμετρικά FIR φίλτρα με υποδειγματοληψία στην έξοδο και άρτιο αριθμό σημείων δεν έχουν αποδοτική υλοποίηση στην transpose μορφή.**

Αυτός είναι ένας από τους λόγους που εν γένει τα wavelets με άρτιο αριθμό σημείων στα φίλτρα τους δεν έχουν τόσο μεγάλη διάδοση σε σχέση με αυτά που έχουν περιττό αριθμό, όπως το CDF 9/7, στο οποίο θα επικεντρωθεί και στη συνέχεια αυτή η διπλωματική εργασία. Ως ακόλουθο αυτού, εμείς στη συνέχεια **θα ασχοληθούμε αποκλειστικά με φίλτρα περιττού αριθμού σημείων**, δηλαδή, με k της μορφής $k=2m+1$

Δείξαμε, λοιπόν, πάνω ότι η αρχιτεκτονική του σχ. 2.10 κάνει αποδοτική εκμετάλλευση της συμμετρίας όταν το φίλτρο είναι περιττό. Ωστόσο, η χρησιμοποίηση του υλικού παραμένει 50%, αφού λόγω της υποδειγματοληψίας στην έξοδο, τα μισά δείγματα εξόδου απορρίπτονται. Έτσι, και εδώ, έχουμε δύο τρόπους να εκμεταλλευτούμε και την υποδειγματοληψία: είτε να διατηρήσουμε τον ίδιο αριθμό στοιχείων και να μειώσουμε το ρυθμό λειτουργίας τους στο μισό, είτε να διατηρήσουμε τον ίδιο ρυθμό λειτουργίας και να οδηγηθούμε σε folded αρχιτεκτονικές, μειώνοντας τον αριθμό των στοιχείων

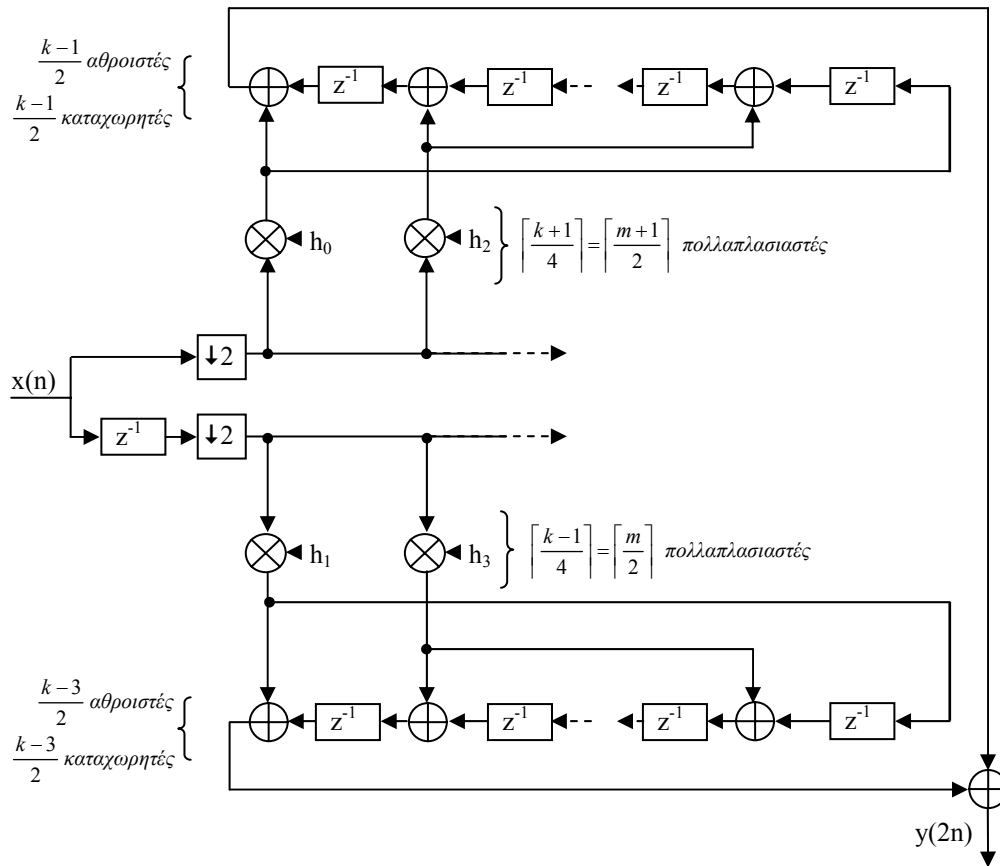
β. Μείωση του ρυθμού λειτουργίας των στοιχείων του φίλτρου

Όπως αναφέραμε και πιο πριν, το ζητούμενο σε αυτήν την περίπτωση είναι να μετακινήσουμε τον υποδειγματολήπτη από το τέλος στην αρχή του γράφου σηματορροής. Τα αξιώματα (2.4b) και (2.4c) δίνουν τις μόνες γνωστές μεθόδους με τις οποίες αυτό μπορεί να γίνει, οπότε, κατ' αντιστοιχία αυτών που έγιναν στην direct μορφή, θα χωρίσουμε και εδώ τα σημεία του φίλτρου σε άρτιου και περιττού βαθμού, όπως φαίνεται παρακάτω (υπενθυμίζουμε ότι το φίλτρο έχει $k=2m+1$ σημεία):



Σχήμα 2.11α Διαχωρισμός αρτίων-περιττών

οπότε, εφαρμόζοντας στον περιττό κλάδο του παραπάνω το αξίωμα (2.4d) έτσι ώστε ο μονός καταχωρητής να μετακινηθεί στην αρχή του κλάδου, και τις (2.4b), (2.4c) για να μετακινήσω τον υποδειγματολήπτη, έχω την τελική αρχιτεκτονική μείωσης του ρυθμού λειτουργίας των στοιχείων/polyphase decomposition όπως φαίνεται στο σχ. 2.11b:



Σχήμα 2.11b Συμμετρία και polyphase decomposition σε transpose μορφή

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.24)

- Πλήθος στοιχείων: $\frac{k-3}{2} + \frac{k-1}{2} + 1 = k-1$ καταχωρητές, $\frac{k-3}{2} + \frac{k-1}{2} + 1 = k-1$ αθροιστές, 2

υποδειματολήπτες. Ο αριθμός των πολλαπλασιαστών ισούται με $\left\lfloor \frac{m+1}{2} \right\rfloor + \left\lfloor \frac{m}{2} \right\rfloor$. Από τα

$(m+1)$ και (m) τουλάχιστον το ένα είναι άρτιο, οπότε το προηγούμενο γράφεται $\left(\frac{m+1}{2}\right) + \left(\frac{m+1}{2}\right) = m+1 = \left\lfloor \frac{k}{2} \right\rfloor$ ή $\left(\frac{m+2}{2}\right) + \left(\frac{m}{2}\right) = m+1 = \left\lfloor \frac{k}{2} \right\rfloor$, οπότε, σε κάθε

περίπτωση, ο αριθμός των πολλαπλασιαστών ισούται με $\left\lfloor \frac{k}{2} \right\rfloor$

- Το κρίσιμο μονοπάτι ισούται με $T_M + 2T_A$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται ισούται με $\left\lfloor \frac{k}{2} \right\rfloor \frac{P_M}{2} + (k-1) \frac{P_A}{2} + (k-2) \frac{P_R}{2} + P_R = \frac{1}{2} \left(\left\lfloor \frac{k}{2} \right\rfloor P_M + (k-1) P_A + k P_R \right)$

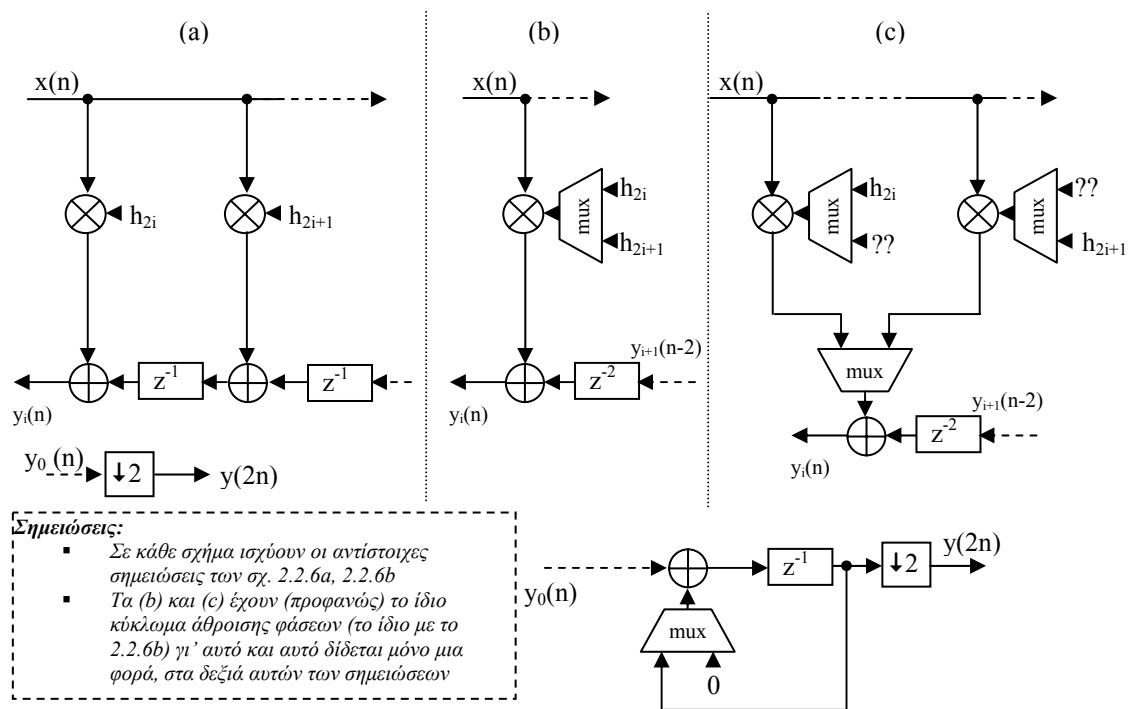
γ. Folding

Όπως και στην περίπτωση των φίλτρων σε direct μορφή, και εδώ το “folding” έγκειται ουσιαστικά στη χρονική δίπλωση του άρτιου με το περιττό φίλτρο. Ωστόσο, για λόγους ευκολίας θα ακολουθήσουμε και εδώ διαφορετικό δρόμο για την ανάδειξη της τελικής αρχιτεκτονικής και των επιδόσεών της.

Ας θεωρήσουμε, λοιπόν, το γενικό (όχι απαραίτητα συμμετρικό) FIR φίλτρο $k=2m+1$ σημείων με υποδειματοληψία στην έξοδο, στην transpose μορφή του

σχ. 2.9. Οποιαδήποτε προσπάθεια για «δίπλωση» των στοιχείων του θα ξεκίναγε από την αρχή του φίλτρου, συγχωνεύοντας ανά δύο τα γειτονικά σημεία. Το φίλτρο έχει $\left\lceil \frac{k}{2} \right\rceil$ τέτοια διαδοχικά ζεύγη, από τα οποία το τελευταίο θεωρείται ότι «διπλώνεται» με έναν μηδενικό συντελεστή (ή εναλλακτικά, μπορούμε να θεωρήσουμε ότι μεταξύ αυτού και του $x(n)$ υπάρχει ένας 2-σε-1 πολυπλέκτης, που τις περιττές χρονικές στιγμές επιλέγει μηδενική είσοδο: με αυτόν τον τρόπο θα μπορούσαμε να χρησιμοποιήσουμε έναν σταθερό πολλαπλασιαστή).

Τότε, το i -οστό τέτοιο ζευγάρι γινομένων/σημείων θα μπορούσε να διπλωθεί με έναν από τους δύο τρόπους το σχ. 2.12:



Σχήμα 2.12 Ισοδύναμες μορφές folding του i -οστού ζευγαριού

Στο παραπάνω σχήμα, η επιλογή 2.12c φαντάζει μάλλον άκομψη: ωστόσο έχει μεγάλη σημασία στα συμμετρικά φίλτρα όπως θα δούμε ευθύς αμέσως.

Στο αρχικό μας φίλτρο, λοιπόν, μετατρέπουμε από τη μορφή 2.12a στην 2.12b τα $\left\lceil \frac{k}{2} \right\rceil$ ζευγάρια, στα οποία τα σημεία του φίλτρου ομαδοποιούνται στη μορφή (h_{2i}, h_{2i+1}) ως εξής:

$$(h_0, h_1), (h_2, h_3), (h_4, h_5), \dots, (h_{m-4}, h_{2m-3}), (h_{2m-2}, h_{2m-1}), (h_{2m}, 0)$$

Βλέπουμε, λοιπόν, ότι αν το φίλτρο είναι συμμετρικό, δηλαδή ισχύει η (2.3), τότε οι μεταξύ τους ίσοι (λόγω συμμετρίας) όροι δεν περιέχονται στα ίδια ζευγάρια. Δηλαδή, ο h_{2i} βρίσκεται στο ίδιο ζευγάρι με τον h_{2i+1} , ενώ ο συμμετρικός του h_{2i} , h_{2m-2i} είναι στο ίδιο ζευγάρι με τον $h_{2m-2i+1}$ (ή το 0) και όχι με τον $h_{2m-2i-1}$ που είναι ο συμμετρικός του h_{2i+1} . Αυτό οδηγεί σε δύο λύσεις: ή διπλώνουμε όλα τα ζευγάρια

όπως στο 2.12b και ξεχγάμε τη συμμετρία (που δεν είναι επιτρεπτό, καθώς χάρη σε αυτή μπορούμε να μειώσουμε τον αριθμό των πολλαπλασιαστών σχεδόν στο μισό), ή διπλώνουμε τα πρώτα $m+1$ γινόμενα ανά δύο όπως στο 2.12b και στη συνέχεια χρησιμοποιούμε τους πολλαπλασιαστές τους για να διπλώσουμε τα υπόλοιπα γινόμενα όπως στο 2.12c (μειώνοντας, έτσι, τον αριθμό των πολλαπλασιαστών στο $\left\lceil \frac{k}{4} \right\rceil$).

Ωστόσο, δεν είμαστε αναγκασμένοι να επιλέξουμε τα πρώτα $m+1$ γινόμενα για δίπλωση όπως στο σχ. 2.12b: είμαστε εξ' ίσου ελεύθεροι να κάνουμε το ίδιο πράγμα με τα τελευταία $m+1$ γινόμενα (ο σκοπός είναι πάντοτε, προφανώς, να καλύπτουμε μέσω της συμμετρίας των συντελεστών, όλα τα σημεία του φίλτρου). Το ποια από τις δύο μεθοδολογίες είναι αποδοτικότερη (ή, αν είναι ισοδύναμες) θα αναλυθεί στη συνέχεια, ανάλογα με το αν το m είναι άρτιο ή περιττό.

Είναι προφανές ότι, αφού ο αριθμός των πολλαπλασιαστών είναι πάντοτε ο ίδιος, θέλουμε για οικονομία στο υλικό (λιγότεροι πολυπλέκτες) όσον το δυνατόν περισσότερα ζευγάρια να διπλώνονται όπως στο 2.12b.

m άρτιο

Αν το m είναι άρτιο, τότε οι ομαδοποιήσεις γινομένων (σημείων) σε ζευγάρια έχουν τη μορφή τη μορφή του παρακάτω πίνακα:

άρτιος	h_0	h_2	...	h_m	h_{m-2}	...	h_2	h_0
περιττός	h_1	h_3	...	h_{m-1}	h_{m-3}	...	h_1	

1^η περίπτωση: τα μαύρα σημεία διπλώνονται σύμφωνα με το 2.12b, ενώ τα υπόλοιπα με το 2.12c

άρτιος	h_0	h_2	...	h_k	h_{k-2}	...	h_2	h_0
περιττός	h_1	h_3	...	h_{k-1}	h_{k-3}	...	h_1	

Σε αυτήν την περίπτωση, από τα ζευγάρια, $\frac{m}{2}$ διπλώνονται σύμφωνα

με το 2.12b, ενώ $\frac{m}{2} + 1$ σύμφωνα με το 2.12c.

2^η περίπτωση: τα μαύρα σημεία διπλώνονται σύμφωνα με το 2.12b, ενώ τα υπόλοιπα με το 2.12c

άρτιος	h_0	h_2	...	h_m	h_{m-2}	...	h_2	h_0
περιττός	h_1	h_3	...	h_{m-1}	h_{m-3}	...	h_1	

Σε αυτήν την περίπτωση $\frac{m}{2} + 1$ ζευγάρια διπλώνονται σύμφωνα με το

2.12b, ενώ $\frac{m}{2}$ σύμφωνα με το 2.12c.

Βλέπουμε, λοιπόν, ότι αυτή η μορφή (επιλογή των ζευγαριών από το τέλος) είναι πιο αποδοτική όταν το m είναι άρτιο (2.25a) (Σημείωση: στην πολύπλεξη του h_0 με το θ , ο πολυπλέκτης στην πρόσθεση του σχ. 2.12c είναι στην ουσία μια σειρά από πύλες AND, ενώ αντίθετα, αν το h_m πολυπλέκοταν με το h_{m-1} , θα χρειαζόμασταν κανονικό πολυπλέκτη, γεγονός που θα αύξανε το υλικό)

m περιττό

Αν το m είναι περιττό, τότε οι ομαδοποιήσεις γινομένων (σημείων) σε ζευγάρια έχουν τη μορφή τη μορφή του παρακάτω πίνακα:

άρτιος	h_0	h_2	...	h_{m-1}	h_{m-1}	...	h_2	h_0
περιττός	h_1	h_3	...	h_m	h_{m-2}	...	h_1	

1^η περίπτωση: τα μαύρα σημεία διπλώνονται σύμφωνα με το 2.12b, ενώ τα υπόλοιπα με το 2.12c

άρτιος	h_0	h_2	...	h_{m-1}	h_{m-1}	...	h_2	h_0
περιττός	h_1	h_3	...	h_m	h_{m-2}	...	h_1	

Σε αυτήν την περίπτωση $\frac{m+1}{2}$ ζευγάρια διπλώνονται σύμφωνα με το

2.12b, ενώ $\frac{m+1}{2}$ σύμφωνα με το 2.12c

2^η περίπτωση: τα μαύρα σημεία διπλώνονται σύμφωνα με το 2.12b, ενώ τα υπόλοιπα με το 2.12c

άρτιος	h_0	h_2	...	h_{m-1}	h_{m-1}	...	h_2	h_0
περιττός	h_1	h_3	...	h_m	h_{m-2}	...	h_1	

Σε αυτήν την περίπτωση $\frac{m-1}{2}$ ζευγάρια διπλώνονται σύμφωνα με το

2.12b, ενώ $\frac{m+3}{2}$ σύμφωνα με το 2.12c

Βλέπουμε, λοιπόν, ότι η προηγούμενη μορφή (επιλογή των ζευγαριών από την αρχή) είναι πιο αποδοτική όταν το m είναι περιττό (2.25b)

Συνολικά, στην περίπτωση που ακολουθήσουμε τη βέλτιστη αρχιτεκτονική,

$\left\lceil \frac{m+1}{2} \right\rceil = \left\lceil \frac{k+1}{4} \right\rceil$ διπλώνονται σύμφωνα με το 2.2.12b και

$\left\lfloor \frac{m+1}{2} \right\rfloor = \left\lfloor \frac{k+1}{4} \right\rfloor$ σύμφωνα με το 2.2.12c. Άρα, οι συνολικές επιδόσεις της folded

transpose αρχιτεκτονικής περιττού συμμετρικού FIR φίλτρου με υποδειγματοληψία ανά δύο στοιχεία στην έξοδο είναι οι εξής (2.26):

-Υλικό: $\left\lceil \frac{k}{4} \right\rceil$ πολλαπλασιαστές,

$\left\lfloor \frac{k+1}{4} \right\rfloor + \left\lfloor \frac{k+1}{4} \right\rfloor - 1 + 1 = \frac{k+1}{2} = \left\lfloor \frac{k}{2} \right\rfloor$ αθροιστές,

$\left\lfloor \frac{k+1}{4} \right\rfloor + \left\lfloor \frac{k+1}{4} \right\rfloor + 1 = \frac{k+3}{2} = \left\lfloor \frac{k}{2} \right\rfloor + 1$ πολυπλέκτες,

$$2\left(\left\lceil \frac{k+1}{4} \right\rceil + \left\lfloor \frac{k+1}{4} \right\rfloor\right) - 2 + 1 = 2\frac{k+1}{2} - 1 = k + 1 - 1 = k \text{ καταχωρητές, } 1$$

υποδειγματολήπτης

-Κρίσιμο μονοπάτι:

- Αν m άρτιο, $T_M + T_{mux} + T_A + T_A + \max(T_{mux}, T_A)$
- Αν m περιττό, $T_{mux} + T_M + T_A + \max(T_{mux}, T_A)$

-Αναμενόμενη κατανάλωση ισχύος: $\left\lceil \frac{k}{4} \right\rceil P_M + \left(\left\lceil \frac{k}{2} \right\rceil + 1\right) P_{mux} + \left\lceil \frac{k}{2} \right\rceil P_A + kP_R$

Είδαμε, λοιπόν, και τους τρόπους με τους οποίους είναι δυνατό να αξιοποιήσουμε την transpose μορφή του φίλτρου. Ακολουθούν τα γενικότερα συμπεράσματα και συγκρίσεις μεταξύ των διαφορετικών αρχιτεκτονικών.

4. Συμπεράσματα

Στο μέρος 3, για κάθε μορφή φίλτρου (direct ή transpose) καταλήξαμε σε δύο αρχιτεκτονικές, μια με τα εσωτερικά στοιχεία να δουλεύουν σε μισό ρυθμό λειτουργίας, και μια folded με τα μισά (σχεδόν) σε αριθμό στοιχεία να δουλεύουν σε πλήρη ρυθμό λειτουργίας. Επιπλέον, αν έχουμε ζευγάρι περιττών φίλτρων, στις αρχιτεκτονικές που προέρχονται από την direct μορφή ωφελεί αυτά να διαμοιράζονται κοινή γραμμή καταχωρητών, όπως δείξαμε στις σχέσεις (2.20). Επίσης ενδεχομένως να ωφελεί να διατάξουμε τους αθροιστές σε δένδρο όπως προκύπτει από τις σχέσεις (2.21).

Έστω, λοιπόν, ότι θέλουμε να υλοποιήσουμε ένα συμμετρικό FIR φίλτρο με υποδειγματοληψία στην έξοδο, με $k=2m+1$ σημεία. Τότε, τα χαρακτηριστικά για τις έξι αρχιτεκτονικές στις οποίες καταλήξαμε, δίδονται συγκεντρωμένα στον πίνακα 2.3:

Πίνακας 2.3 Σύγκριση αρχιτεκτονικών για περιττό φίλτρο k στοιχείων

Αρχιτεκτονική	Σχ.	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Direct Half-Rate	2.4a	$(k-1)R + (k-1)A + \left\lceil \frac{k}{2} \right\rceil M + 2S$	$T_M + \left(\left\lceil \frac{k+1}{4} \right\rceil + 1\right)T_A$	$\left\lceil \frac{k}{2} \right\rceil \frac{P_M}{2} + k \frac{P_R}{2} + (k-1) \frac{P_A}{2}$
Direct Half-Rate +Tree	2.4a	$(k-1)R + (k-1)A + \left\lceil \frac{k}{2} \right\rceil M + 2S$	$T_M + \left(\left\lceil \log_2 \left\lceil \frac{k+1}{4} \right\rceil \right\rceil + 2\right)T_A$	$\left\lceil \frac{k}{2} \right\rceil \frac{P_M}{2} + k \frac{P_R}{2} + (k-1) \frac{P_A}{2}$
Direct Folded	2.6 2.7	$(k)R + \left\lceil \frac{k}{2} \right\rceil A + \left(\left\lceil \frac{k}{2} \right\rceil + 1\right) mux + \left\lceil \frac{k}{4} \right\rceil M + 1 S$	$T_{mux} + T_M + \left(\left\lceil \frac{k}{4} \right\rceil + 2\right)T_A$	$\left\lceil \frac{k}{4} \right\rceil P_M + \left(\left\lceil \frac{k}{2} \right\rceil + 1\right)P_{mux} + \left\lceil \frac{k}{2} \right\rceil P_A + kP_R$
Direct Folded + Tree	2.6 2.7	$(k)R + \left\lceil \frac{k}{2} \right\rceil A + \left(\left\lceil \frac{k}{2} \right\rceil + 1\right) mux + \left\lceil \frac{k}{4} \right\rceil M + 1 S$	$T_{mux} + T_M + \left(\left\lceil \log_2 \left\lceil \frac{k}{4} \right\rceil \right\rceil + 2\right)T_A$	$\left\lceil \frac{k}{4} \right\rceil P_M + \left(\left\lceil \frac{k}{2} \right\rceil + 1\right)P_{mux} + \left\lceil \frac{k}{2} \right\rceil P_A + kP_R$
Transpose Half-Rate	2.11b	$(k-1)R + (k-1)A + \left\lceil \frac{k}{2} \right\rceil M + 2S$	$T_M + 2T_A$	$\frac{1}{2} \left(\left\lceil \frac{k}{2} \right\rceil P_M + (k-1)P_A + kP_R\right)$
Transpose Folded	2.12	$(k)R + \left\lceil \frac{k}{2} \right\rceil A + \left(\left\lceil \frac{k}{2} \right\rceil + 1\right) mux + \left\lceil \frac{k}{4} \right\rceil M + 1 S$	$T_{mux} + T_M + T_A + \max(T_{mux}, T_A)$	$\left\lceil \frac{k}{4} \right\rceil P_M + \left(\left\lceil \frac{k}{2} \right\rceil + 1\right)P_{mux} + \left\lceil \frac{k}{2} \right\rceil P_A + kP_R$

Σημείωση: στην (2.16) ισχύει $k=2m+1$, m άρτιο, οπότε, $2 * \left\lfloor \frac{k}{4} \right\rfloor + 1 = 2 \left\lfloor \frac{2m+1}{4} \right\rfloor + 1 = 2 \frac{m}{2} + 1 = m+1 = \left\lfloor \frac{k}{2} \right\rfloor$, ενώ στην (2.18) $k=2m+3$, m άρτιο, οπότε, $2 * \left\lfloor \frac{k}{4} \right\rfloor + 2 = 2 \left\lfloor \frac{2m+3}{4} \right\rfloor + 2 = 2 \frac{m}{2} + 2 = m+2 = \left\lfloor \frac{k}{2} \right\rfloor$, και επιπλέον και στις δύο περιπτώσεις (αφού k περιττό) $\left\lfloor \frac{k}{4} \right\rfloor + 1 = \left\lfloor \frac{k}{4} \right\rfloor$. Με αυτόν τον τρόπο προκύπτουν οι προηγούμενες ενιαίες σχέσεις για τα direct folded φίλτρα

Ωστόσο, ο πίνακας 2.3 δε μας δείχνει το πλεονέκτημα των direct μορφών όταν έχουμε να υλοποιήσουμε ζευγάρια από φίλτρα, και ο μετασχηματισμός wavelet ενός σήματος έγκειται πάντα στη δημιουργία ενός τέτοιου ζευγαριού. Έστω λοιπόν ότι θέλουμε να υλοποιήσουμε ένα ζευγάρι τέτοιων φίλτρων, από τα οποία το πρώτο είναι t σημείων και το δεύτερο u σημείων, με t, u περιττά και $t > u$. Τότε, και σύμφωνα με τις (2.21) οι επιδόσεις της κάθε μίας αρχιτεκτονικής δίδονται στον πίνακα 2.4:

Πίνακας 2.4 Σύγκριση αρχιτεκτονικών για ζευγάρι περιττών φίλτρων t και u σημείων

Αρχιτεκτονική	Σχ.	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Direct Half-Rate	2.4a	$(t-1)R + (t+u-2)A + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right)M + 2S$	$T_M + \left(\left\lfloor \frac{t+1}{4} \right\rfloor + 1\right)T_A$	$\left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right) \frac{P_M}{2} + t \frac{P_R}{2} + (t+u-2) \frac{P_A}{2}$
Direct Half-Rate +Tree	2.4a	$(t-1)R + (t+u-2)A + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right)M + 2S$	$T_M + \left(\left\lfloor \log_2 \left\lfloor \frac{t+1}{4} \right\rfloor \right\rfloor + 2\right)T_A$	$\left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right) \frac{P_M}{2} + t \frac{P_R}{2} + (t+u-2) \frac{P_A}{2}$
Direct Folded	2.6 2.7	$(t+1)R + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right)A + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor + 2\right) \text{mux} + \left(\left\lfloor \frac{t}{4} \right\rfloor + \left\lfloor \frac{u}{4} \right\rfloor\right)M + 2S$	$T_{\text{mux}} + T_M + \left(\left\lfloor \frac{t}{4} \right\rfloor + 2\right)T_A$	$\left(\left\lfloor \frac{t}{4} \right\rfloor + \left\lfloor \frac{u}{4} \right\rfloor\right) P_M + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor + 2\right) P_{\text{mux}} + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right) P_A + (t+1)P_R$
Direct Folded + Tree	2.6 2.7	$(t+1)R + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right)A + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor + 2\right) \text{mux} + \left(\left\lfloor \frac{t}{4} \right\rfloor + \left\lfloor \frac{u}{4} \right\rfloor\right)M + 2S$	$T_{\text{mux}} + T_M + \left(\left\lfloor \log_2 \left\lfloor \frac{t}{4} \right\rfloor \right\rfloor + 2\right)T_A$	$\left(\left\lfloor \frac{t}{4} \right\rfloor + \left\lfloor \frac{u}{4} \right\rfloor\right) P_M + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor + 2\right) P_{\text{mux}} + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right) P_A + (t+1)P_R$
Transpose Half-Rate	2.11b	$(t+u-3)R + (t+u-2)A + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right)M + 1S$	$T_M + 2T_A$	$\left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right) \frac{P_M}{2} + (t+u-2) \frac{P_A}{2} + (t+u-2) \frac{P_R}{2}$
Transpose Folded	2.12	$(t+u)R + \left\{ \left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor \right\} A + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor + 2\right) \text{mux} + \left(\left\lfloor \frac{t}{4} \right\rfloor + \left\lfloor \frac{u}{4} \right\rfloor\right)M + 2S$	$T_{\text{mux}} + T_M + T_A + \max(T_{\text{mux}}, T_A)$	$\left(\left\lfloor \frac{t}{4} \right\rfloor + \left\lfloor \frac{u}{4} \right\rfloor\right) P_M + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor + 2\right) P_{\text{mux}} + \left(\left\lfloor \frac{t}{2} \right\rfloor + \left\lfloor \frac{u}{2} \right\rfloor\right) P_A + (t+u)P_R$

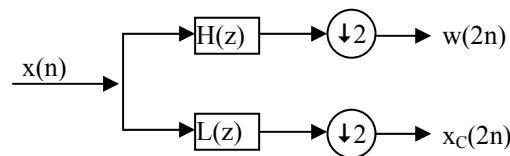
Σημείωση: στο transpose polyphase θεωρούμε ότι ο διαχωρισμός άρτιων-περιττών μπορεί να γίνει από την ίδια μονάδα καταχωρητής-δύο υποδειγματολήπτες (συνεπώς, εξοικονομούμε έναν καταχωρητή)

Ποια λοιπόν είναι η καλύτερη αρχιτεκτονική για να υλοποιήσουμε έναν διορθογώνιο μετασχηματισμό wavelet; Η απάντηση είναι εξαρτάται τι επιθυμούμε. Προφανώς, τις καλύτερες επιδόσεις στο θέμα της ταχύτητας τις έχουν οι transpose μορφές, από την άλλη όμως οι direct μορφές έχουν το πλεονέκτημα της σημαντικής εξοικονόμησης ενέργειας λόγω της χρησιμοποίησης των ίδιων καταχωρητών και από τα δύο φίλτρα.

Επιπλέον, οι επιδόσεις σε ισχύ μεταξύ half-rate και folded εκδοχών είναι σχεδόν παρόμοιες, με τις half-rate ενδεχομένως να υπερτερούν ελαφρά. Από την άλλη, οι folded αρχιτεκτονικές απαιτούν πολύ μικρότερο χώρο στο κύκλωμα, γεγονός που μπορεί να δράσει είτε θετικά είτε αρνητικά για την επιλογή τους. Για παράδειγμα, αν το μόνο που μας απασχολεί είναι ο χώρος στο κύκλωμα, προφανώς θα επιλέξουμε την folded αρχιτεκτονική. Αν όμως έχουμε πρόβλημα με την απαγωγή θερμότητας από το κύκλωμα, τότε θα προτιμήσουμε την half-rate αρχιτεκτονική, καθώς απαιτεί την ίδια ισχύ αλλά έχει αρκετά μεγαλύτερη επιφάνεια (και, άρα, καλύτερη απαγωγή θερμότητας)

Γ. Αρχιτεκτονικές βασισμένες στη συνέλιξη

Όπως αναφέραμε και στο προηγούμενο κεφάλαιο, η κυριότερη συνεισφορά του Mallat στον κλάδο των wavelets είναι η απόδειξη ότι ο μετασχηματισμός wavelet ενός σήματος ανάγεται στο φιλτράρισμά του από δύο κατάλληλα ψηφιακά φίλτρα, ένα υψιπερατό, και ένα βαθυπερατό, και κατόπιν στην υποδειγματοληγία της εξόδου του καθενός, έτσι ώστε το μεν υψιπερατό να μας δίδει τους συντελεστές του wavelet ανώτερης κλίμακας και το βαθυπερατό μια «εξομαλυμένη» εκδοχή του σήματος (δηλαδή, το σήμα χωρίς τις λεπτομέρειες που δίνονται από τους συντελεστές των wavelets που λάβαμε από το υψιπερατό φίλτρο), το οποίο δύναται στη συνέχεια να αναλυθεί και αυτό από ένα ίδιο ζεύγος φίλτρων και να λάβουμε τους συντελεστές των wavelet χαμηλότερων συχνοτήτων



Σχήμα 2.13 Διακριτός Μετασχηματισμός Wavelet

Στο παραπάνω σχήμα βλέπουμε μια σχηματική αναπαράσταση των προηγούμενων, όπου $H(z)$ το υψιπερατό φίλτρο, $L(z)$ το βαθυπερατό, $x(n)$ το αρχικό σήμα, $w(2n)$ οι συντελεστές των wavelet της υψηλότερης συχνότητας και $x_c(2n)$ η «εξομαλυμένη» εκδοχή του αρχικού σήματος. Για λεπτομερέστερη περιγραφή και επεξήγηση των παραπάνω παρακαλώ ανατρέξτε στο προηγούμενο κεφάλαιο.

Από τη θεωρία είναι γνωστό ότι για το wavelet CDF 9/7, τα φίλτρα αυτά έχουν συναρτήσεις μεταφοράς:

$$H(z) = h_6 z^{-1} + h_5 z^{-2} + h_4 z^{-3} + h_3 z^{-4} + h_2 z^{-5} + h_1 z^{-6} + h_0 z^{-7} \quad (2.26a)$$

και

$$L(z) = l_8 z^{-1} + l_7 z^{-2} + l_6 z^{-3} + l_5 z^{-4} + l_4 z^{-5} + l_3 z^{-6} + l_2 z^{-7} + l_1 z^{-8} + l_0 z^{-9} \quad (2.26b)$$

όπου, λόγω των συμμετρικών ιδιοτήτων του διορθωτικού wavelet CDF 9/7, ισχύει:

$$\begin{aligned} h_0 &= h_6 = -0.06453888262870 \\ h_1 &= h_5 = 0.04068941760916, \\ h_2 &= h_4 = 0.41809227322162, \\ h_3 &= -0.78848561640558 \end{aligned} \quad (2.27a)$$

και,

$$\begin{aligned} l_0 &= l_8 = 0.037828455507264, \\ l_1 &= l_7 = -0.0238494650195568, \\ l_2 &= l_6 = -0.110624404418437, \\ l_3 &= l_5 = 0.377402855612831 \text{ και τέλος} \\ l_4 &= 0.852698679008894 \end{aligned} \quad (2.27b)$$

Είναι προφανές ότι για να επιλέξουμε την βέλτιστη αρχιτεκτονική υλοποίηση των δύο αυτών φίλτρων που πραγματοποιούν το μετασχηματισμό, αρκεί να εφαρμόσουμε στην παρούσα περίπτωση τα συμπεράσματα της προηγούμενης ενότητας. Πιο συγκεκριμένα, θέτοντας στον πίνακα 2.4 $t=9$ και $u=7$, έχω τα παρακάτω συγκριτικά αποτελέσματα:

Πίνακας 2.5 Σύγκριση αρχιτεκτονικών βασισμένων στη συνέλιξη του CDF 9/7

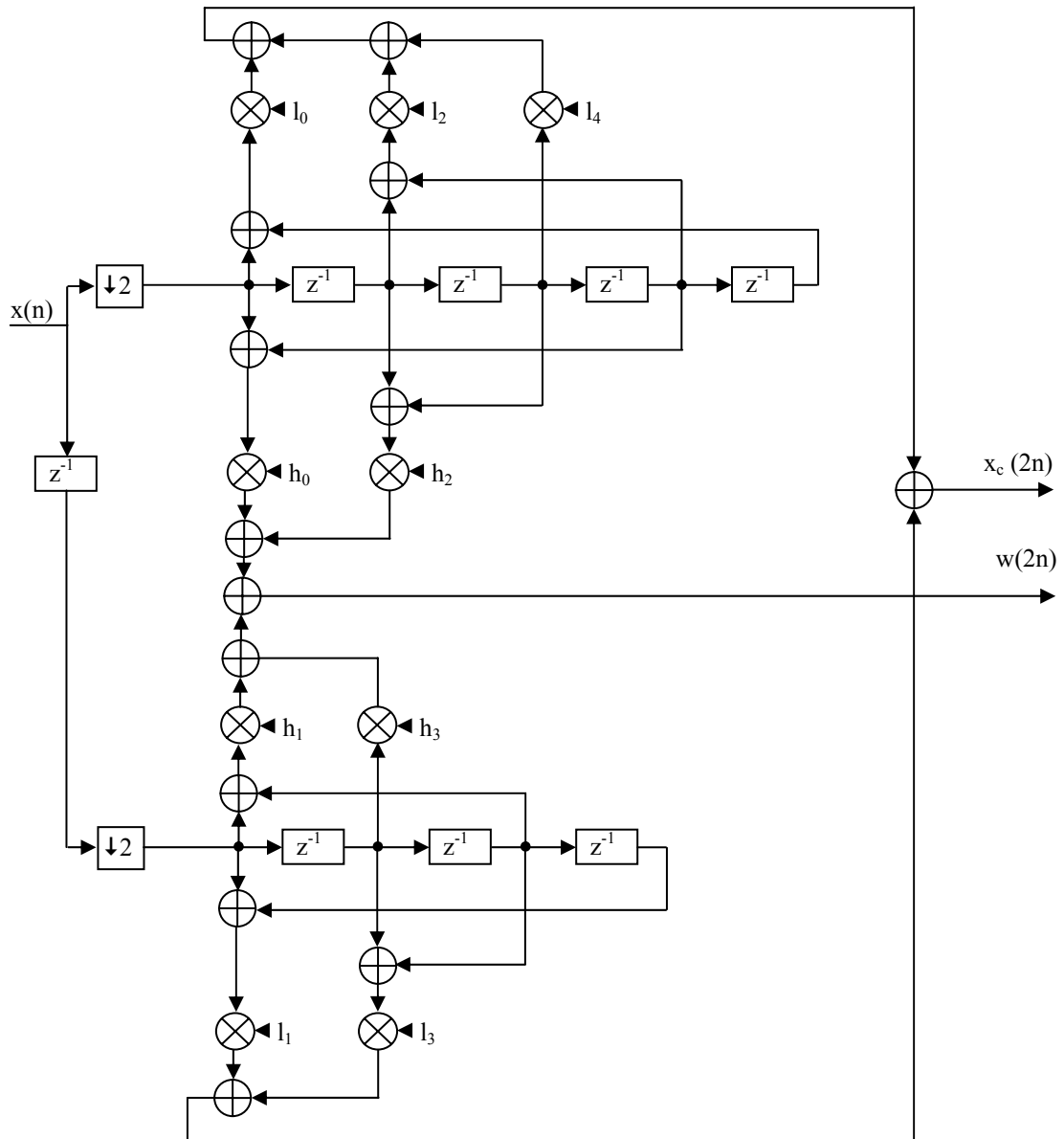
Αρχιτεκτονική	Σχ.	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Direct Half-Rate	2.3.2	8 R + 14A + 9M + 2S	$T_M + 4T_A$	$4.5P_M + 4.5P_R + 7P_A$
Direct Half-Rate +Tree	-	8 R + 14A + 9M + 2S	$T_M + 4T_A$	$4.5P_M + 4.5P_R + 7P_A$
Direct Folded	2.3.3	10 R + 9A + 11 mux + 5M + 2 S	$T_{mux} + T_M + 4T_A$	$5P_M + 11P_{mux} + 9P_A + 10P_R$
Direct Folded + Tree	-	10 R + 9 A + 11 mux + 5 M + 2 S	$T_{mux} + T_M + 4T_A$	$5P_M + 11P_{mux} + 9P_A + 10P_R$
Transpose Half-Rate	2.3.4	13 R + 14 A + 9 M + 1 S	$T_M + 2T_A$	$4.5P_M + 7P_A + 7P_R$
Transpose Folded	2.3.5	16 R + 9A + 11 mux + 5M + 2 S	$T_{mux} + T_M + T_A + \max(T_{mux}, T_A)$	$5P_M + 11P_{mux} + 9P_A + 16P_R$

Το πρώτο συμπέρασμα που βγαίνει από τον παραπάνω πίνακα είναι ότι, στην περίπτωση μας, η οργάνωση των αθροιστών σε δένδρο δε μειώνει το κρίσιμο μονοπάτι. Συνεπώς, δε θα ασχοληθούμε με τέτοιες υλοποιήσεις.

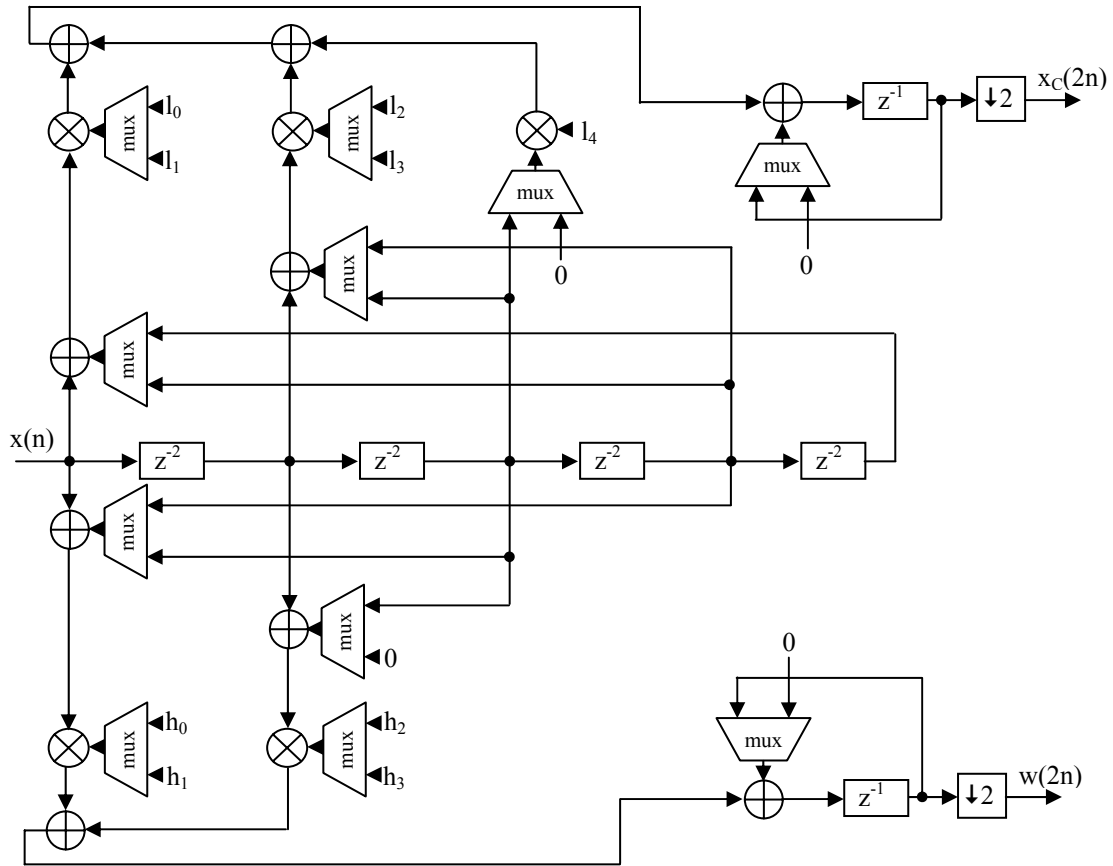
Επιπλέον, σε κάθε περίπτωση παρατηρούμε ότι η Transpose μορφή προσφέρει κρίσιμο μονοπάτι μόλις κατά δύο αθροιστές μικρότερο, και αυτό έρχεται στο κόστος της αύξησης κατά πάνω 50% των απαιτούμενων καταχωρητών, γεγονός που οδηγεί σε αύξηση της επιφάνειας και της κατανάλωσης του κυκλώματος. Επιπλέον, όσον αφορά τη σύγκριση μεταξύ folded και half-rate αρχιτεκτονικών, οι half-rate αναμένεται να έχουν κατά τι μικρότερη κατανάλωση ισχύος, χωρίς να επιβαρύνουν το κρίσιμο μονοπάτι.

Επομένως, στο επόμενο κεφάλαιο θα αναπτύξουμε σε χαμηλότερο επίπεδο μόνο τις Half-Rate αρχιτεκτονικές. Αυτό δε σημαίνει, φυσικά, ότι οι άλλες αρχιτεκτονικές δεν έχουν εφαρμογή. Αν, για παράδειγμα, μας ενδιαφέρει να μεγιστοποιήσουμε την ταχύτητα με οποιοδήποτε κόστος σε χώρο/ισχύ, ή ο χώρος στο ολοκληρωμένο είναι η σημαντικότερη παράμετρος, ενδεχομένως να επιλέξουμε μια άλλη αρχιτεκτονική.

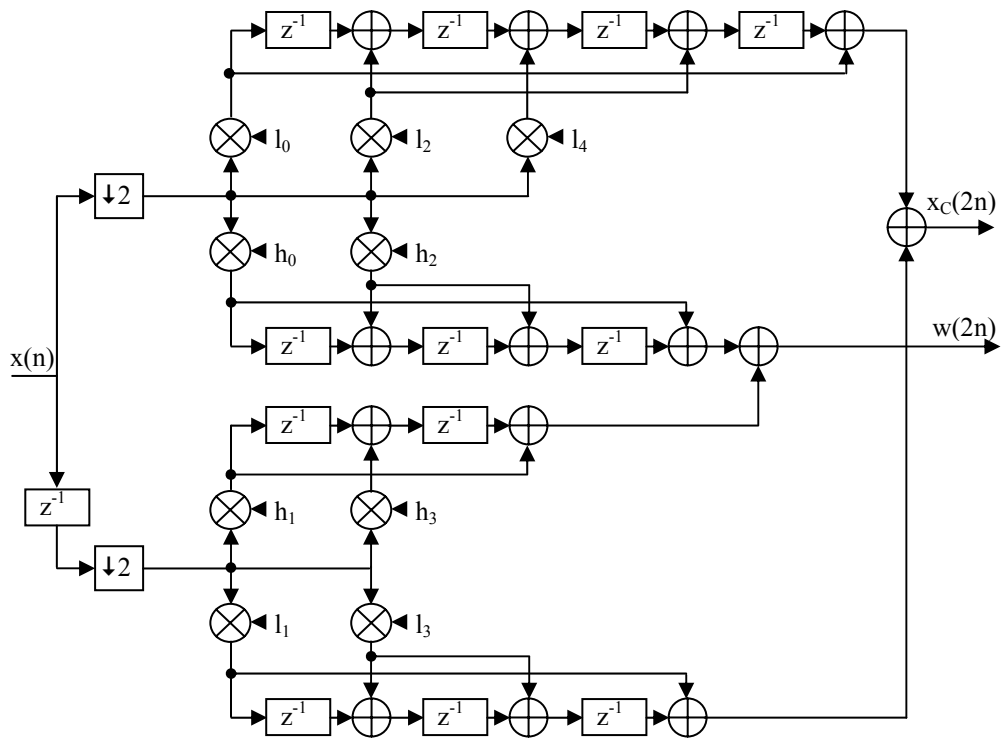
Ακολουθούν (όπως δίνονται και στον παραπάνω πίνακα) τα σχήματα κάθε μιας από τις τέσσερις επικρατέστερες αρχιτεκτονικές βασισμένες στη συνέλιξη.



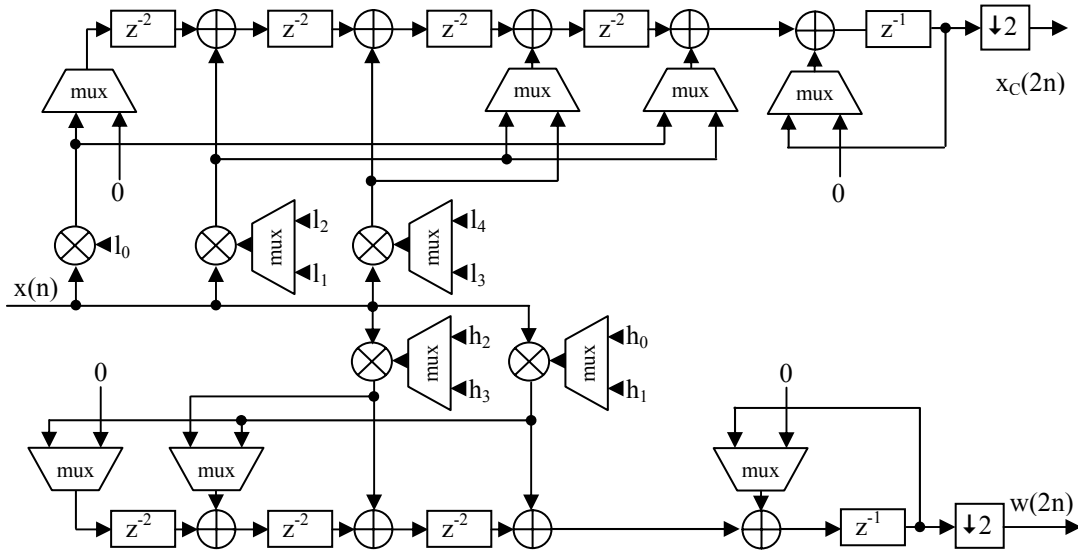
Σχήμα 2.14 Direct Half-Rate του CDF 9/7



Σχήμα 2.15 Direct Folded του CDF 9/7



Σχήμα 2.16 Transpose Half-Rate του CDF 9/7



Σχήμα 2.17 Transpose Folded του CDF 9/7

Δ. Αρχιτεκτονικές βασισμένες στη b-spline παραγοντοποίηση

1. Γενικά

Στην προηγούμενη ενότητα παρουσιάσαμε μια σειρά από βελτιστοποιημένες αρχιτεκτονικές ζευγών FIR φίλτρων που υλοποιούν το μετασχηματισμό wavelet CDF 9/7. Φυσικά, οι τοπολογίες αυτές θα μπορούσαν με πολύ μικρές τροποποιήσεις να εφαρμοστούν σε οποιοδήποτε ζεύγος περιττών συμμετρικών FIR φίλτρων με υποδειγματοληψία στην έξοδο, και όχι μόνο σε φίλτρα που πραγματοποιούν μετασχηματισμούς wavelet.

Στην ενότητα αυτή θα εκμεταλλευτούμε ειδικά μαθηματικά θεωρήματα που ισχύουν **μόνο** για ζεύγη συμπληρωματικών φίλτρων που πραγματοποιούν μετασχηματισμούς wavelet. Η σχετική μαθηματική θεωρία είναι μάλλον πρόσφατη (2003) και εξηγείται για κάθε ενδιαφερόμενο από τους Unser και Blu στο [D1]. Η πρώτη απόπειρα χρήσης της τεχνικής b-spline factorization στην υλοποίηση μετασχηματισμών wavelet σε υλικό έγινε από τους Huang et. al. Από το 2005 και ύστερα στα [D2] και [D3] και τα αποτελέσματα αυτής της έρευνας θα παρουσιάσουμε εδώ.

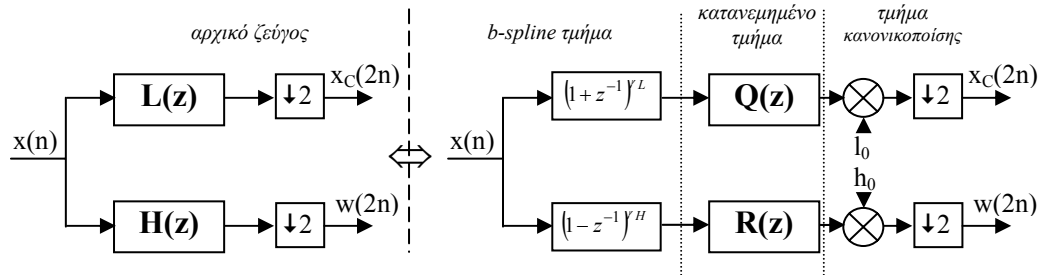
Σύμφωνα, λοιπόν, με τη θεωρία των Unser και Blu [1], το βαθυπερατό φίλτρο $L(z)$ και το υψιπερατό $H(z)$ ενός μετασχηματισμού wavelet δύνανται να παραγοντοποιηθούν στη μορφή:

$$L(z) = (1 + z^{-1})^{\gamma_L} * Q(z) * l_0 \quad (2.28a)$$

$$\text{και } H(z) = (1 - z^{-1})^{\gamma_H} * R(z) * h_0 \quad (2.28b)$$

όπου $Q(z)$, $R(z)$ μικρότερης τάξης φίλτρα από τα $L(z)$, $H(z)$ αντίστοιχα. Όπως φαίνεται και στο σχ. 2.4.1, η παραγοντοποίηση αυτή μπορεί να οδηγήσει σε αποδοτικές υλοποιήσεις, καθώς τα μεν $Q(z)$ και $R(z)$ είναι φίλτρα πολύ λιγότερων

σημείων από τα $L(z)$ και $H(z)$, άρα χρειάζονται λιγότερους πολλαπλασιαστές, το δε «πολυωνυμικό» κομμάτι $(1+z^{-1})^k$ μπορεί να υλοποιηθεί είτε ως μια σειρά διαδοχικών καθυστερήσεων και προσθέσεων είτε ως ένα εκφυλισμένο FIR φίλτρο που δεν απαιτεί πολλαπλασιαστές. Δηλαδή, το b-spline factorization μειώνει τον αριθμό των πολλαπλασιαστών, αυξάνοντας τον αριθμό των αθροιστών. Το αν το τελικό αποτέλεσμα είναι καλύτερο από άποψης επιδόσεων είναι κάτι που θα εξεταστεί παρακάτω.



Σχήμα 2.18 Αρχή b-spline παραγοντοποίησης

Στη βιβλιογραφία, το τμήμα $(1+z^{-1})^k$ ονομάζεται b-spline part, το τμήμα $Q(z)$ distributed part και ο όρος l_0 normalization part. Στα ελληνικά οι όροι αυτοί μπορούν να αποδοθούν ως τμήμα b-spline, καταναμημένο τμήμα και τμήμα κανονικοποίησης. Την ορολογία αυτή θα τη χρησιμοποιήσουμε στο εφεξής. Παρατηρούμε στο σχ. 2.18, ότι αν ενσωματώσουμε το τμήμα κανονικοποίησης στο καταναμημένο τμήμα, τα $Q(z)$ και $R(z)$ αποτελούν ένα ζεύγος FIR φίλτρων με υποδειγματοληψία στην έξοδο, συνεπώς, αν και εφ'όσον είναι συμμετρικά μπορούμε να εφαρμόσουμε άμεσα σε αυτά τα συμπεράσματα της ενότητας (B). Η μόνη διαφορά που υπάρχει εδώ είναι ότι επειδή τα δύο φίλτρα δεν τροφοδοτούνται από το ίδιο σήμα εισόδου (διότι το εκάστοτε b-spline τμήμα είναι διαφορετικό) δεν μπορούμε στην direct μορφή να επιτύχουμε οικονομία στους καταχωρητές, μέσω της χρήσης κοινής αλυσίδας.

Σύμφωνα, λοιπόν, με το [D2] το ζεύγος των φίλτρων του μετασχηματισμού CDF 9/7 μπορεί να γραφεί στη μορφή:

$$L(z) = (1+z^{-1})^4 (1+t_1 z^{-1} + t_2 z^{-2} + t_1 z^{-3} + z^{-4}) l_0 \quad (2.29a)$$

$$\text{και } H(z) = (1-z^{-1})^4 (1+t_3 z^{-1} + z^{-2}) h_0 \quad (2.29b)$$

όπου

$$t_1 = -4.630464$$

$$t_2 = 9.597484$$

$$t_3 = 3.369536$$

$$l_0 = 0.037828455507264$$

$$h_0 = -0.06453888262870$$

$$(2.29c)$$

Για λόγους μείωσης του κρίσιμου μονοπατιού θα ενσωματώσουμε τα τμήματα κανονικοποίησης στα καταναμημένα τμήματα, και επιπλέον θα αναπτύξω τα b-spline τμήματα. Έχουμε:

$$L_{bs}(z) = (1+z^{-1})^4 = 1 + 4z^{-1} + 6z^{-2} + 4z^{-3} + z^{-4} \quad (2.30a)$$

$$\text{και, } H_{bs}(z) = (1-z^{-1})^4 = 1 - 4z^{-1} + 6z^{-2} - 4z^{-3} + z^{-4} \quad (2.30b)$$

οπότε, την (2.29) ξαναγράφεται ως:

$$L(z) = (1 + 4z^{-1} + 6z^2 + 4z^{-3} + z^{-4})(l_0 + l_0 t_1 z^{-1} + l_0 t_2 z^{-2} + l_0 t_1 z^{-3} + l_0 z^{-4}) \quad (2.31a)$$

$$\text{και } H(z) = (1 - 4z^{-1} + 6z^2 - 4z^{-3} + z^{-4})(h_0 + h_0 t_3 z^{-1} + h_0 z^{-2}) \quad (2.31b)$$

Βλέπουμε, λοιπόν, ότι τα συμμετρικά φίλτρα παραγοντοποιούνται σε δύο συμμετρικά υπο-φίλτρα. Αυτό σημαίνει ότι μπορούμε σε αυτά να εφαρμόσουμε άμεσα τα συμπεράσματα της ενότητας (B) για την αποδοτικότερη υλοποίηση αυτών. Θα ξεκινήσουμε πρώτα με το καταναμημένο τμήμα και στη συνέχεια θα προχωρήσουμε στο b-spline τμήμα

2. Καταναμημένο τμήμα

Σύμφωνα με τις (2.31a), (2.31b) και το σχ. 2.18, τα καταναμημένα τμήματα των φίλτρων που απαιτούνται για τον CDF 9/7 είναι ένα ζεύγος συμμετρικών FIR φίλτρων με υποδειγματοληψία στην έξοδο, τα οποία όμως **δεν** έχουν την ίδια είσοδο. Συνεπώς, για το κάθε ένα από αυτά, έχω από τον πίνακα 2.3:

Πίνακας 2.6 Σύγκριση αρχιτεκτονικών για συμμετρικό φίλτρο 3 σημείων

Αρχιτεκτονική	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Direct Half-Rate	2 R + 2A + 2M + 2S	$T_M + 2T_A$	$P_M + 1.5P_R + P_A$
Direct Half-Rate +Tree	2 R + 2A + 2M + 2S	$T_M + 2T_A$	$P_M + 1.5P_R + P_A$
Direct Folded	3 R + 2 A + 3 mux + 1M + 1 S	$T_{mux} + T_M + 2T_A$	$P_M + 3P_{mux} + 2P_A + 3P_R$
Direct Folded + Tree	3 R + 2 A + 3 mux + 1M + 1 S	$T_{mux} + T_M + 2T_A$	$P_M + 3P_{mux} + 2P_A + 3P_R$
Transpose Half-Rate	2 R + 2 A + 2M + 2 S	$T_M + 2T_A$	$P_M + P_A + 1.5P_R$
Transpose Folded	3 R + 2A + 3 mux + 1 M + 1 S	$T_{mux} + T_M + T_A + \max(T_{mux}, T_A)$	$P_M + 3P_{mux} + 2P_A + 3P_R$

Πίνακας 2.7 Σύγκριση αρχιτεκτονικών για συμμετρικό φίλτρο 5 σημείων

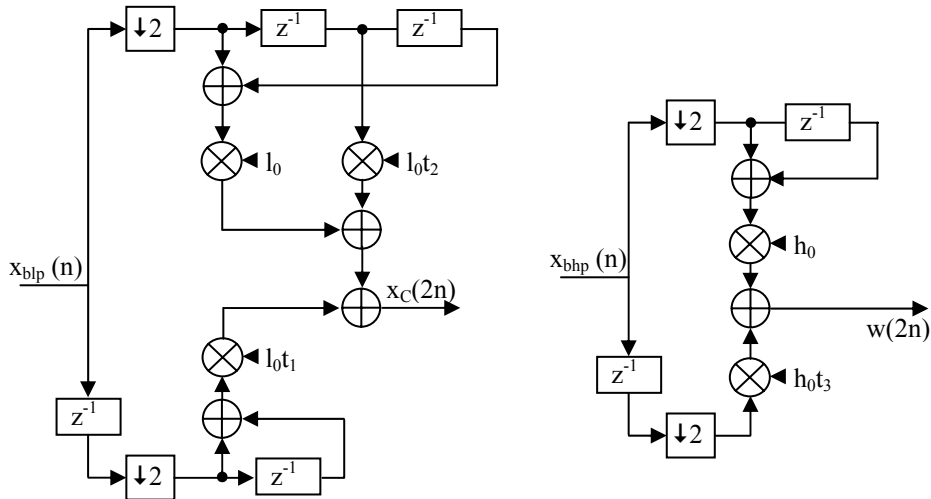
Αρχιτεκτονική	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Direct Half-Rate	4 R + 4A + 3M + 2S	$T_M + 3T_A$	$1.5P_M + 2.5P_R + 2P_A$
Direct Half-Rate +Tree	4 R + 4A + 3M + 2S	$T_M + 3T_A$	$1.5P_M + 2.5P_R + 2P_A$
Direct Folded	5 R + 3A + 4 mux + 2 M + 1 S	$T_{mux} + T_M + 3T_A$	$2P_M + 4P_{mux} + 3P_A + 5P_R$
Direct Folded + Tree	5 R + 3A + 4 mux + 2 M + 1 S	$T_{mux} + T_M + 3T_A$	$2P_M + 4P_{mux} + 3P_A + 5P_R$
Transpose Half-Rate	4 R + 4 A + 3 M + 2 S	$T_M + 2T_A$	$1.5P_M + 2P_A + 2.5P_R$
Transpose Folded	5 R + 3A + 4 mux + 2M + 1 S	$T_{mux} + T_M + T_A + \max(T_{mux}, T_A)$	$2P_M + 4P_{mux} + 3P_A + 5P_R$

Από τους παραπάνω πίνακες βλέπουμε ότι (και εδώ) η οργάνωση των αθροιστών σε δένδρο δε μειώνει το κρίσιμο μονοπάτι, οπότε, συγκεντρωτικά, έχω και για τα δύο φίλτρα:

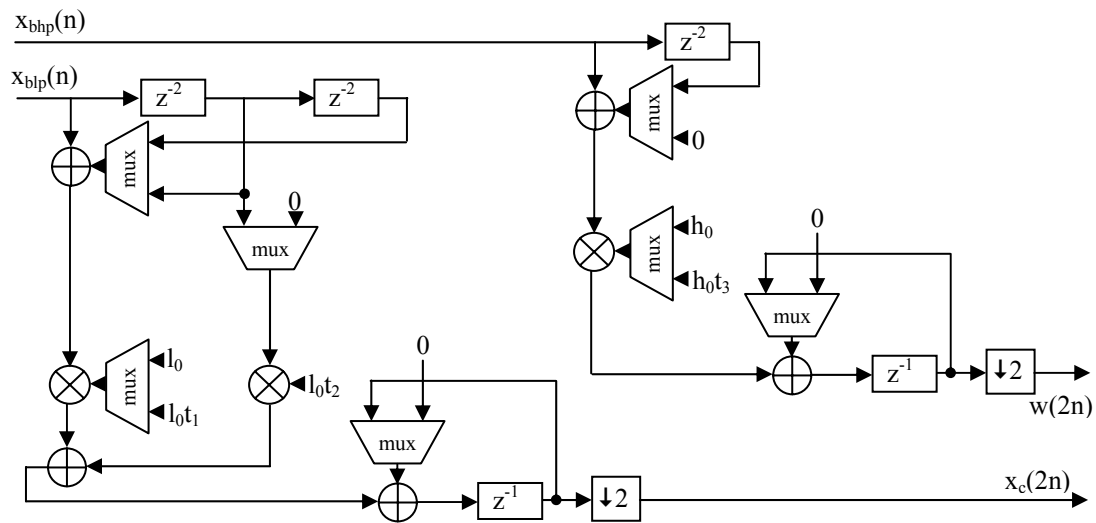
Πίνακας 2.8 Σύγκριση αρχιτεκτονικών του καταναμημένου τμήματος του CDF 9/7

Αρχιτεκτονική	Σχ.	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Direct Half-Rate	2.19	6 R + 6A + 5M + 4S	$T_M + 3T_A$	$2.5P_M + 4P_R + 3P_A$
Direct Folded	2.20	8 R + 5 A + 7 mux + 3 M + 2 S	$T_{mux} + T_M + 3T_A$	$3P_M + 7P_{mux} + 5P_A + 8P_R$
Transpose Half-Rate	2.21	6 R + 6 A + 5 M + 4 S	$T_M + 2T_A$	$2.5P_M + 3P_A + 4P_R$
Transpose Folded	2.22	8 R + 5A + 7 mux + 3 M + 2 S	$T_{mux} + T_M + T_A + \max(T_{mux}, T_A)$	$3P_M + 7P_{mux} + 5P_A + 8P_R$

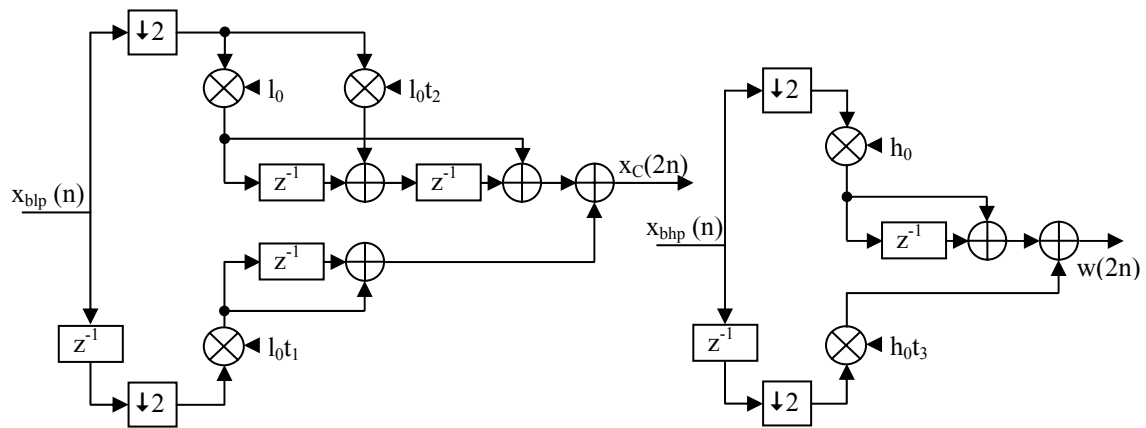
Παρατηρούμε ότι οι transpose αρχιτεκτονικές προσφέρουν πάντοτε μειωμένο κρίσιμο μονοπάτι σε σχέση με τις αντίστοιχες direct, χωρίς να απαιτούν περισσότερο υλικό ή ισχύ. Συνεπώς, το κατανεμημένο τμήμα θα υλοποιηθεί σε transpose μορφή. Ακολουθούν τα σχήματα των κυκλωμάτων του πίνακα 2.8.



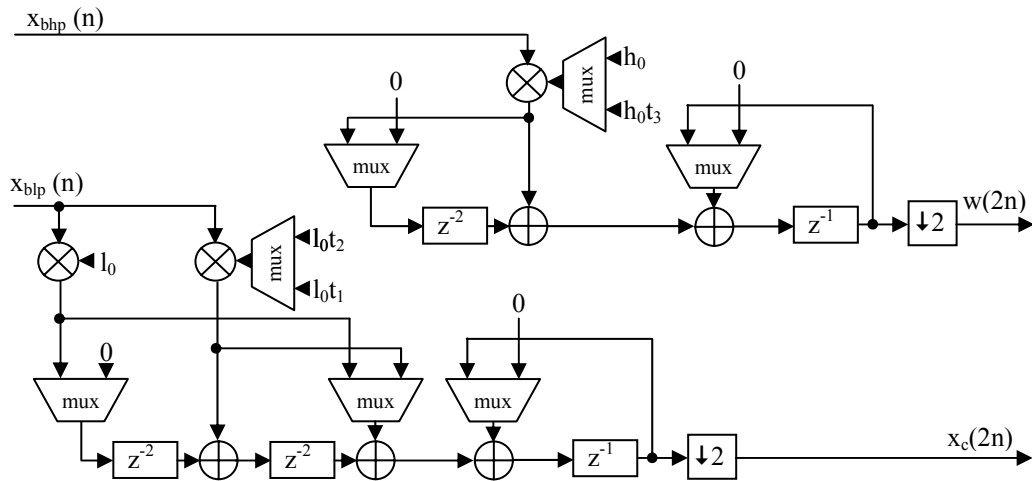
Σχήμα 2.19 Direct Half-Rate του κατανεμημένου τμήματος του CDF 9/7



Σχήμα 2.20 Direct Folded του κατανεμημένου τμήματος του CDF 9/7



Σχήμα 2.21 Transpose Half-Rate του καταμεμημένου τμήματος του CDF 9/7



Σχήμα 2.22 Transpose Folded του καταμεμημένου τμήματος του CDF 9/7

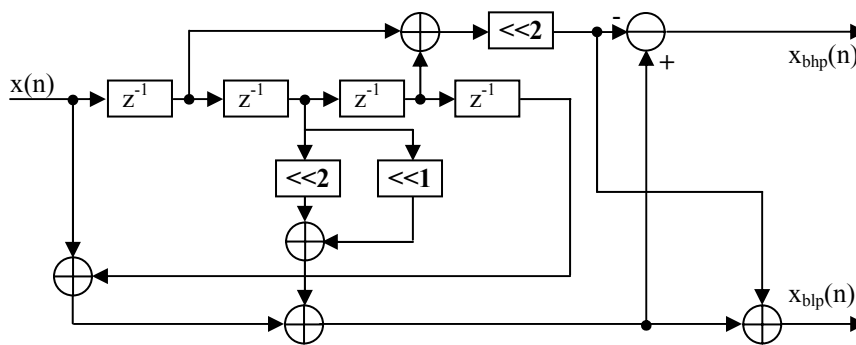
3. B-Spline Τμήμα

Από τα 2.30 έχω ότι τα φίλτρα που απαρτίζουν το b-spline τμήμα για το CDF 9/7 δίδονται από τις συναρτήσεις μεταφοράς:

$$L_{bs}(z) = 1 + 4z^{-1} + 6z^2 + 4z^{-3} + z^{-4}$$

$$H_{bs}(z) = 1 - 4z^{-1} + 6z^2 - 4z^{-3} + z^{-4}$$

Παρατηρούμε ότι τα φίλτρα είναι συμμετρικά και επιπλέον οι συντελεστές των σημείων τους είναι δυνάμεις του 2 ή άθροισμα αυτών ($6=4+2$). Συνεπώς, ο πολλαπλασιασμός με τους συντελεστές μπορεί να γίνει με μετατόπιση των bits (που είναι «δωρεάν» στο υλικό) και προσθέτες, χωρίς να απαιτούνται πολλαπλασιαστές. Επιπρόσθετα, βλέπω ότι τα φίλτρα είναι ίδια, εκτός από τους συντελεστές 1^{ns} και 3^{ns} τάξης, που είναι αντίστροφοι. Τέλος, τα δύο φίλτρα έχουν κοινή είσοδο, επομένως σε direct μορφή μπορώ να επιτύχω οικονομία στα στοιχεία καθυστέρησης (καταχωρητές) χρησιμοποιώντας κοινή γραμμή καταχωρητών. Από όλα τα προηγούμενα, υλοποιούμε το ζεύγος σε direct μορφή όπως φαίνεται στο ακόλουθο σχήμα:

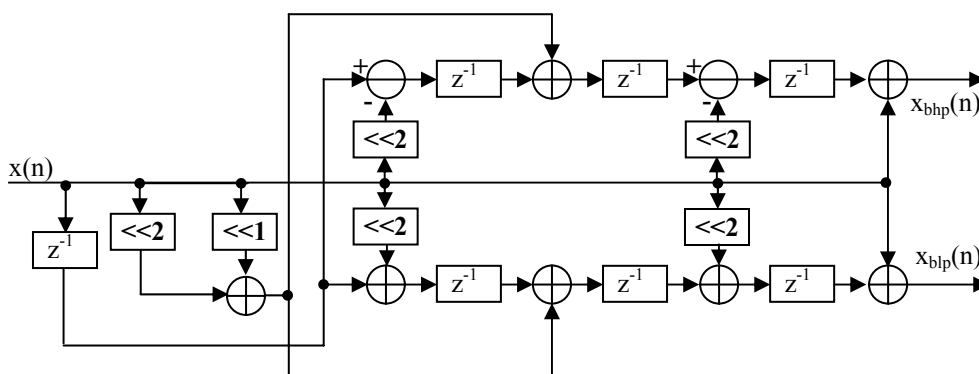


Σχήμα 2.23 Direct υλοποίηση του b-spline τμήματος του CDF9/7

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.31)

- Πλήθος στοιχείων: 4 καταχωρητές, 5 αθροιστές, 1 αφαιρέτης
- Το κρίσιμο μονοπάτι ισούται με $3T_A$ (θεωρούμε ότι ο αφαιρέτης εισάγει την ίδια καθυστέρηση με τον αθροιστή)
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται ισούται με $4P_R+5P_A+P_{SB}$, όπου P_{SB} η αναμενόμενη ισχύς του αφαιρέτη

Εναλλακτικά, η υλοποίηση των φίλτρων σε transpose μορφή έχει ως εξής:



Σχήμα 2.24 Transpose υλοποίηση του b-spline τμήματος του CDF9/7

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.32)

- Πλήθος στοιχείων: 7 καταχωρητές, 7 αθροιστές, 2 αφαιρέτες
- Το κρίσιμο μονοπάτι ισούται με $2T_A$ (και λόγω των μεγαλύτερων κρίσιμων μονοπατιών του καταναμημένου τμήματος, δεν έχει νόημα η μείωσή του, αφού δεν προστίθεται σε αυτά)
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται ισούται με $7P_R+7P_A+2P_{SB}$

Παρατηρούμε ότι η transpose υλοποίηση, παρά το ότι έχει μικρότερο κρίσιμο μονοπάτι (το οποίο, μάλιστα, δεν προστίθεται στο αντίστοιχο του καταναμημένου τμήματος) έχει ιδιαίτερα αυξημένες απαιτήσεις σε υλικό και ισχύ, συγκρινόμενη με την direct.

4. Συνολικά αποτελέσματα

Από τα χαρακτηριστικά των transpose καταναμημένων τμημάτων (πίνακας 2.8) και από τις (2.31), (2.32) παίρνω τον παρακάτω πίνακα που περιγράφει τις επιδόσεις των τελικών αρχιτεκτονικών βασισμένων στη b-spline παραγοντοποίηση:

Πίνακας 2.9 Σύγκριση b-spline-based αρχιτεκτονικών για το CDF 9/7

Αρχιτεκτονική	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Half-Rate κατανεμημένο, Direct b-spline	10 R + 11 A + 1SB + 5 M + 4 S	$T_M + 5T_A$	$2.5P_M + 8P_A + 8P_R + P_{SB}$
Half-Rate κατανεμημένο, Transpose b-spline	13 R + 13 A + 2 SB + 5 M + 4 S	$T_M + 3T_A$	$2.5P_M + 10P_A + 2P_{SB} + 11P_R$
Folded κατανεμημένο, Direct b-spline	12 R + 10A + 1SB + 7 mux + 3 M + 2 S	$T_{mux} + T_M + 5T_A$	$3P_M + 7P_{mux} + 10P_A + 12P_R +$ $+ P_{SB}$
Folded κατανεμημένο, Transpose b-spline	15 R + 12A + 2SB + 7 mux + 3 M + 2 S	$T_{mux} + T_M + 3T_A$	$3P_M + 7P_{mux} + 12P_A + 15P_R +$ $+ 2P_{SB}$

Παρατηρούμε ότι οι αρχιτεκτονικές με transpose b-spline προσφέρουν μικρότερο κρίσιμο μονοπάτι κατά $2T_A$, ωστόσο αυτό γίνεται με επιβάρυνση σε κύκλωμα και σε κατανάλωση ισχύος. Επομένως, και παρά το ελαφρά μεγαλύτερο κρίσιμο μονοπάτι τους, ως καταλληλότερες για περαιτέρω διερεύνηση στο επόμενο κεφάλαιο κρίνονται οι δύο αρχιτεκτονικές που έχουν το b-spline τμήμα τους σε direct μορφή. Φυσικά, σε εφαρμογές όπου το πρώτιστο μέλημα, με οποιοδήποτε κόστος, είναι η ταχύτητα, αναμένουμε να επιλεγούν τα κυκλώματα όπου το b-spline τμήμα είναι σε transpose μορφή

Ε. Αρχιτεκτονικές βασισμένες στο Lifting Scheme

Στα προηγούμενα κεφάλαια είδαμε πως μπορούμε με διάφορες τεχνικές να απλοποιήσουμε τις αρχιτεκτονικές των φίλτρων που εκτελούν τον CDF 9/7 ή και τα ίδια τα φίλτρα. Στο κεφάλαιο αυτό θα δούμε μια εναλλακτική αρχιτεκτονική που εκμεταλλεύεται τις ιδιότητες του ζεύγους των φίλτρων προκειμένου να τα ενοποιήσει.

Σύμφωνα με τα όσα αναφέραμε στη θεωρία και στις προηγούμενες ενότητες, ο μετασχηματισμός wavelet ανάγεται κατ' ουσίαν στο φιλτράρισμα του σήματος εισόδου από δύο παράλληλα φίλτρα, ένα βαθυπερατό και ένα υψιπερατό, και κατόπιν τον υποδειγματισμό των εξόδων τους ανά δεύτερο δείγμα (βλ. και σχ. 2.18). Το σύστημα εξισώσεων που περιγράφει την λειτουργία του μετασχηματισμού είναι το παρακάτω:

$$w(2n) = \sum_i h_i * x(2n - i) \quad (2.33a)$$

$$x_c(2n) = \sum_i l_i * x(2n - i) \quad (2.33b)$$

Στην ανάλυση των συμμετρικών FIR φίλτρων με υποδειματοληψία στην έξοδο, αναφέραμε το polyphase decomposition, καταλήγοντας εμμέσως στο ότι στα σήματα εξόδου υπ' όπιν λαμβάνονται μόνο τα γινόμενα των περιττών όρων των φίλτρων με τα περιττά (σε σειρά άφιξης) δείγματα του x , και των άρτιων όρων των φίλτρων με τα άρτια (σε σειρά άφιξης) δείγματα του x . Αν $h_e(z)$ και $h_o(z)$ τα άρτια και τα υποφίλτρα που προκύπτουν από τους άρτιους και τους περιττούς όρους του υψιπερατού $h(z)$ αντίστοιχα, και $l_e(z)$ και $l_o(z)$ τα αντίστοιχα υποφίλτρα του $l(z)$, τότε στο χώρο κατάστασης παίρνω (με βάση την προηγούμενη ανάλυση) ότι:

$$h(z) = h_e(z^2) + z^{-1}h_o(z^2) \quad (2.34a)$$

$$\text{και, } l(z) = l_e(z^2) + z^{-1}l_o(z^2) \quad (2.34b)$$

συνεπώς, μετά από την εκτέλεση πράξεων, όπως αποδεικνύεται και στα [C3],[C4], η λειτουργία του μετασχηματισμού δίδεται στο χώρο κατάστασης από το γινόμενο

$$\begin{bmatrix} x_c(z) \\ w(z) \end{bmatrix} = \begin{bmatrix} l_e(z) & l_o(z) \\ h_e(z) & h_o(z) \end{bmatrix} \begin{bmatrix} x_e(z) \\ z^{-1}x_o(z) \end{bmatrix} \quad (2.35)$$

Ένα χαρακτηριστικό των δύο αυτών φίλτρων, του h και του l που δεν εκμεταλλευτήκαμε καθόλου ως τώρα είναι ότι είναι συμπληρωματικά και ότι μπορούν, με τα αντίστροφά τους, να δώσουν τέλεια ανακατασκευή του σήματος. Η συμπληρωματικότητα έγκειται στο ότι στο σύνολό τους και τα δύο φίλτρα είναι ολοπερατά, δηλαδή οι συχνότητες που αποκόπτει το ένα επιτρέπει να περάσουν το άλλο και αντίστροφα, ενώ η τέλεια ανακατασκευή σημαίνει ότι αν λάβουμε τα σήματα εξόδου τους, τα περάσουμε από τα αντίστροφα φίλτρα και τα προσθέσουμε κατάλληλα με υπερδειγματοληψία, θα λάβουμε ακριβώς το αρχικό μας σήμα.

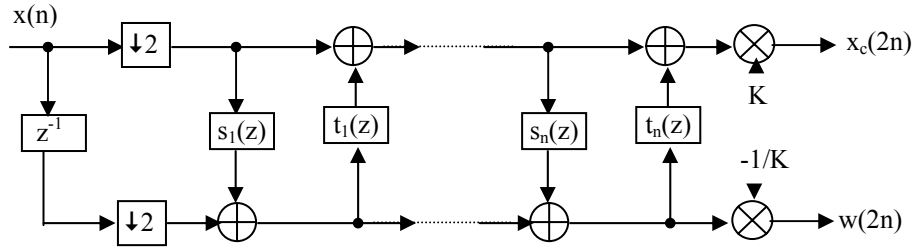
Το 1996 στο [C6] ο Wim Sweldens βρήκε (μαθηματικά) έναν αποδοτικό τρόπο εκμετάλλευσης των κοινών σημείων δύο φίλτρων που τηρούν τις προδιαγραφές συμπληρωματικότητας και τέλει ανακατασκευής (όλα τα φίλτρα που χρησιμοποιούνται για μετασχηματισμούς wavelet ικανοποιούν τις συνθήκες αυτές εξ' ορισμού των wavelets) και το 1996 δημοσίευσε μαζί με την Ingrid Daubechies το [C7] στο οποίο περιγράφουν έναν αλγόριθμο μετατροπής του παραπάνω γινομένου σε μια πιο αποδοτική μορφή. Συγκεκριμένα, έδειξαν ότι ο πίνακας

$$\begin{bmatrix} l_e(z) & l_o(z) \\ h_e(z) & h_o(z) \end{bmatrix}$$

μπορεί να παραγοντοποιηθεί σε γινόμενο πινάκων της μορφής:

$$\begin{bmatrix} l_e(z) & l_o(z) \\ h_e(z) & h_o(z) \end{bmatrix} = \left\{ \prod_i \begin{bmatrix} 1 & s_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i & 1 \end{bmatrix} \right\} \begin{bmatrix} K & 0 \\ 0 & -\frac{1}{K} \end{bmatrix} \quad (2.36)$$

Η μορφή αυτή λέγεται lifting scheme και είναι αποδοτική γιατί το μεν γινόμενο με τον τελευταίο διαγώνιο πίνακα είναι απλώς ένας πολλαπλασιασμός (κανονικοποίησης) επί κάποιον παράγοντα του εξομαλυμένου σήματος και των συντελεστών του wavelet, αντίστοιχα, οι δε διαγώνιοι πίνακες υλοποιούνται περνώντας το ένα εκ των δύο τμημάτων (βαθυπερατό ή υψιπερατό) μέσα από ένα μικρό φίλτρο, και προσθέτοντάς το στον εαυτό του και το άλλο τμήμα, πάντα με μια κατάλληλη χρονική καθυστέρηση:



Σχήμα 2.25 Γενική αρχιτεκτονική lifting scheme

Ειδικά για τον μετασχηματισμό που βασίζεται στο wavelet CDF 9/7, με το οποίο και ασχολούμαστε σε αυτήν τη διπλωματική εργασία, ισχύει (από τα [C3], C[4]):

$$s_1(z) = a(1 + z^{-1})$$

$$t_1(z) = b(1 + z^{-1})$$

$$s_2(z) = c(1 + z^{-1}), \text{ και,}$$

$$t_2(z) = d(1 + z^{-1})$$

όπου,

$$a = -1.586134342,$$

$$b = -0.05298011854,$$

$$c = 0.8829110762,$$

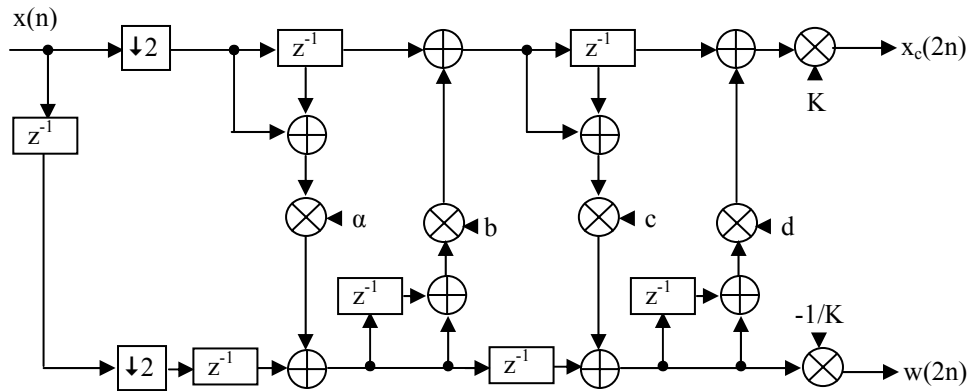
$$d = 0.4435068522,$$

$$K = 1.149604398, \text{ και,}$$

$$-1/K = -0.86986445227569$$

(1.35)

συνεπώς, ο lifting-based μετασχηματισμός του CDF 9/7 υλοποιείται ως εξής [C4]:



Σχήμα 2.26 Lifting υλοποίηση του CDF 9/7

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής:

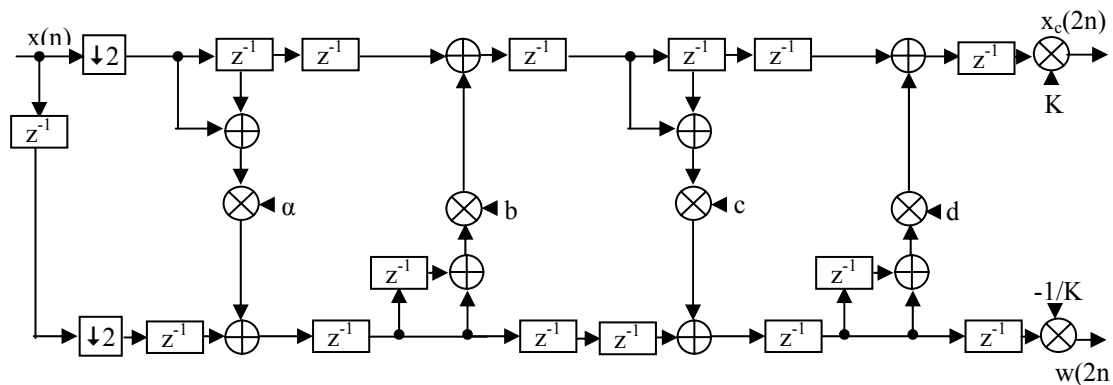
(2.36)

- Πλήθος στοιχείων: 7 καταχωρητές, 8 αθροιστές, 6 πολλαπλασιαστές, 2 υποδειγματολήπτες.
- Το κρίσιμο μονοπάτι ισούται με $8T_A + 5T_M$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται να ισούται με $6\frac{P_M}{2} + 6\frac{P_R}{2} + P_R + 8\frac{P_A}{2} = 3P_M + 4P_R + 4P_A$

Βλέπουμε, λοιπόν, ότι το lifting scheme επιτρέπει στο κύκλωμά μας να λειτουργεί στη μισή συχνότητα ρολογιού απ'ότι ο ρυθμός εισόδου των $x(n)$ και

επιπλέον υλοποιεί και το υψιπερατό και το βαθυπερατό τμήμα του μετασχηματισμού με μόλις 6 πολλαπλασιαστές και 8 προσθέτες: στον πίνακα 2.5 βλέπουμε ότι η direct half-rate αρχιτεκτονική χρειαζόταν 9 πολλαπλασιαστές και 14 προσθέτες. Συνεπώς, το νέο κύκλωμα έχει αρκετά μικρότερες απαιτήσεις σε υλικό και άρα σε ισχύ. Δυστυχώς, όμως, υστερεί σημαντικά στον τομέα των επιδόσεων: ενώ στην direct half-rate το κρίσιμο μονοπάτι ισούται με $3T_A+T_M$, στο παραπάνω σχήμα ισούται με $8T_A+5T_M$.

Μια ενδεχόμενη λύση θα ήταν να καταφεύγαμε στο pipelining, προσθέτοντας ένα ζευγάρι καταχωρητών στο τέλος κάθε σταδίου, όπως φαίνεται στο σχήμα 2.27, αυτό όμως θα προσέθετε άλλους 8 καταχωρητές στο σύστημα, γεγονός μη ανεκτό τόσο από την άποψη χώρου όσο από την άποψη κατανάλωσης ενέργειας.



Σχήμα 2.27 Lifting υλοποίηση του CDF 9/7 + pipelining

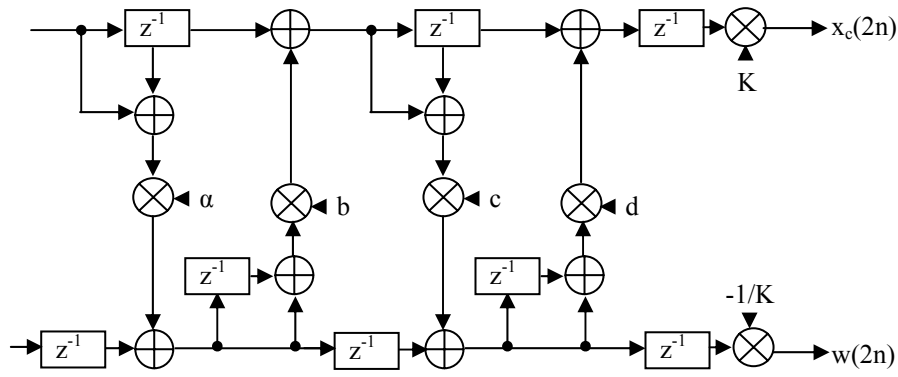
Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής

(2.37)

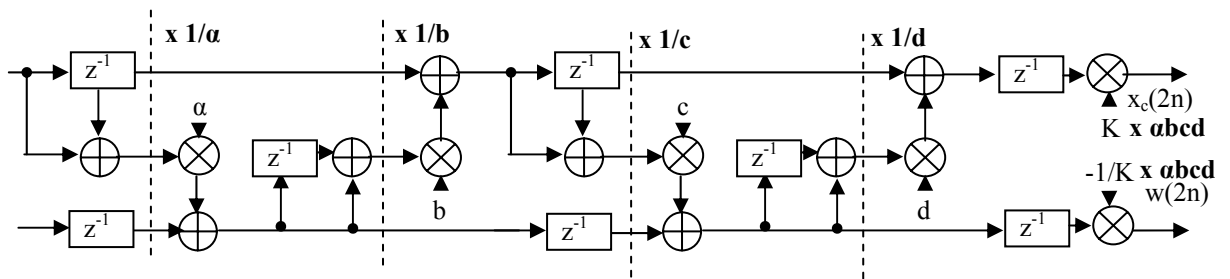
- Πλήθος στοιχείων: 15 καταχωρητές, 8 αθροιστές, 6 πολλαπλασιαστές, 2 υποδειγματολήπτες.
- Το κρίσιμο μονοπάτι ισούται με $2T_A + T_M$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται να ισούται με

$$6 \frac{P_M}{2} + 14 \frac{P_R}{2} + P_R + 8 \frac{P_A}{2} = 3P_M + 8P_R + 4P_A$$

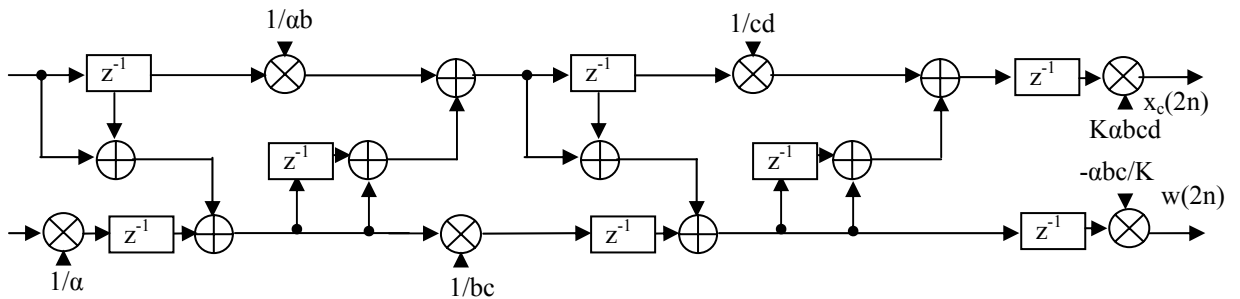
Τελικά, το 2004 στο [C6] οι Huang et al. βρήκαν έναν αποδοτικό τρόπο να μειώσουν το κρίσιμο μονοπάτι σε $5T_A+T_M$, κρατώντας το δεξιότερο ζευγάρι καταχωρητών στο σχ. 2.27 και αλλάζοντας τις θέσεις των πολλαπλασιαστών και των προσθετών (επιμερίζοντας τα κατάλληλα γινόμενα). Στο παρακάτω σχήμα θα δείξουμε πως η αρχιτεκτονική του σχ. 2.26 (χωρίς τον διαχωριστή άρτιων-περιττών στην αρχή, για λόγους ευκρίνειας) μετατρέπεται σε αυτή των Huang et. al.



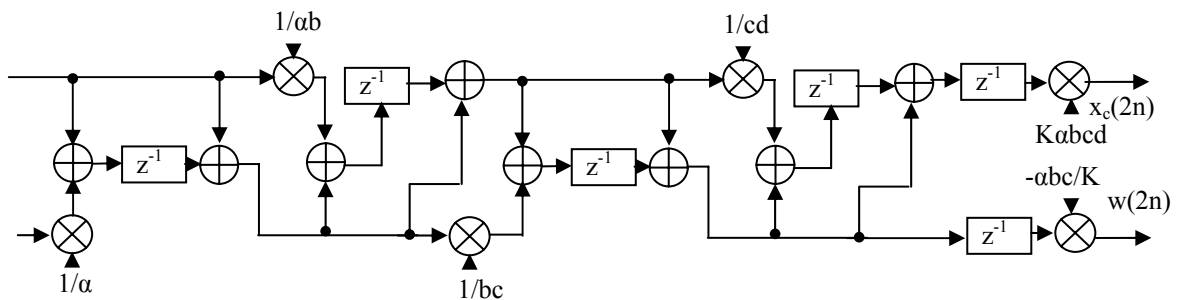
Σχήμα 2.28a Αρχικό Κύκλωμα



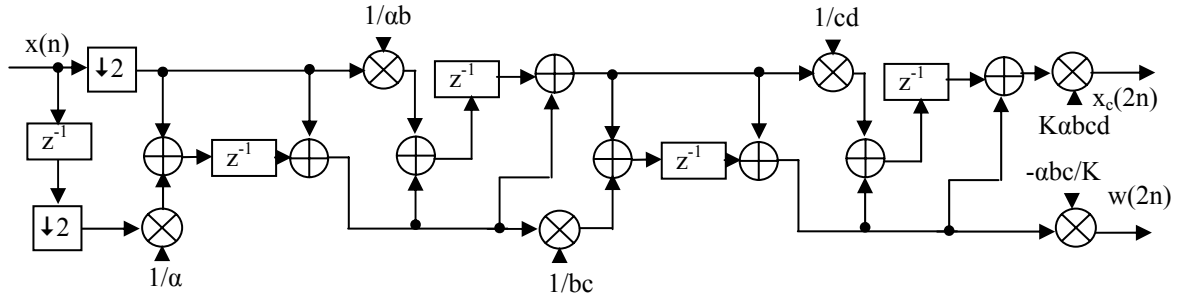
Σχήμα 2.28b Αναδιάταξη γράφου σηματορροής και καθορισμός «περιοχών διαίρεσης»



Σχήμα 2.28c Πολλαπλασιασμός με τους ανάστροφους συντελεστές, μετακίνηση καταχωρητών και συγχώνευση διαδοχικών πολλαπλασιαστών



Σχήμα 2.28d Αλλαγή σειράς προσθέσεων και μετακίνηση των καταχωρητών από τους προσθετέους στο αποτέλεσμα



Σχήμα 2.28e Η τελική βελτιωμένη lifting-based, flipping structure αρχιτεκτονική

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.38)

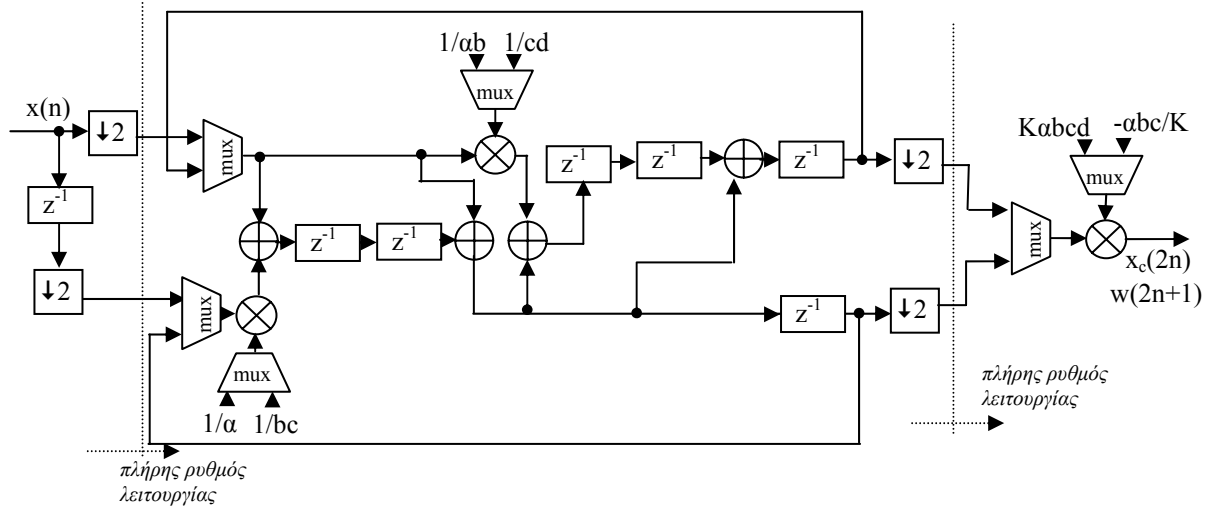
- Πλήθος στοιχείων: 5 καταχωρητές, 8 αθροιστές, 6 πολλαπλασιαστές, 2 υποδειγματολήπτες.
- Το κρίσιμο μονοπάτι ισούται με $3T_A + T_M$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται να ισούται με

$$6\frac{P_M}{2} + 6\frac{P_R}{2} + P_R + 8\frac{P_A}{2} = 3P_M + 4P_R + 4P_A$$

Όπως βλέπουμε και στα σχήματα 2.28a-2.28, η κύρια λογική πίσω από το flipping structure είναι η αφαίρεση των πολλαπλασιαστών από το κρίσιμο μονοπάτι (μέσω του πολλαπλασιασμού όλων των παράλληλων κλάδων με τον αντίστροφο του συντελεστή- εξ' ου και το όνομα "flipping structure". Στην πραγματικότητα, όπως αναφέρουν και στο [6], αφού οι πολλαπλασιασμοί-διαιρέσεις με το 2 είναι «δωρεάν» σε υλικό, μπορούμε να πολλαπλασιάσουμε με μια δύναμη του 2 επί τον αντίστροφο, ανάλογα με το τι μας συμφέρει από άποψη ακρίβειας πράξεων. Θα ασχοληθούμε με αυτό περισσότερο στο κεφάλαιο της λεπτομερέστερης υλοποίησης Έτσι, το κρίσιμο μονοπάτι μειώνεται σε $T_M + 5T_A$.

Στο 2.28d, πηγαίνουμε την αρχιτεκτονική των Huang et. al. ένα βήμα μπροστά με κατάλληλους επιμερισμούς των προσθέσεων και μετακινήσεις των καταχωρητών, και έτσι πετυχαίνουμε μείωση των καταχωρητών κατά δύο καθώς και μείωση του κρίσιμου μονοπατιού σε $T_M + 3T_A$, επίδοση καλύτερη ακόμα και από την PLSA στο [C7]. Συνεπώς, η αρχιτεκτονική του σχ. 2.28e είναι η αποδοτικότερη lifting-based και αξίζει να μελετηθεί περαιτέρω η λεπτομερής υλοποίησή της στο επόμενο κεφάλαιο.

Είναι προφανές ότι η αρχιτεκτονική του σχ. 2.28e αλλά και όλες οι lifting-based αρχιτεκτονικές ουσιαστικά αποτελούνται από επανάληψη των δύο πρώτων «βημάτων» τους (κάτι τέτοιο εξ' άλλου υποδηλώνεται και από το γινόμενο των ζευγών στη σχέση που ορίζει το lifting scheme). Συνεπώς, είναι δυνατό να εφαρμόσουμε τεχνικές folding στο σχ. 2.28e, έτσι ώστε (πάντα με κόστος τον διπλασιασμό ρυθμού λειτουργίας των πολλαπλασιαστών-προσθετών και με επιπλέον πολυπλέκτες και καταχωρητές) να επιτύχουμε μείωση του υλικού:



Σχήμα 2.29 Folding της αποδοτικότερης lifting αρχιτεκτονικής

Τα χαρακτηριστικά της παραπάνω αρχιτεκτονικής είναι τα εξής: (2.30)

- Πλήθος στοιχείων: 7 καταχωρητές, 4 αθροιστές, 3 πολλαπλασιαστές, 5 πολυπλέκτες, 4 υποδειγματολήπτες.
- Το κρίσιμο μονοπάτι ισούται με $2T_A + T_M + T_{mux}$
- Η συνολική κατανάλωση ισχύος του κυκλώματος αναμένεται να ισούται με $3P_M + 7P_R + 5P_{mux} + 4P_A$

Η αρχιτεκτονική του 2.29 είναι στον «πυρήνα» (δηλαδή, χωρίς τον πολλαπλασιασμό κανονικοποίησης στο τέλος) παρόμοια σε απαιτήσεις υλικού με την αντίστοιχη folding των Acharya et. al. στο [C4], μόνο που έχει κρίσιμο μονοπάτι $T_M + 2T_A$, αντί του $2T_M + 4T_A$ (αν αγνοήσουμε τις καθυστερήσεις που εισάγουν οι πολυπλέκτες 2-σε-1)

ΣΤ. Συμπεράσματα

Στο κεφάλαιο αυτό αναλύσαμε στο επίπεδο βασικών μονάδων μια σειρά από αρχιτεκτονικές που πραγματοποιούν τον ευθύ διακριτό μετασχηματισμό wavelet για το wavelet CDF 9/7. Τα κυκλώματα είχαν ως βάση τρία είδη αλγορίθμων: το απλό φιλτράρισμα από ένα ζεύγος (convolution), την παραγοντοποίηση b-splines, που στην πραγματικότητα αποτελεί μια βελτιωμένη μορφή του προηγούμενου όπου οι πολλαπλασιαστές (και η ευελιξία!) αντικαθίστανται από αθροιστές/αφαιρέτες, και τέλος το lifting scheme, όπου, εκμεταλλευόμενοι τις ειδικές τους ιδιότητες, ενοποιούμε το ζεύγος των φίλτρων σε ένα ενιαίο, μη διαχωρίσιμο δικτύωμα.

Τα χαρακτηριστικά των καλύτερων από κάθε κατηγορία αρχιτεκτονικών παρατίθενται στον ακόλουθο πίνακα, όπου με διπλό πλαίσιο επισημαίνονται οι αρχιτεκτονικές που θα μελετηθούν περαιτέρω (ως το επίπεδο πύλης) στο επόμενο κεφάλαιο:

Πίνακας 2.10 Σύγκριση αρχιτεκτονικών που πραγματοποιούν τον ευθύ μετασχηματισμό wavelet CDF 9/7

Αρχιτεκτονική	Στοιχεία	Κρίσιμο Μονοπάτι	Αναμεν. Ισχύς
Convolution , Direct Half-Rate	8 R + 14A + 9M + 2S	$T_M + 4T_A$	$4.5P_M + 4.5P_R + 7P_A$
Convolution , Direct Folded	10 R + 9A + 11 mux + 5M + 2 S	$T_{mux} + T_M + 4T_A$	$5P_M + 11P_{mux} + 9P_A + 10P_R$
Convolution , Transpose Half-Rate	13 R + 14 A + 9 M + 1 S	$T_M + 2T_A$	$4.5P_M + 7P_A + 7P_R$
Convolution , Transpose Folded	16 R + 9A + 11 mux + 5M + 2 S	$T_{mux} + T_M + T_A + \max(T_{mux}, T_A)$	$5P_M + 11P_{mux} + 9P_A + 16P_R$
B-Spline , Half-Rate καταναμημένο, Direct b-spline	10 R + 11 A + 1SB + 5 M + 4 S	$T_M + 5T_A$	$2.5P_M + 8P_A + 8P_R + P_{SB}$
B-Spline , Half-Rate καταναμημένο, Transpose b-spline	13 R + 13 A + 2 SB + 5 M + 4 S	$T_M + 3T_A$	$2.5P_M + 10P_A + 2P_{SB} + 11P_R$
B-Spline , Folded καταναμημένο, Direct b-spline	12 R + 10A + 1SB + 7 mux + 3 M + 2 S	$T_{mux} + T_M + 5T_A$	$3P_M + 7P_{mux} + 10P_A + 12P_R + P_{SB}$
B-Spline , Folded καταναμημένο, Transpose b-spline	15 R + 12A + 2SB + 7 mux + 3 M + 2 S	$T_{mux} + T_M + 3T_A$	$3P_M + 7P_{mux} + 12P_A + 15P_R + 2P_{SB}$
Lifting , βέλτιστο	75R + 8 A + 6 M + 2 S	$T_M + 3T_A$	$3P_M + 3P_R + 4P_A$
Lifting , βέλτιστο, folded	7R + 4 A + 3 M + 5 mux + 4 S	$T_M + T_{mux} + 2T_A$	$3P_M + 7P_R + 5P_{mux} + 4P_A$

Είναι αξιοσημείωτο το γεγονός ότι ταχύτερη αρχιτεκτονική απ' όλες παραμένει η «απλή» transpose half-rate που βασίζεται στη συνέλιξη. Βέβαια, οι επιδόσεις όλων στην ταχύτητα διαφέρουν μόνο κατά λίγους αθροιστές, γεγονός που δείχνει ότι οι απλούστεροι αλγόριθμοι (convolution-based) επιδέχονται περισσότερης διερεύνησης και δίνουν αποτελέσματα παραπλήσια των πιο προχωρημένων. Στο θέμα της ισχύος που αναμένουμε να καταναλώνει το κύκλωμα (καθώς σε τέτοιο επίπεδο όλα είναι εικασίες) φαίνεται οι lifting-based αρχιτεκτονικές να ανταγωνίζονται της b-spline-based για τον τίτλο της οικονομικότερης.

Ωστόσο, το επίπεδο βασικών μονάδων δεν είναι ικανό να μας δώσει σαφή αποτελέσματα για τις τελικές επιδόσεις του κυκλώματος, τόσο σε χώρο όσο και σε ταχύτητα/ισχύ. Για παράδειγμα, μπορεί μια αρχιτεκτονική που φαίνεται πιο «σπάταλη» να χρησιμοποιεί αποκλειστικά σταθερούς πολλαπλασιαστές, οπότε να είναι πολύ πιο οικονομική σε χώρο και ισχύ από μια άλλη «καλύτερη» στο επίπεδο βασικών μονάδων. **Για τον σκοπό αυτόν ακριβώς, δηλαδή την υλοποίηση σταθερών πολλαπλασιαστών, στο επόμενο κεφάλαιο θα μελετήσουμε ως το επίπεδο πύλης μόνο αρχιτεκτονικές με σταθερούς πολλαπλασιαστές.**

Επίσης, μπορεί το μήκος λέξης των ενδιάμεσων δεδομένων να είναι μεγαλύτερο σε μία περίπτωση απ'ότι σε μία άλλη, γεγονός που επίσης σημαίνει επιβάρυνση σε υλικό, ισχύ, και κρίσιμο μονοπάτι. Με αυτά τα ζητήματα ασχολείται το επόμενο κεφάλαιο, όπου οι τέσσερις επιλεγμένες αρχιτεκτονικές αυτού του κεφαλαίου αναλύονται ως το επίπεδο πύλης και bit.

Κεφάλαιο 3

Υλοποίηση σε επίπεδο πύλης

A. Εισαγωγή

Στο κεφάλαιο αυτό θα υλοποιήσουμε σε γλώσσα verilog ως το επίπεδο πύλης/D Flip-Flop τις βέλτιστες αρχιτεκτονικές του κεφαλαίου 2 με σταθερούς πολλαπλασιαστές (βλ. και πίνακα 2.10). Σκοπός μας είναι να αναδείξουμε ενδεχόμενα πλεονεκτήματα και μειονεκτήματα της κάθε μιας, τα οποία δεν είναι δυνατό να εντοπιστούν από την γενικόλογη ανάλυση σε αφαιρετικές βασικές μονάδες που έγινε πριν. Επιπλέον, καθώς επιλέγουμε κυκλώματα με σταθερούς πολλαπλασιαστές, θα χρειαστούμε για τον κάθε έναν από αυτούς σημαντικά λιγότερο κύκλωμα απ'ότι αν χρησιμοποιούσαμε γενικούς πολλαπλασιαστές. Από την άλλη όμως, ο όγκος του κυκλώματός τους εξαρτάται άμεσα από την τιμή του συντελεστή και τον τρόπο με τον οποίον αυτός κωδικοποιείται: έτσι, είναι πιθανό μια αρχιτεκτονική να είναι πιο οικονομική σε υλικό και πιο γρήγορη από τις υπόλοιπες χάρη (και μόνο) στις τιμές των σταθερών συντελεστών που χρησιμοποιεί.

Όπως θα αναφερθεί και παρακάτω, θεωρούμε ότι όλα τα σήματα που συναντώνται μέσα στα κυκλώματά μας είναι σε μορφή συμπληρώματος ως προς 2. Τόσο οι αθροιστές/αφαιρέτες όσο και οι εσωτερικοί αθροιστές/αφαιρέτες που σχηματίζουν τους σταθερούς πολλαπλασιαστές είναι διάδοσης κρατουμένου (carry-propagate). Παρά το ότι τα κυκλώματα διάδοσης κρατουμένου είναι λιγότερο γρήγορα από αυτά που χρησιμοποιούν άλλες τεχνικές (π.χ. carry-save) έχουν σχετικά μικρές απαιτήσεις σε υλικό και επιπλέον είναι ευκολότερη η συστηματική υλοποίησή τους, κάτι πολύ σημαντικό στην ανάπτυξη του λογισμικού που θα χρησιμοποιηθεί για την αυτοματοποιημένη κατασκευή των κυκλωμάτων (βλ. ενότητα 3Δ)

Η πορεία μας σε αυτό το κεφάλαιο θα είναι η εξής: αρχικά, θα κάνουμε μια στοιχειώδη εισαγωγή στα αριθμητικά συστήματα που θα χρησιμοποιήσουμε, καθώς και στις έννοιες που σχετίζονται με αυτά. Στη συνέχεια, θα αναφερθούμε στα βασικά αριθμητικά κυκλώματα που αποτελούν τις αρχιτεκτονικές με τις οποίες ασχολούμαστε, τα αξιώματα που τα διέπουν και τις παραδοχές/συμβάσεις που θα υιοθετήσουμε για τη συνέχεια της εργασίας. Τέλος, θα υλοποιήσουμε (με χρήση των αρχών που θα έχουμε αναδείξει προθύστερα) τα κυκλώματα που μας ενδιαφέρουν και θα συγκρίνουμε τα αποτελέσματα.

B. Αριθμητικά συστήματα

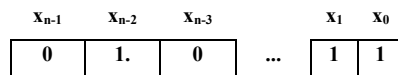
1. Εισαγωγή

Στην ενότητα αυτή θα αναφερθούμε στα αριθμητικά συστήματα που θα χρησιμοποιήσουμε στη συνέχεια, καθώς και αρχές και έννοιες συνυφασμένες με αυτά. Σκοπός παράθεσης αυτών των εννοιών είναι η όσο το δυνατόν αυτοτέλεια του κειμένου της διπλωματικής. Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στα [E6], [E10] και [E13] για θέματα αριθμητικών συστημάτων και κυκλωμάτων, γενικότερα, όπως και στο [E12] για θέματα κλιμάκωσης και μήκους λέξης.

2. Μορφή συμπληρώματος ως προς 2

Ένα σήμα (που, για παράδειγμα, υφίσταται διακριτό μετασχηματισμό wavelet προκειμένου να διαχωρίσουμε τις λεπτομέρειες από τη βασική δομή του) μπορεί να λάβει αρνητικές ή θετικές τιμές. Για το σκοπό αυτό, είναι απαραίτητο όλα τα εξωτερικά (είσοδοι-έξοδοι) και εσωτερικά σήματα των κυκλωμάτων μας να είναι σε μορφή που να επιτρέπει την αναπαράσταση αρνητικών αριθμών. Επιπρόσθετα, όπως είδαμε και στο προηγούμενο κεφάλαιο, και οι συντελεστές των (σταθερών) πολλαπλασιαστών λαμβάνουν αρνητικές τιμές, επομένως και αυτοί χρειάζεται να είναι σε μορφή που να επιτρέπει την αναπαράσταση τέτοιων τιμών.

Για τους σκοπούς αυτούς, λοιπόν, αποφασίζουμε όλα τα ψηφιακά σήματα εντός των κυκλωμάτων μας να είναι σε μορφή συμπληρώματος ως προς 2. Έστω μια αναπαράσταση συμπληρώματος ως προς 2 μήκους n bits. Τότε, κάθε ακέραιος αριθμός που μπορεί να αναπαρασταθεί με αυτόν τον τρόπο έχει τη μορφή του παρακάτω σχήματος:



Σχήμα 3.1 Μορφή συμπληρώματος ως προς 2

Η (δεκαδική) τιμή κάθε αριθμού που μπορεί να αναπαρασταθεί όπως στο σχ. 3.1 δίδεται από τη σχέση:

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \quad (3.1)$$

όπου X η δεκαδική τιμή του και x_i ($i \in [0, n-1], x_i \in \{0,1\}$) τα ψηφία της δυαδικής αναπαράστασης. Η μεταβλητή n λέγεται **μήκος λέξης** της δυαδικής αναπαράστασης, και είναι φανερό ότι με την (3.1) μπορούν να αναπαρασταθούν όλοι οι ακέραιοι μεταξύ $[-2^{n-1}, 2^{n-1}-1]$. Το πιο σημαντικό bit (δηλαδή, αυτό που έχει αξία 2^{n-1}) καθορίζει και το πρόσημο του αριθμού, γι' αυτό και συμβατικά ονομάζεται και **bit πρόσημου**.

Στο εξής, οι έννοιες «δυαδική αναπαράσταση αριθμού» και «αριθμός» θα χρησιμοποιούνται ως συνώνυμες. Η αναπαράσταση που ορίζει τη σχέση (3.1) είναι *μοναδική* για κάθε ακέραιο αριθμό, δηλαδή, σε κάθε ακέραιο αντιστοιχεί μονοσήμαντα μια ακολουθία από x_i που τον αναπαριστά στο δυαδικό σύστημα σε μορφή συμπληρώματος ως προς 2, με μήκος λέξης n bits.

3. Κανονική αναπαράσταση προσημασμένου ψηφίου (csd)

Θα προσπαθήσουμε εδώ να γενικεύσουμε τη σχέση (3.1). Εκεί, δείξαμε ότι κάθε προσημασμένος δεκαδικός αριθμός μπορεί να αναπαρασταθεί ως μια ακολουθία bit από τα οποία το μεγαλύτερης (απόλυτης) αξίας έχει αρνητική τιμή. Μια γενικότερη αναπαράσταση n -ψηφίων θα ήταν η παρακάτω:

$$X = \sum_{i=0}^{n-1} x_i 2^i \quad (3.2)$$

όπου, X η δεκαδική τιμή του αριθμού και $x_i \in \{-1, 1, 0\}$. Με τον τρόπο αυτό μπορούμε να αναπαραστήσουμε (κατά μη μοναδικό τρόπο, όμως) κάθε ακέραιο μεταξύ $[-2^{n+1}, 2^n - 1]$. Είναι φανερό ότι η σχέση (3.1) προκύπτει από τη σχέση (3.2) «περιορίζοντας» τις τιμές που μπορούν να πάρουν τα x_i . Γενικότερα, όλα τα αριθμητικά συστήματα με βάση το 2 προκύπτουν από τη σχέση (3.2) με περιορισμούς του βαθμού ελευθερίας των x_i . Όσα από αυτά επιτρέπουν περισσότερες από μια αναπαραστάσεις για ένα συγκεκριμένο αριθμό ονομάζονται *redundant* αριθμητικά συστήματα. Μια πολύ καλή και λεπτομερέστερη εισαγωγή σε αυτά γίνεται στο [E6].

Θα αναρωτηθεί κανείς, βέβαια, ποιος ο λόγος να χρησιμοποιήσει κανείς μια αναπαράσταση σαν την (3.2) σε ψηφιακά κυκλώματα, από τη στιγμή που αυτά είναι δισταθή, δηλαδή μπορούν να λάβουν μόνο δύο τιμές για κάθε x_i . Πράγματι, όλα τα σήματα στα κυκλώματά μας θα είναι σε μορφή συμπληρώματος ως προς 2. Ωστόσο, βολεύει μερικές ποσότητες (και συγκεκριμένα, τους συντελεστές των σταθερών πολλαπλασιαστών) να τις έχουμε σε εναλλακτικές μορφές. Οι λόγοι γι' αυτό θα αναλυθούν παρακάτω στους πολλαπλασιαστές.

Μια τέτοια εναλλακτική μορφή με ιδιαίτερο ενδιαφέρον είναι αυτή που μας διασφαλίζει ότι η αναπαράσταση ενός αριθμού σύμφωνα με την (3.2) περιέχει όσο το δυνατόν περισσότερα μηδενικά x_i . Ένα τέτοιο διάνυσμα από x_i , δηλαδή, αυτό που απ' όλα τα διανύσματα $\{x_i\}$ που ικανοποιούν την (3.2) για κάποιον αριθμό X έχει τα λιγότερα μη-μηδενικά ψηφία, λέγεται **κανονικό**, η δε αναπαράσταση του αριθμού υπό αυτό το διάνυσμα λέγεται **κανονική αναπαράσταση προσημασμένου αριθμού** (canonical signed digit representation – CSDR). Στην εργασία αυτή θα χρησιμοποιούμε ισοδύναμα και τον όρο «csd μορφή».

Είναι πιθανό να υπάρχουν περισσότερες από μια csd μορφές για έναν αριθμό. Ωστόσο, ο Reitweisner απέδειξε ότι υπάρχει, για όλους τους αριθμούς, csd μορφή στην οποία το γινόμενο των δύο σημαντικότερης αξίας x_i είναι διαφορετικό του 1, ενώ επιπλέον παρουσίασε έναν αλγόριθμο με τον οποίον μετατρέπεται, με συστηματικό τρόπο, οποιοσδήποτε αριθμός από μορφή συμπληρώματος ως προς 2 σε μορφή csd. Ο αλγόριθμος αυτός ονομάζεται αλγόριθμος του Reitweisner και ο ενδιαφερόμενος αναγνώστης μπορεί να τον βρει στο [E6]. Εδώ δεν τον παραθέτουμε αν και στα πλαίσια της εργασίας αναπτύχθηκε σε γλώσσα C για την κατασκευή του προγράμματος sfc (βλ. ενότητα 3Δ, Υλοποίηση σε επίπεδο πύλης). Στο εξής, όταν θα αναφέρεται «CSD μορφή του αριθμού X » θα εννοείται αποκλειστικά το κανονικό διάνυσμα που προκύπτει από τον αλγόριθμο του Reitweisner. Εν γένει, αποδεικνύεται (στο E13] ότι η CSD μορφή ενός αριθμού X περιέχει κατά μέσο όρο $n/3$ μη μηδενικά ψηφία.

4. Μη ακέραιοι αριθμοί: κλιμάκωση και μήκος λέξης

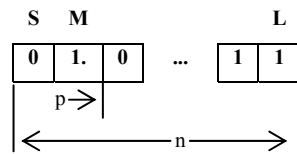
Ως τώρα αναφερθήκαμε αποκλειστικά σε ακέραιους αριθμούς. Ωστόσο, είναι φανερό ότι μπορούμε να προεκτείνουμε τη σχ. (3.1) ή τη σχέση (3.2) για να περιλαμβάνουν και ρητούς αριθμούς. Αν και στο εξής θα αναφερθούμε μόνο σε μορφές συμπληρώματος ως προς 2, τα ίδια ακριβώς συμπεράσματα ισχύουν και για τη csd μορφή, αφού ο αλγόριθμος του Reitweisner μας δίνει το κανονικό διάνυσμα με μήκος (το πολύ) όσο η αναπαράσταση ως προς 2.

Επεκτείνοντας την (3.1), μπορούμε να αναπαραστήσουμε *προσεγγιστικά* κάθε ρητό αριθμό X , με ακρίβεια ως 2^{-k} , σε μορφή συμπληρώματος ως προς 2 και μήκος λέξης $n+k$ όπως ορίζεται από τη σχέση:

$$X \approx -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i + \sum_{j=-1}^{-k} x_j 2^{-j} \quad (3.3)$$

Η (3.3) ισοδυναμεί ουσιαστικά με πολλαπλασιασμό του X με 2^k , αποκοπή του μη ακέραιου μέρους του γινομένου και αναπαράστασή του σε μορφή συμπληρώματος ως προς 2 με μήκος λέξης n+k.

Στο [E12] παρουσιάζεται κομψότερα η έννοια αυτή. Εποπτικά, έστω ότι θέλουμε να αναπαραστήσουμε έναν ρητό αριθμό X σε μορφή συμπληρώματος ως προς 2, με μήκος λέξης n. Έτσι, θέτουμε ότι κάποια από τα λιγότερο σημαντικά bits έχουν αξία μικρότερη της μονάδας (στο εξής, θα τα ονομάζουμε *καταχρηστικά «δεκαδικά»*), δηλαδή θεωρούμε ότι αυτά βρίσκονται δεξιά της υποδιαστολής. Τότε, ο αριθμός θα αναπαρίσταται όπως στο παρακάτω σχήμα:



Σχήμα 3.2 Δυαδική αναπαράσταση προσημασμένου ρητού αριθμού σε μορφή συμπληρώματος ως προς 2

Δύο είναι οι παράμετροι που ορίζουν μονοσήμαντα μια τέτοια αναπαράσταση: αφενός μεν το μήκος λέξης n, αφετέρου δε η θέση της υποδιαστολής. Στο [E12] η θέση της υποδιαστολής δίδεται από το p, που ορίζεται ως «πόσες θέσεις δεξιότερα του bit προσήμου βρίσκεται η υποδιαστολή» (σημειώστε εδώ ότι στο [E12] το n δεν περιλαμβάνει το bit προσήμου, σε αντίθεση με αυτή την εργασία).

Ένας ισοδύναμος (και πιο «επίσημος») ορισμός για το p που θα χρησιμοποιήσουμε εδώ είναι ότι **το p είναι τέτοιο έτσι ώστε η απόλυτη αξία του MSB της αναπαράστασης ισούται με 2^p** . Έστω ότι έχουμε κάποια μεταβλητή, η οποία λαμβάνει τη μέγιστη απόλυτη τιμή k (k ρητός) και θέλουμε να βρούμε εκείνο το p που να μας εξασφαλίζει την αποδοτικότερη αναπαράσταση, δηλαδή, αυτή με τα λιγότερα ίδια με το bit προσήμου ψηφία, αμέσως δεξιά από αυτό. Τότε, θα πρέπει αναγκαστικά να ισχύει, από την (3.1):

$$2^{p-1} \leq k < 2^p \Rightarrow p-1 \leq \log_2 k < p \Rightarrow p \leq \log_2 k + 1 < p+1 \Rightarrow p = \lfloor \log_2 k + 1 \rfloor \Rightarrow p = \lfloor \log_2 k \rfloor + 1 \quad (3.4)$$

Συνεπώς, η (3.4) μας δίνει τον τρόπο να βρούμε το κατάλληλο p έτσι ώστε να έχουμε τη βέλτιστη αναπαράσταση για κάποιο σύνολο ρητών αριθμών. Παρατηρήστε ότι το p μπορεί να λάβει αρνητικές τιμές, ή τιμές μεγαλύτερες από το n, το οποίο σημαίνει ότι ασχολούμαστε με πολύ μικρούς ή πολύ μεγάλους αριθμούς, αντίστοιχα.

Η ποσότητα p ονομάζεται *scaling* στο [E12]. Εμείς εδώ θα χρησιμοποιήσουμε τον όρο **κλιμάκωση**. Όπως προαναφέραμε, κάθε (προσεγγιστική) αναπαράσταση ρητού αριθμού σε μορφή συμπληρώματος ως προς 2 ορίζεται μονοσήμαντα από το ζευγάρι (n,p). Είναι φανερό από το σχ. 3.2 ότι για μια τέτοια αναπαράσταση ισχύουν:

- Αριθμός «ακεραίων» bits, δηλαδή bit με απόλυτη αξία μεγαλύτερη ή ίση της μονάδας i: $i = p+1$ (3.5)

Αν $i < 1$, τότε προφανώς δεν υπάρχουν τέτοια bits

- Ακρίβεια δεκαδικής αναπαράστασης, δηλαδή η μικρότερη δύναμη του 2 που αναπαριστά η μορφή αυτή: $d=n-p-1$ (3.6)
Αυτό σημαίνει ότι μπορούν να αναπαρασταθούν δεκαδικοί αριθμοί με ακρίβεια ως το 2^{-d} . Για την περίπτωση που $i < 1$, τότε το d μπορεί να γίνει ίσο ή μεγαλύτερο από το μήκος λέξης n .
- Αριθμός «δεκαδικών» bits, δηλαδή bit με απόλυτη αξία μεταξύ (0, 1):
$$f = \max(n, d) \quad (3.7)$$
αφού ο αριθμός τους δεν μπορεί να υπερβεί το μήκος λέξης.

Παραδοσιακά, σε κυκλώματα που δεν έχουν μονάδες κινητής υποδιαστολής, όπως αυτά που μελετούμε σε αυτή τη διπλωματική εργασία, ορίζονται εξ' αρχής οι ίδιες παράμετροι (n, p) για όλα τα σήματα εντός του κυκλώματος και τα διάφορα υπολογιστικά στοιχεία (αθροιστές, πολλαπλασιαστές) είναι σχεδιασμένα να διατηρούν αυτή τη σύμβαση. Αυτό είναι λογικό, αν αναλογιστεί κανείς ότι π.χ. σε έναν ψηφιακό επεξεργαστή σήματος το μήκος λέξης των δεδομένων είναι σταθερό, και συνεπώς, η πρόσθεση δύο αριθμών με διαφορετική κλιμάκωση ή θα έδινε λάθος αποτέλεσμα, ή θα έπρεπε ταυτόχρονα να υπάρχει και η πληροφορία της κλιμάκωσης του σήματος που βρίσκεται στο διάδρομο, κάτι που απαιτεί πρόσθετο κυκλωματικό κόστος και σίγουρα ανήκει στο πεδίο της κινητής υποδιαστολής.

Ωστόσο, σε εξειδικευμένα κυκλώματα που σχεδιάζονται από την αρχή για έναν αριθμητικό υπολογισμό, όπως οι αρχιτεκτονικές μας που υλοποιούν το διακριτό ευθύ μετασχηματισμό wavelet, υπάρχει απόλυτη ελευθερία στον προκαθορισμό του μήκους λέξης των ενδιάμεσων σημάτων, καθώς και της κλιμάκωσής τους. Έτσι είναι δυνατό να μεταβάλλονται, ανάλογα με τις ανάγκες, το μήκος λέξης και η κλιμάκωση από εξάρτημα σε εξάρτημα (τα οποία, ωστόσο, παραμένουν σταθερά στο χρόνο) έτσι ώστε και μεγαλύτερη ακρίβεια να πετυχαίνουμε, και καλύτερη αξιοποίηση του υλικού. Αυτό ακριβώς θα εφαρμοστεί παρακάτω, όπου και θα παρουσιάσουμε στην ενότητα των βασικών βασικών μονάδων, πως καθορίζονται οι τιμές του (n, p) της εξόδου μιας βασικής μονάδας συναρτήσεως των εισόδων της.

Αν και η λογική της μεταβλητής κλιμάκωσης αφορά υπολογισμούς κινητής υποδιαστολής, εδώ όλα τα κυκλώματα εκτελούν πράξεις σταθερής υποδιαστολής: οι πληροφορίες του ζεύγους (n, p) χρησιμοποιούνται μόνο κατά τη δημιουργία του κυκλώματος και δεν εμφανίζονται πουθενά ως σήματα στην τελική υλοποίηση. Οι συνδέσεις μεταξύ των διαφόρων βασικών μονάδων περιέχουν μόνο τη δυαδική μορφή του αριθμού σε συμπλήρωμα ως προς 2 και πουθενά δε δίνεται η κλιμάκωσή του. Έτσι, μπορούμε να πούμε ότι οι υπολογισμοί μας είναι *ψευδο-κινητής υποδιαστολής*, εφ' όσον υλοποιούνται με στοιχεία σταθερής υποδιαστολής αλλά αξιοποιούν τα πλεονεκτήματα των αρχιτεκτονικών κινητής υποδιαστολής.

Γ. Αρχιτεκτονικές βασικών μονάδων

1. Εισαγωγή

Αφού αναφερθήκαμε πριν στις απαιτούμενες γνώσεις αριθμητικών αναπαραστάσεων και συστημάτων, είμαστε πλέον σε θέση να μελετήσουμε τα βασικά εξαρτήματα από τα οποία θα δημιουργήσουμε τα κυκλώματά μας, και τα οποία συμβολίζαμε ως «μαύρα κουτιά» στο προηγούμενο κεφάλαιο.

Στη συνέχεια αυτής της εργασίας θα ασχοληθούμε μόνον με τους αθροιστές/αφαιρέτες, τους πολλαπλασιαστές (οι ολισθήσεις περιλαμβάνονται σε

αυτούς) και τους καταχωρητές. Δε θα ασχοληθούμε με τους διαχωριστές άρτιων-περιττών, καθώς η πολύ συγκεκριμένη λειτουργία τους δεν δίνει ευκαιρίες για περαιτέρω μελέτη ή βελτιώσεις. Έτσι, θεωρούμε στο εξής ότι τα δικτυώματα των κυκλωμάτων μας αποτελούνται αποκλειστικά από τα τέσσερα αυτά είδη βασικών μονάδων, τα οποία και καθορίζουν τις τελικές επιδόσεις σε ισχύ, μέγεθος και ταχύτητα.

Όπως αναφέρθηκε και στην εισαγωγή, όλα τα εξαρτήματα θα είναι παράλληλα και διάδοσης κρατουμένου (carry-propagate). Αρχικά θα αναφερθούμε στους πλήρεις αθροιστές και στους αθροιστές/αφαιρέτες αριθμών σε μορφή συμπληρώματος ως προς 2. Οι γνώσεις αυτές είναι μάλλον βασικές και παρατίθενται για λόγους αυτάρκειας του κειμένου. Στη συνέχεια, θα δούμε πως η ύπαρξη σημάτων εισόδου με διαφορετικά μήκος λέξης και κλιμάκωση επηρεάζει τις επιδόσεις των στοιχείων αυτών, καθώς και πως προκύπτουν η κλιμάκωση και το μήκος λέξης των εξόδων τους, λόγω των συμβάσεων που θα γίνουν. Κατόπιν, θα γίνει μελέτη των σταθερών πολλαπλασιαστών που βασίζονται στη χρήση αθροιστών σταθερού μήκους, οι οποίοι και θα χρησιμοποιηθούν στη συνέχεια. Τέλος, θα αναφέρουμε επιγραμματικά τον ρόλο των καταχωρητών (στοιχεία καθυστέρησης) και τα κυριότερα χαρακτηριστικά τους.

2. Αθροιστές-Αφαιρέτες

Τα εξαρτήματα αυτά έχουν αρκετές ομοιότητες, καθώς είναι τα μόνα τα οποία «συγχωνεύουν» δύο σήματα. Επιπλέον, αποτελούν βασική δομική μονάδα των πολλαπλασιαστών που θα εισάγουμε παρακάτω. Για το σκοπό αυτό θα μελετηθούν μαζί, αναφέροντας τις διαφορές τους όπου αυτές υπάρχουν

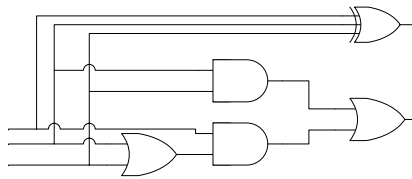
α. Πρόσθεση σε επίπεδο bit: πλήρεις αθροιστές

Βασική δομική μονάδα όλων των αθροιστών/αφαιρέτων, και κατ' επέκταση όλων των βασικών μονάδων που εκτελούν πράξεις, είναι ο πλήρης αθροιστής (full adder). Το στοιχείο αυτό δέχεται, στη βασική του μορφή, ως εισόδους 3 bits (a , b , c_{in}) και τα αθροίζει, δίνοντας ως έξοδο ένα bit αθροίσματος (s) και ένα bit κρατουμένου (c_{out}). Σε ορολογία σύμφωνη με αυτή των αριθμητικών συστημάτων, τα bits εισόδου όπως και το bit αθροίσματος έχουν την ίδια αξία, ενώ η αξία του bit κρατουμένου είναι μεγαλύτερη κατά 1. Όπως προκύπτει πολύ εύκολα, ο λογικός πίνακας ενός πλήρη αθροιστή με τις τρεις εισόδους του θετικής αξίας (δηλαδή, 0 αν η είσοδος είναι 0, 1 αν είναι -1) έχει ως εξής:

Πίνακας 3.1 Πίνακας αληθείας πλήρη αθροιστή

a (+1)	b (+1)	c_{in} (+1)	s (+1)	c_{out} (+2)
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Παρά το ότι ο παραπάνω πίνακας μπορεί άμεσα να μας δώσει τα s και c_{out} ως αθροίσματα ελαχιστόρων, συνήθως ο πλήρης αθροιστής αναλύεται σε πύλες σύμφωνα με το ακόλουθο σχήμα.



Σχήμα 3.3 Εσωτερική δομή πλήρη αθροιστή

Ο κώδικας verilog που υλοποιεί το σχ. 3.3 είναι ο εξής:

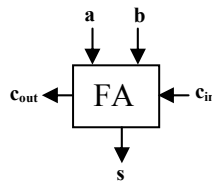
```

module fulladder(c_out, s, a, b, c_in);
  output c_out, s;
  input a, b, c_in;
  wire and_ab, or_ab, and_2;
  xor(s, a, b, c_in);
  or(or_ab, a, b);
  and(and_ab, a, b);
  and(and_2, or_ab, c_in);
  or(c_out, and_2, and_ab);
endmodule

```

c_{in}

Κατά τα γνωστά, οι εισοδοί a , b ονομάζονται «είσοδοι» του αθροιστή, ενώ η c_{in} «κρατούμενο εισόδου». Στα εξαρτήματα που τον χρησιμοποιούν, ο πλήρης αθροιστής εμφανίζεται με το εξής σύμβολο:



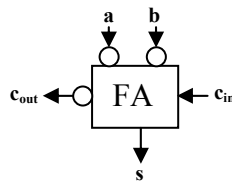
Σχήμα 3.4 Κυκλωματικό σύμβολο πλήρη αθροιστή με θετικές εισόδους

Όπως θα φανεί παρακάτω, πολλές φορές θα μας ζητηθεί να αθροίσουμε τρία bits, από τα οποία τα δύο (έστω το a και το b , αν και τα αποτελέσματα είναι πολύ παρόμοια σε περίπτωση που επιλέξουμε άλλο ζευγάρι) έχουν αρνητική αξία, δηλαδή, αν είναι 0 αναπαριστούν τιμή 0, ενώ αν είναι 1 αναπαριστούν τιμή -1. Τότε, η έξοδος θα κυμαίνεται μεταξύ -2 και 1, άρα, υποχρεωτικά, θα πρέπει να οδηγηθούμε σε μια δομή παρόμοια με του πλήρη αθροιστή όπου όμως το κρατούμενο εξόδου έχει αρνητική τιμή, καθώς μόνο αυτό έχει απόλυτη αξία 2. Ο πίνακας αληθείας αυτού του κυκλώματος θα είναι ο εξής:

Πίνακας 3.2 Πίνακας αληθείας αθροιστή δύο αρνητικών και ενός θετικού bit

a (-1)	b (-1)	c_{in} (+1)	s (+1)	c_{out} (-2)
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	0	0	1
0	0	1	1	0
1	0	1	0	0
0	1	1	1	0
1	1	1	1	1

Όπως μπορούμε να διαπιστώσουμε, και αποδεικνύεται και στο [Ε6], το κύκλωμα που υλοποιεί τον παραπάνω πίνακα είναι ίδιο με αυτό του πλήρη αθροιστή, όπου όμως στις εισόδους/εξόδους που έχουν αρνητική αξία έχουμε μια πύλη NOT. Συνεπώς, το κύκλωμα έχει την παρακάτω μορφή:



Σχήμα 3.5 Κυκλωματικό σύμβολο πλήρη αθροιστή με δύο αρνητικές εισόδους

Στη συνέχεια θα δούμε την άμεση εφαρμογή αυτών των κυκλωμάτων.

β. Αθροιστής συμπληρώματος ως προς 2

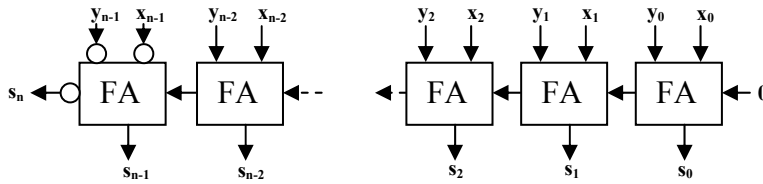
Έστω ότι θέλω να αθροίσω δύο αριθμούς σε μορφή συμπληρώματος ως προς 2, $\{x_i\}$ και $\{y_i\}$. Προς το παρόν θα θεωρήσουμε ότι είναι ακέραιοι και ότι έχουν το ίδιο μήκος λέξης n : η παραδοχή αυτή θα συνεχιστεί και στον αφαιρέτη παρακάτω. Τότε, σύμφωνα με την (3.1), η τιμή του καθενός ισούται με:

$$X = -2^{n-1}x_{n-1} + \sum_{i=0}^{n-2} x_i 2^i \quad \text{και} \quad Y = -2^{n-1}y_{n-1} + \sum_{i=0}^{n-2} y_i 2^i, \quad \text{άρα, θα ισχύει για το}$$

άθροισμά τους:

$$S = X + Y = -2^{n-1}(x_{n-1} + y_{n-1}) + \sum_{i=0}^{n-2} (x_i + y_i) 2^i \quad (3.8)$$

Εύκολα μπορούμε να διαπιστώσουμε ότι το άθροισμά των θετικής αξίας $\{x_i, y_i\}$ μπορεί να προκύψει από μια σειρά πλήρων αθροιστών του σχ. 3.4, το δε άθροισμα των αρνητικής αξίας bit από το κύκλωμα του σχ. 3.5. Άρα, έχω για τον παράλληλο αθροιστή συμπληρώματος ως προς δύο την παρακάτω μορφή:



Σχήμα 3.6 Παράλληλος αθροιστής αριθμών σε μορφή συμπληρώματος ως προς 2

Διαπιστώνουμε ότι ο παράλληλος αθροιστής απαιτεί n πλήρεις αθροιστές και τρεις πύλες NOT. Ο κώδικας verilog που υλοποιεί το παραπάνω σχήμα ως παραμετροποιήσιμο αθροιστή μεταβλητού μήκους είναι ο εξής:

```

module add2c_cp(out, a, b);
  //out=a+b
  parameter inp_size=8;
  output [inp_size:0] out;
  input [inp_size-1:0] a, b;
  wire [inp_size-1:0] c_wire;
  wire [2:0] n_wire;
  not n_i[2:0] ({out[inp_size], n_wire[1:0]},
              {n_wire[2], b[inp_size-1], a[inp_size-1]});
  fulladder f_a[inp_size-1:0] ({n_wire[2], c_wire[inp_size-1:1]},
                              out[inp_size-1:0], {n_wire[0], a[inp_size-2:0]},
                              {n_wire[1], b[inp_size-2:0]}, c_wire[inp_size-1:0]);
  assign c_wire[0]=1'b0;
endmodule

```

γ. Αφαιρέτης συμπληρώματος ως προς 2

Όμοια με πριν, έστω ότι θέλω να αφαιρέσω τον X από τον Y . Ισχύει, όπως και πριν:

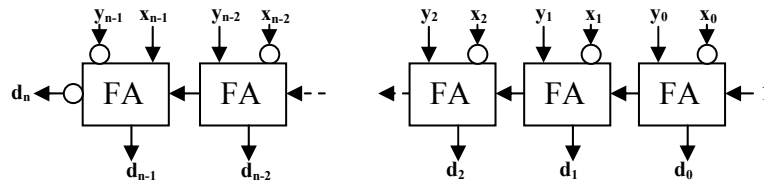
$$D = Y - X = -2^{n-1}(-x_{n-1} + y_{n-1}) + \sum_{i=0}^{n-2} (-x_i + y_i)2^i \quad (3.9)$$

Η (3.9) μπορεί επίσης να γραφεί ως:

$$D = 2^{n-1}(x_{n-1} - y_{n-1}) + \sum_{i=0}^{n-2} (y_i - x_i)2^i - 0 = 2^{n-1}(x_{n-1} - y_{n-1}) + \sum_{i=1}^{n-2} (y_i - x_i)2^i + (y_0 - x_0 - 0)$$

Το τελευταίο μέρος της τελικής μορφής μπορεί να υλοποιηθεί άμεσα με ένα κύκλωμα πλήρη αθροιστή με δύο αρνητικές εισόδους, όπου όμως εδώ αρνητικές εισοδοί θα είναι το κρατούμενο εισόδου και η είσοδος x_0 . Επειδή το κρατούμενο εξόδου προς την επόμενη βαθμίδα θα έχει αρνητική αξία, και τα επόμενα $n-2$ σημαντικότερα bits μπορούν να αθροιστούν με παρόμοιο τρόπο. Τέλος, επειδή το κρατούμενο εξόδου της πρόσθεσης των δεύτερων πιο σημαντικών bit θα είναι

αρνητικό, και τα δύο σημαντικότερα bit μπορούν να προστεθούν με τον ίδιο τρόπο, με αρνητικές εισόδους το κρατούμενο εισόδου και το y_{n-1} . Επιπλέον, παρατηρούμε ότι οι διαδοχικές πύλες NOT στα ενδιάμεσα κρατούμενα αλληλοαναιρούνται, και, τέλος, μπορούμε να αντικαταστήσουμε την πύλη NOT με είσοδο πάντα 0 στην αρχή της αλυσίδας με μια είσοδο 1. Συνεπώς, ο παράλληλος αφαιρέτης δύο αριθμών σε συμπλήρωμα ως προς 2 έχει την παρακάτω μορφή:



Σχήμα 3.7 Παράλληλος αφαιρέτης αριθμών σε μορφή συμπληρώματος ως προς 2 (Y-X)

Διαπιστώνουμε ότι και το αποτέλεσμα είναι σε μορφή συμπληρώματος ως προς 2, συνεπώς μπορούμε ελεύθερα να συνδέουμε μεταξύ τους τις εισόδους και τις εξόδους διαδοχικών αθροιστών και αφαιρέτων. Το κόστος της βασικής μονάδας σε υλικό είναι n πλήρεις αθροιστές και $n+1$ πύλες NOT. Ο κώδικας verillog που το υλοποιεί είναι ο ακόλουθος:

```

module sub2c_cp(out, a, b);
  //o a αφαιρείται από τον b (out=b-a)
  //ckdok, 8/10/2007@10:00
  parameter inp_size=8;
  output [inp_size:0] out;
  input [inp_size-1:0] a, b;
  wire [inp_size-1:0] c_wire;
  wire [inp_size:0] n_wire;
  not n_i[inp_size:0] ({out[inp_size],n_wire[inp_size-1:0]}},
    {n_wire[inp_size],b[inp_size-1],a[inp_size-2:0]});
  fulladder f_a[inp_size-1:0] ({n_wire[inp_size],c_wire[inp_size-1:1]}},
    out[inp_size-1:0], {a[inp_size-1], n_wire[inp_size-2:0]}},
    {n_wire[inp_size-1],b[inp_size-2:0]}},c_wire[inp_size-1:0]);
  assign c_wire[0]=1'b1;
endmodule

```

δ. Κλιμάκωση και μήκος λέξης

Είδαμε ως τώρα τα κυκλώματα που υλοποιούν παράλληλη άθροιση ή αφαίρεση δύο δυαδικών αριθμών σε μορφή συμπληρώματος ως προς 2. Εξαιρέσει του αριθμού και της διάταξης των πυλών NOT στις εισόδους τους, έχουν ακριβώς τις ίδιες διεπαφές με το περιβάλλον και ακριβώς τα ίδια χαρακτηριστικά. Επομένως, πορίσματα που ισχύουν στους αθροιστές σχετικά με το μήκος λέξης και την κλιμάκωση εισόδων/εξόδου, καθώς και με το κρίσιμο μονοπάτι, θα ισχύουν και στους αφαιρέτες. Συνεπώς, στο εξής, αν και θα αναφερόμαστε μόνο σε «αθροιστές» ή «άθροιση», θα εννοούμε και τα δύο είδη βασικών μονάδων, εκτός εάν δηλώνεται ρητά ότι ισχύουν διαφορετικά πράγματα για τους αφαιρέτες.

Έστω λοιπόν, ότι έχουμε να αθροίσουμε δυο γενικά σήματα σε μορφή συμπληρώματος ως προς 2, το X, με παραμέτρους (n_x, p_x) και το Y με παραμέτρους (n_y, p_y) . Πρώτον, θα πρέπει να αποφασίσουμε ποιες θα είναι οι παράμετροι του αθροίσματος, (n_s, p_s) . Μια σύμβαση που θα ακολουθούμε πάντα στο εξής, όταν πρόκειται για άθροιση ή αφαίρεση, είναι ότι **η δεκαδική ακρίβεια του**

αποτελέσματος (d) θα ισούται πάντα με τη δεκαδική ακρίβεια των σημάτων εισόδου του κυκλώματος, d_i . Από τη (3.6) έχω ότι αυτό θα σημαίνει πως για κάθε έξοδο αθροιστή θα πρέπει να ισχύει:

$$d_i = n_s - p_s - 1 \quad (3.10)$$

Συνεπώς, αν καθορίσω την κλιμάκωση της εξόδου μπορώ άμεσα να καθορίσω και το μήκος λέξης της. Στο [E12] προτείνεται μετά από κάθε πρόσθεση/αφαίρεση η κλιμάκωση του αποτελέσματος να ισούται με το $\max(p_x, p_y) + 1$. Ωστόσο, αν και αυτό είναι ένα ασφαλές όριο για να μην γίνει ποτέ υπερχείλιση, δεν είναι αποδοτικό, καθώς σε συνδυασμό με την (3.10) σημαίνει ότι σε κάθε άθροιση, θα αυξάνουμε υποχρεωτικά το μήκος λέξης κατά 1.

Ευτυχώς, υπάρχει ένας πιο οικονομικός αλγόριθμος: αν ξέρουμε τις μέγιστες απόλυτες τιμές των X, Y , τότε, για τη μέγιστη απόλυτη τιμή του αθροίσματος S θα ισχύει:

$$|S| = |X| + |Y| \quad (3.11)$$

Σε ένα κύκλωμα, πάντα ξέρουμε τις μέγιστες απόλυτες τιμές όλων των σημάτων: περιορίζουμε την είσοδο σε ένα συγκεκριμένο εύρος τιμών (απαραίτητο, αν θέλουμε να ορίσουμε την κλιμάκωση για αυτή) και κατόπιν, από τους συντελεστές των πολλαπλασιαστών και τις αθροίσεις/αφαιρέσεις μπορούμε να βρούμε τη μέγιστη απόλυτη τιμή και για όλα τα υπόλοιπα σήματα. Έτσι, με χρήση της (3.11) και της (3.4) μπορούμε να βρούμε το βέλτιστο p για τη μέγιστη απόλυτη τιμή του αποτελέσματος. Θα χρησιμοποιήσουμε την τιμή αυτή; Η απάντηση είναι εξαρτάται, όπως θα δούμε ευθύς αμέσως.

Ας επανέλθουμε στο θέμα της άθροισης. Προφανώς για να έχουμε σωστά αποτελέσματα, θα πρέπει οι τα δεκαδικά bit του ενός σήματος να προστεθούν με τα δεκαδικά bit του άλλου και τα ακέραια bit του ενός με τα ακέραια bit του άλλου, δηλαδή, να υπάρξει μια «ευθυγράμμιση» των δύο τελεστών.

- Πρώτον, θα πρέπει υποχρεωτικά να αθροιστούν όλα τα bit ακέραιας αξίας των X, Y . Από την (3.5) έχω ότι $i_x = p_x + 1$, και $i_y = p_y + 1$. Επομένως, γι'αυτά θα χρειαστώ $\max(i_x, i_y)$ πλήρεις αθροιστές, δηλαδή, $\max(p_x, p_y) + 1$ (3.12)

Είναι φανερό ότι το σήμα με τη μικρότερη κλιμάκωση θα υποστεί ολίσηση $|p_x - p_y|$ θέσεις δεξιά και επέκταση προσήμου. Για λόγους ευκολίας σχεδιασμού, αν η κλιμάκωση του αποτελέσματος εξόδου που βρίσκουμε από τη μέγιστη απόλυτη τιμή του με χρήση της (3.4) είναι μεγαλύτερη της μεγαλύτερης κλιμάκωσης των εισόδων ($p_s > \max(p_x, p_y)$), τότε ενδέχεται να συμβεί υπερχείλιση, οπότε λαμβάνουμε στο αποτέλεσμα και το κρατούμενο εξόδου του αθροιστή και η κλιμάκωσή του ισούται με $\max(p_x, p_y) + 1$.

Σε διαφορετική περίπτωση, το κρατούμενο εξόδου δεν έχει σημασία και δεν το περιλαμβάνουμε στο αποτέλεσμα, το οποίο πλέον έχει κλιμάκωση $p_s = \max(p_x, p_y)$. Σε κάθε περίπτωση, το αποτέλεσμα έχει $p_s + 1$ «ακέραια» bits. Άρα, από την (3.10) ισχύει $n_s = d_i + p_s + 1$ (3.13)

- Στη συνέχεια, θα πρέπει να προστεθούν τα δεκαδικής αξίας bits. Ο X έχει το πολύ $d_x = n_x - p_x - 1$ τέτοια, και ο Y το πολύ $d_y = n_y - p_y - 1$ (εάν κάποιος από τους δύο έχει λιγότερα, η επέκταση προσήμου που θα γίνει πριν θα αυξήσει τον αριθμό τους σε αυτόν που αναφέρουμε). Εδώ τα πράγματα είναι πιο περίπλοκα: αν το $\min(d_x, d_y)$ είναι μεγαλύτερο ή ίσο του d_i , τότε, αθροίζονται τα $\min(d_x, d_y)$ δεκαδικά, και από το αποτέλεσμά τους κρατάμε μόνο τα d_i σημαντικότερα (το

κρατούμενο εξόδου τροφοδοτείται στον αθροιστή των ακεραίων bit). Συνεπώς, θα χρειαστούμε $\max(p_x, p_y) + \min(n_x - p_x, n_y - p_y)$ πλήρεις αθροιστές για τον παράλληλο αθροιστή (3.14)

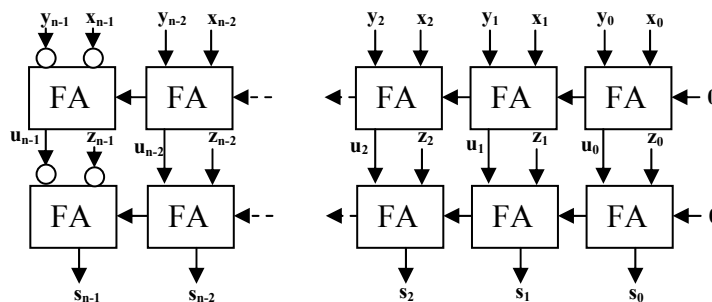
- Αν τώρα το $\min(d_x, d_y) < d_i$, τότε, θα πρέπει τα δεκαδικά bit από το σήμα με τα περισσότερα δεκαδικά να προωθηθούν κατ'ευθείαν στο αποτέλεσμα (ο αριθμός των πλήρων αθροιστών δίνεται πάλι από την (3.14)) Αν δεν ισχύει $\max(d_x, d_y) < d_i$, τότε, τα πιο σημαντικά από τα δεκαδικά του αριθμού με τα περισσότερα δεκαδικά που δεν προστίθενται θα καλύψουν τις κενές θέσεις. Σε αντίθετη περίπτωση θα πρέπει, αφού γίνει αυτό, τα εναπομείναντα δεκαδικά bit του αποτελέσματος να λάβουν την τιμή 0.

Διαπιστώνουμε ότι η άθροιση δύο σημάτων με διαφορετικές παραμέτρους (n, p) είναι μια αρκετά περίπλοκη-αν υλοποιηθεί σε συστηματικό αλγόριθμο- διαδικασία. Αυτό αποτελεί το σχεδιαστικό κόστος που πληρώνουμε επειδή χρησιμοποιούμε αριθμούς με διαφορετική κλιμάκωση.

ε. Κρίσιμο μονοπάτι και αθροιστές

Ας δούμε τώρα τι συμβαίνει με το θέμα του κρίσιμου μονοπατιού. Πρώτα από όλα, θα κάνουμε μια παραδοχή η οποία θα τηρείται και στην υπόλοιπη εργασία: **η χρονική καθυστέρηση που εμφανίζουν οι πύλες NOT είναι αμελητέα**. Συνεπώς, στο κρίσιμο μονοπάτι μας απασχολούν μόνο οι πλήρεις αθροιστές που αυτό περιλαμβάνει και, συνεπώς, αυτό μετράται σε πλήθος διαδοχικών πλήρων αθροιστών.

Στην (3.14) αποδείξαμε ότι ένας παράλληλος αθροιστής περιλαμβάνει $\max(p_x, p_y) + \min(n_x - p_x, n_y - p_y)$ πλήρεις αθροιστές στη σειρά. Συνεπώς, αν τα x και y προέρχονται από είσοδο του κυκλώματος ή καταχωρητή, δηλαδή, τα x_i, y_i είναι διαθέσιμα την ίδια χρονική στιγμή, τότε, προφανώς η επιβάρυνση στο κρίσιμο μονοπάτι θα ισούται με το πλήθος των πλήρων αθροιστών. Αξίζει όμως εδώ να μελετήσουμε την περίπτωση που η μια από τις δύο εισόδους προκύπτει από αθροιστή (ή πολλαπλασιαστή, όπως θα δείξουμε στη συνέχεια), δηλαδή, τα λιγότερο σημαντικά bit εμφανίζονται πρώτα. Ας θεωρήσουμε το παρακάτω σχήμα όπου προστίθενται τρεις αριθμοί μήκους n bit, όπου θεωρούμε ότι όλα τα σήματα εισόδου και εξόδου έχουν ακριβώς την ίδια κλιμάκωση:



Σχήμα 3.8 Άθροιση τριών αριθμών σε μορφή συμπληρώματος ως προς 2 με αθροιστές διάδοσης κρατούμενου

Μπορούμε να διαπιστώσουμε στο σχ. 3.8 ότι, αν τα x_i, y_i και z_i είναι διαθέσιμα την ίδια χρονική στιγμή, η συνολική καθυστέρηση για την παραγωγή των s_i είναι $n+1$, δηλαδή, ο δεύτερος παράλληλος αθροιστής αύξησε το κρίσιμο μονοπάτι μόνο κατά έναν πλήρη αθροιστή. Εύκολα μπορούμε, δε, να διαπιστώσουμε ότι αν τα

$\{u_i\}$ είχαν υποστεί ολίσθηση προς τα δεξιά κατά μία θέση (λόγω διαφορετικής κλιμάκωσης σε σχέση με τα z_i) τότε, το κρίσιμο μονοπάτι θα αυξανόταν κατά 1, ενώ, αν είχαν υποστεί ολίσθηση προς τα αριστερά κατά μια θέση, τότε η συνολική καθυστέρηση θα ισούταν πάλι με n . Αυτό μας οδηγεί στη διατύπωση του ακόλουθου θεωρήματος για τη συνεισφορά των παράλληλων αθροιστών διάδοσης κρατουμένου στο κρίσιμο μονοπάτι ενός κυκλώματος:

Έστω ένας αθροιστής με εισόδους δύο σήματα X και Y , εκ των οποίων το πρώτο έχει καθυστέρηση t_X και το δεύτερο t_Y , ενώ λόγω διαφορετικής κλιμάκωσης το ένα ολισθαίνει u_x θέσεις δεξιά πριν προστεθεί, και το δεύτερο u_y (προφανώς ένα από τα u_x, u_y θα είναι μηδέν). Τότε, χρειαζόμαστε συνολικά n πλήρεις αθροιστές για την πρόσθεσή τους:

- Αν το X προέρχεται από καταχωρητή ή είσοδο του φίλτρου, τότε $t_1=n$ (3.15a)
- Αν το X δεν προέρχεται από καταχωρητή ή είσοδο του φίλτρου, τότε $t_1=t_X+u_x+1$ (3.15b)
- Αν το Y προέρχεται από καταχωρητή ή είσοδο του φίλτρου, τότε $t_2=n$ (3.15c)
- Αν το Y δεν προέρχεται από καταχωρητή ή είσοδο του φίλτρου, τότε $t_2=t_Y+u_y+1$ (3.15d)
- Η συνολική καθυστέρηση για την παραγωγή όλου του αθροίσματος ισούται με $\max(t_1, t_2, 0)$ (3.15e)

Έτσι, μπορούμε με συστηματικό τρόπο να υπολογίσουμε τη συνεισφορά ενός αθροιστή στο κρίσιμο μονοπάτι του κυκλώματος, αφού τα n, u_x και u_y μπορούν να υπολογιστούν από τα $(n_x, p_x), (n_y, p_y)$ όπως δείξαμε στις (3.13), (3.14)

3. Πολλαπλασιαστές

α. Εισαγωγή

Στην υποενότητα αυτή θα ασχοληθούμε με τους παράλληλους πολλαπλασιαστές των κυκλωμάτων μας, οι οποίοι θα σχηματιστούν από τους αθροιστές και τους αφαιρέτες που μελετήσαμε πριν. Όπως αναφέραμε και στην αρχή του κεφαλαίου, οι αρχιτεκτονικές που θα υλοποιήσουμε ως το επίπεδο πύλης επιλέχθηκαν να είναι εκείνες των οποίων όλοι οι πολλαπλασιαστές είναι σταθεροί, δηλαδή, πολλαπλασιάζουν το σήμα εισόδου τους επί έναν σταθερό παράγοντα. Υπενθυμίζουμε, επίσης, εδώ ότι τα κυκλώματά μας είναι γραμμικά, δηλαδή ένας πολλαπλασιαστής θα πολλαπλασιάζει πάντα ένα σήμα επί ένα συντελεστή και όχι με μια συνάρτηση της εισόδου.

Όπως τα σήματα εισόδου (και κατ'επέκτασιν και όλα τα εσωτερικά σήματα του κυκλώματος) λαμβάνουν και αρνητικές τιμές και οι συντελεστές μπορεί να είναι αρνητικοί ή θετικοί. Κυκλώματα που υλοποιούν γενικούς πολλαπλασιασμούς αναφέρονται συχνά στη βιβλιογραφία, π.χ. στο [E6], όπου δείχνεται ότι ένα γενικός πολλαπλασιαστής $n \times n$ bit απαιτεί n^2 πλήρεις αθροιστές. Εμείς εδώ δε θα αναφερθούμε καθόλου σε αυτούς παρά μόνο σε σταθερές περιπτώσεις, όπου θα δείξουμε ότι κατά μέσο όρο χρειάζονται μόλις $n^2/3$ αθροιστές

β. Σταθεροί πολλαπλασιαστές δυαδικού δένδρου (carry-propagate)

Έστω ότι έχω ένα σήμα X σε μορφή συμπληρώματος ως προς 2 και μήκους n bit (προς το παρόν θα αγνοήσουμε χωρίς βλάβη της γενικότητας την κλιμάκωση) το οποίο σκοπεύουμε να το πολλαπλασιάσουμε με έναν σταθερό συντελεστή C (ο οποίος μπορεί να είναι και αρνητικός). Έστω $\{c_i\}$ η αναπαράσταση του συντελεστή

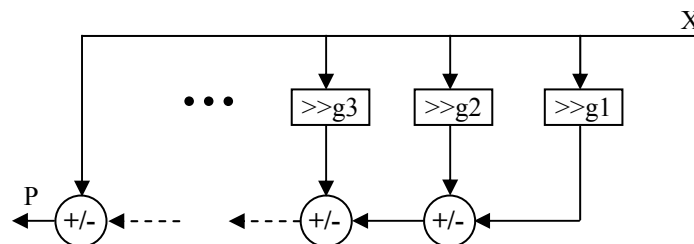
σε συμπλήρωμα ως προς 2, με μήκος m bits. Τότε, από την (3.1) έχω ότι το γινόμενο P θα ισούται με:

$$P = X * C = X * \left(-c_{m-1}2^{m-1} + \sum_{i=0}^{m-2} c_i 2^i \right) = -c_{m-1}(2^{m-1} * X) + \sum_{i=0}^{m-2} c_i(2^i * X) \quad (3.16)$$

Από την (3.16) έχω ότι, αν το C είναι σταθερό, τότε ο υπολογισμός του P ανάγεται σε διαδοχικές ολισθήσεις και αθροίσεις/αφαιρέσεις του σήματος εισόδου X. Είναι φανερό ότι όσο μικρότερος ο αριθμός των μη μηδενικών c_i , τόσο μικρότερος θα είναι και ο αριθμός των πράξεων που θα χρειάζονται και τόσο μικρότερο θα είναι το κύκλωμα και το συνολικό κρίσιμο μονοπάτι. Συνεπώς, συμφέρει σε αυτές τις περιπτώσεις να έχουμε το c_i σε μορφή csd: αφού η μορφή csd διασφαλίζει τα λιγότερα μη-μηδενικά στοιχεία, διασφαλίζει και τις λιγότερες δυνατές πράξεις. Έτσι, με τον αλγόριθμο του Reitwiesner μετατρέπουμε τον c_i από συμπλήρωμα του 2 σε csd μορφή.

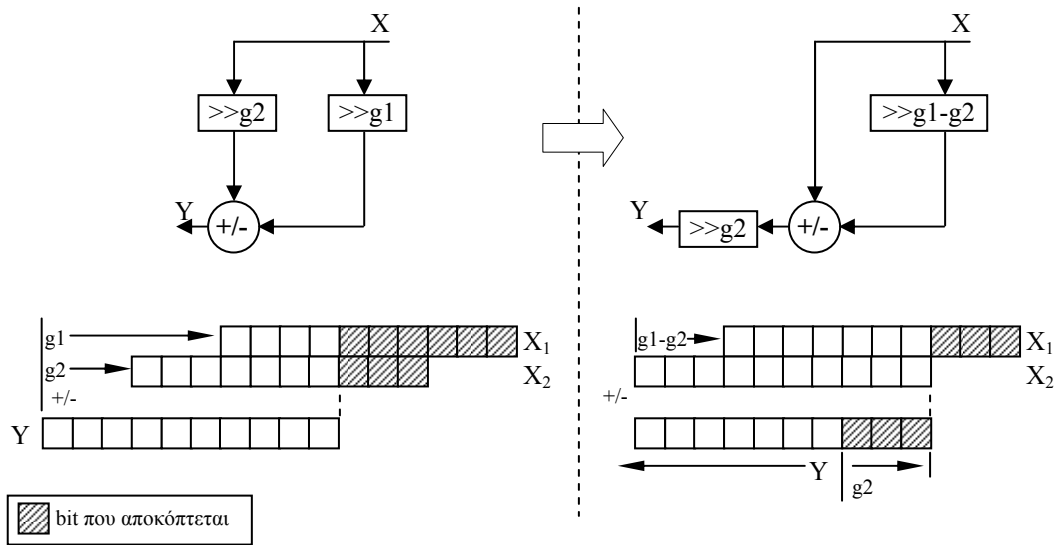
Σε αρκετές περιπτώσεις, μας ενδιαφέρει από ένα γινόμενο να λαμβάνουμε μόνο τα n πιο σημαντικά bit, όπου n το μήκος λέξης της εισόδου του πολλαπλασιαστή. Αυτό οδήγησε τον Parhi στο [C13] να προτείνει την υλοποίηση της (3.16) (με το C σε csd μορφή) με τη χρήση αθροιστών διάδοσης κρατούμενου μήκους n. Ωστόσο, και παρά τις βελτιώσεις που προτείνει, προειδοποιεί για την ενδεχόμενη μικρή ακρίβεια των αποτελεσμάτων. Εμείς εδώ, ωστόσο, θα αναπτύξουμε ένα συστηματικό τρόπο για τη δημιουργία και τη βελτιστοποίηση των πολλαπλασιαστών αυτών και θα τους χρησιμοποιήσουμε στα κυκλώματά μας. Εάν τα σφάλματα που εισάγουν στους υπολογισμούς είναι ανεκτά ή όχι, είναι κάτι που θα κριθεί εκ του αποτελέσματος.

Ας θεωρήσουμε, λοιπόν, τον C υπό csd μορφή $\{c_i\}$, και ας θέσουμε ως g_1 τις θέσεις δεξιότερα του πιο σημαντικού μη μηδενικού ψηφίου που είναι ένα το λιγότερο σημαντικό μη-μηδενικό ψηφίο του C, g_2 το δεύτερο λιγότερο σημαντικό κ.ο.κ. Τότε, ο υπολογισμός του γινομένου με αθροιστές/αφαιρέτες σταθερού μεγέθους μπορεί αν γίνει όπως παρακάτω:



Σχήμα 3.9 Απλή υλοποίηση σταθερού πολλαπλασιαστή με χρήση αθροιστών/αφαιρέτων σταθερού μήκους

Στο παραπάνω σχήμα μπορούν να γίνουν δυο σημαντικές βελτιώσεις: η πρώτη είναι να διατάξουμε τους αθροιστές/αφαιρέτες σε δένδρο: αν έχουμε συνολικά l μη-μηδενικά c_i , τότε είναι καλύτερο για το κρίσιμο μονοπάτι να έχουμε ένα δυαδικό δένδρο $\lceil \log_2 l \rceil + 1$ επιπέδων παρά l-1 διαδοχικές αθροίσεις.

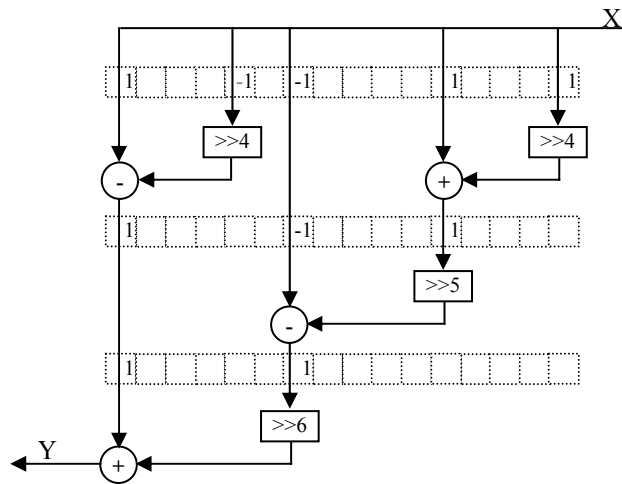


Σχήμα 3.10 Βελτίωση της ακρίβειας

Ένα άλλο πολύ πιο σημαντικό πρόβλημα είναι οι πολύ μεγάλες αποκοπές που λαμβάνουν χώρα: αν το g_1 π.χ. είναι πολύ μεγάλο τότε ίσως μόνο το πιο σημαντικό bit του X να αθροιστεί, από αυτό το συντελεστή. Αυτό οδηγεί σε μια απαράδεκτη αποκοπή των λιγότερο σημαντικών bit, των οποίων η άθροιση παράγει κρατούμενα που διαδίδονται εν τέλει και στα bit που αποτελούν το τελικό αποτέλεσμα. Ένας τρόπος να μετριαστεί αυτό είναι η πρόσθεση (που γίνεται σε δένδρο) να γίνεται με χρήση του συνόλου του μήκους του σήματος που αντιστοιχεί στο πιο σημαντικό από τα δύο ψηφία του συντελεστή και να γίνεται αποκοπή στο αποτέλεσμα. Αυτό γίνεται καλύτερα εμφανές στο σχήμα 3.10.

Βλέπουμε εκεί ότι ενώ στη δεύτερη περίπτωση το αποτέλεσμα Y είναι της ίδιας ακρίβειας με την πρώτη, στον υπολογισμό του έχουν ληφθεί υπ'όψιν περισσότερα bits από τα σήματα X_1 και X_2 : επομένως, μερικά κρατούμενα που επηρεάζουν το πρώτο bit του αποτελέσματος που δεν αποκόπτεται, δε χάνονται όπως στην πρώτη περίπτωση, αλλά διαδίδονται προς τα επόμενα στάδια. Έτσι πετυχαίνουμε καλύτερη ακρίβεια.

Η δημιουργία του πολλαπλασιαστή με χρήση των δύο παραπάνω κανόνων ουσιαστικά ανάγεται στην εύρεση ενός βέλτιστου δυαδικού δένδρου, δηλαδή, ενός δένδρου που ομαδοποιεί τις διπλανές εισόδους (έτσι ώστε να ελαχιστοποιεί το σφάλμα αποκοπής) και προωθεί το άθροισμά τους στην ίδια αξία με τη μεγαλύτερη είσοδο, όπως στο παρακάτω σχήμα.



Σχήμα 3.11 Αρχικό δένδρο πολλαπλασιαστή (ο συντελεστής είναι ο αριθμός 0.922423 σε μορφή csd και κλιμάκωση 0)

Το βέλτιστο δυαδικό δένδρο βρίσκεται περίπου ως εξής: αν σε ένα επίπεδο ο αριθμός των μη-μηδενικών ψηφίων είναι άρτιος, τότε προφανώς αθροίζουμε τα γειτονικά ψηφία, και το άθροισμά τους προωθείται στο επόμενο επίπεδο, με αξία όσο αυτή του μεγαλύτερου τελεστέου (βλ. και παραπάνω σχήμα). Σε διαφορετική περίπτωση, βρίσκουμε το πιο απομονωμένο μη μηδενικό ψηφίο (σε περίπτωση ισοπαλίας, προτιμούμε αυτό με τη μεγαλύτερη αξία καθώς επηρεάζεται λιγότερο από το σύνολο των αποκοπών), το προωθούμε αυτούσιο στην επόμενη βαθμίδα και το αφαιρούμε από το αρχικό επίπεδο, όπου κάνουμε την ίδια διαδικασία με πριν. Συνεχίζουμε παρόμοια στα επόμενα επίπεδα, έως ότου ο αριθμός των μη μηδενικών ψηφίων γίνει 1.

Ωστόσο, προτού διατυπώσουμε τον αλγόριθμο δημιουργίας πολλαπλασιαστών δυαδικού δένδρου carry-save, θα πρέπει να λύσουμε ένα σημαντικό πρόβλημα που προκύπτει από τη χρήση αθροιστών σταθερού μήκους. Όπως βλέπουμε και στα σχήματα (3.6), (3.7), όταν προσθέτουμε/αφαιρούμε δύο αριθμούς μήκους n bit, το αποτέλεσμα έχει μήκος $n+1$ bit. Συνεπώς, εφ'όσον ο επόμενος αθροιστής, στον οποίον αυτό θα διοχετευθεί ως είσοδος, έχει μήκος εισόδων επίσης n bit, μια λύση είναι να συμπεριλάβουμε σε αυτόν το κρατούμενο εξόδου ως το πιο σημαντικό bit της εισόδου και να κάνουμε αποκοπή του λιγότερο σημαντικού bit του αποτελέσματος της προηγούμενης άθροισης. Αυτό προφανώς θα εισάγει μια πολύ μεγάλη αποκοπή στα αποτελέσματα, η οποία μάλιστα δεν είναι αναγκαία. Θα πρέπει να βρούμε έναν τρόπο να προβλέπουμε πότε μας χρειάζεται το κρατούμενο μιας πράξης στο επόμενο στάδιο, δηλαδή, πότε ενδέχεται να υπάρχει υπερχειλίση. Ευτυχώς, όπως θα αποδείξουμε αμέσως, αν εφαρμόζουμε πάντα την τεχνική του σχ. 3.10, υπερχειλίση υπάρχει μόνο αν ο τελεστέος που δεν υφίσταται αποκοπή των περισσότερων σημαντικών bit είναι το σήμα εισόδου του φίλτρου και σε αυτόν προστίθεται μια ομόσημη ποσότητα ή αφαιρείται μια ετερόσημη ποσότητα(3.17). Σε αυτή την περίπτωση, η αξία της εξόδου αυξάνει κατά μια θέση

Απόδειξη: Έστω ότι στο σχ. 3.10, στη δεύτερη περίπτωση ο τελεστέος X_2 είναι το σήμα εισόδου του πολλαπλασιαστή X επί την αξία του αντίστοιχου ψηφίου του τελεστή, έστω 2^k . Τότε προφανώς ισχύει για τον X_1 (όπως μπορούμε να

διαπιστώσουμε από το σχήμα 3.11), ότι $X_1 = \sum_{i \in \{e, f, g, \dots\}} X * c_i * 2^i$, όπου όμως τα e, f, g, \dots

είναι μικρότερα του k . Άρα, σε κάθε περίπτωση, $|X_1| < |X_2|$, αφού $\sum_{i=1}^n \frac{1}{2^i} < 1$ για κάθε

πεπερασμένο n . Συνεπώς:

- Αν ο X_1 είναι ομόσημος του X_2 και αφαιρείται από αυτόν, ή είναι ετερόσημος και προστίθεται σε αυτόν, το αποτέλεσμα προφανώς θα είναι μικρότερο κατ' απόλυτη τιμή από τον X_2 , οπότε δε θα χρειαζούμαστε το κρατούμενο εξόδου. Συνεπώς, μπορούμε στην επόμενη πράξη να χρησιμοποιήσουμε την έξοδο του αθροιστή/αφαιρέτη με όλα τα bit της. Αν τώρα, αυτή η έξοδος χρησιμοποιηθεί ως ο μεγαλύτερης αξίας τελεστέος σε μια επόμενη πράξη, τότε, ο άλλος αριθμός που θα προστεθεί σε αυτή θα έχει, για τους ίδιους λόγους με πριν, μικρότερη απόλυτη τιμή από το X_1 . Επομένως, το μεγαλύτερο αποτέλεσμα που θα έχουμε από την δεύτερη πράξη (είτε είναι πρόσθεση είτε αφαίρεση) θα είναι σίγουρα μικρότερο (απόλυτα) από το X_2 , οπότε, πάλι δε χρειαζόμαστε το κρατούμενο εξόδου, αφού το αρχικό μήκος λέξης είναι αρκετό για κάθε X_2
- Εάν τώρα ο X_1 είναι ομόσημος του X_2 και προστίθεται σε αυτόν, ή είναι ετερόσημος και αφαιρείται, τότε, μπορεί να υπάρξει υπερχειλίση, οπότε θα πρέπει στην έξοδο να λάβουμε υπ' όψιν το κρατούμενο εξόδου και να αποκόψουμε το λιγότερο σημαντικό bit του αθροίσματος, μετατοπίζοντας το αποτέλεσμα μια θέση αριστερά στο επόμενο στάδιο (δηλαδή, αυξάνουμε την αξία του). Ωστόσο, αν το αποτέλεσμα αυτό προστεθεί ως τελεστέος υψηλότερης αξίας σε κάποια άλλη πράξη, τότε το αποτέλεσμα αυτής αποκλείεται να υπερχειλίσει. Ο λόγος είναι ότι ο επόμενος προσθετέος X' θα έχει απόλυτη τιμή σίγουρα μικρότερη του X_1 , και επιπλέον, για το νέο άθροισμα θα ισχύει:

$$S = X_2 + X_1 + X' = X * 2^k + \sum_{i \in \{e, f, g, \dots\}} X * c_i * 2^i + \sum_{j \in \{o, p, q, \dots\}} X * c_j * 2^j =$$

$$X * 2^k * \left(1 + \sum_{i \in \{e, f, g, \dots\}} c_i * 2^{i-k} + \sum_{j \in \{o, p, q, \dots\}} c_j * 2^{j-k} \right)$$

- Λόγω του ότι τα o, p, q είναι μικρότερα από τα e, f, g τα οποία είναι με τη σειρά τους μικρότερα από το k , ισχύει ότι:

$$S = X * 2^k * \left(1 + \sum_{i \in \{e, f, g, \dots\}} c_i * 2^{i-k} + \sum_{j \in \{o, p, q, \dots\}} c_j * 2^{j-k} \right) < X * 2^k \left(1 + \sum_{i=1}^{\infty} 2^i \right) = 2X * 2^k$$

Όμως, η μορφή με την οποία διαδίδεται το άθροισμα/διαφορά των X_1, X_2 , είναι ικανό να «χωρέσει» κάθε $2^{k+1}X$, αφού προήλθε από το σήμα που μετέφερε το $2^k X$ με αύξηση της κλιμάκωσης κατά 1 (αυτό έγινε όταν λάβαμε στην έξοδο το κρατούμενο εξόδου και απορρίψαμε το μικρότερης αξίας bit του αποτελέσματος). Άρα, δε θα υπάρξει υπερχειλίση.

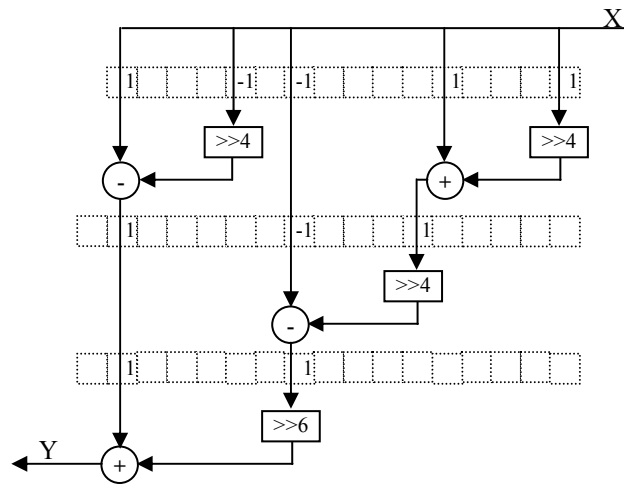
Συνεπώς, είμαστε πλέον σε θέση να διατυπώσουμε τον αλγόριθμο δημιουργίας σταθερών πολλαπλασιαστών δυαδικών δένδρων με σταθερού μήκους αθροιστές/αφαιρέτες (**αλγόριθμος βέλτιστου δένδρου πολλαπλασιαστή**) (3.18)

Αλγόριθμος βέλτιστου δένδρου πολλαπλασιαστή

1. Μετέτρεψε το σταθερό συντελεστή σε μορφή csd (από n bit αναπαράσταση ως προς 2, **n σταθερό**). Το διάνυσμα αυτό λέγεται διάνυσμα εισόδου και σε κάθε έναν από τους μη-μηδενικούς όρους αντιστοίχισε το σήμα εισόδου.
2. Δημιούργησε ένα καινούριο (κενό) διάνυσμα αναπαράστασης δυαδικών αριθμών, το οποίο στο εξής θα ονομάζουμε διάνυσμα εξόδου, μήκους $n+1$. Αν στο διάνυσμα εισόδου ο αριθμός των μη

- μηδενικών ψηφίων είναι άρτιος, προχώρα στο επόμενο βήμα. Αλλιώς, βρες το μεγαλύτερης αξίας πιο απομακρυσμένο μη-μηδενικό όρο, προώθησέ τον στην ίδια θέση στο διάνυσμα εξόδου, σβήσ τον από το διάνυσμα εισόδου και προχώρα στο επόμενο βήμα
3. Αρχισε να αθροίζεις τους γειτονικούς όρους ως εξής, ξεκινώντας από τα χαμηλότερης αξίας στοιχεία του διανύσματος εισόδου μέχρι να φτάσεις στο τέλος του:
 - a. Σημείωσε τη θέση, την τιμή και το σήμα στο οποίο αντιστοιχεί ο πρώτος μη μηδενικός όρος που συναντάς
 - b. Όταν συναντήσεις και τον δεύτερο μη-μηδενικό όρο, ανάλογα με τη θέση του και τη θέση του προηγούμενου μη-μηδενικού όρου, ολίσθησε κατάλληλες θέσεις δεξιά το σήμα του προηγούμενου όρου και άθροισέ το στο σήμα του όρου αυτού ως εξής: αν και οι δύο όροι είναι ομόσημοι, προσέθεσε τα σήματά τους και θέσε τον όρο του αποτελέσματος ίσο με 1 ή -1, ανάλογα με την τιμή των δύο όρων. Αν οι δύο όροι είναι ετερόσημοι, αφάιρεσε τα σήματά τους και θέσε τον όρο του αποτελέσματος ίσο με 1, αφού φροντίσεις να αφαιρεθεί το σήμα του όρου που έχει τιμή -1. Έτσι, διασφαλίζεται ο μικρότερος δυνατός αριθμός αφαιρέτων (που όπως είδαμε πριν, έχουν ελαφρά μεγαλύτερες απαιτήσεις σε υλικό)
 - c. Από το είδος της πράξης και από την αξία των σημάτων εισόδου, καθόρισε την τιμή (ομόσημη/ετερόσημη του σήματος εισόδου του πολλαπλασιαστή) του σήματος εξόδου. Επιπλέον, σύμφωνα με την αρχή (3.17) καθόρισε αν το αποτέλεσμα πρόκειται να υπερχειλίσει ή όχι. Αν ναι, τότε προώθησε τον όρο αποτελέσματος στο διάνυσμα εξόδου στην αμέσως μεγαλύτερης αξίας θέση από αυτή του δεύτερου όρου, και αντιστοίχισε σε αυτόν (τον όρο εξόδου) ως σήμα το κρατούμενο εξόδου της πράξης και τα (n-1) σημαντικότερα bit του αποτελέσματος της. Σε διαφορετική περίπτωση, τοποθέτησε τον όρο εξόδου στο διάνυσμα εξόδου σε θέση ίσης αξίας με τον δεύτερο όρο και αντιστοίχισε σε αυτόν ως σήμα τα n bit του αποτελέσματος της πράξης, χωρίς το κρατούμενο εξόδου.
 - d. Προχώρα στον αμέσως σημαντικότερο όρο στο διάνυσμα εισόδου. Εάν δε συναντήσεις το τέλος του διανύσματος, επέστρεψε στο a., αλλιώς προχώρα στο 4
 4. Αν ο αριθμός των μη-μηδενικών όρων στο διάνυσμα εξόδου είναι μεγαλύτερος του 1, τότε αντικατέστησε το διάνυσμα εισόδου με το διάνυσμα εξόδου, και εκτέλεσε το βήμα 2. Αλλιώς, ανέθεσε στην έξοδο του πολλαπλασιαστή το μοναδικό σήμα που υπάρχει στο διάνυσμα εξόδου.

Για παράδειγμα, έστω πως θέλουμε να εφαρμόσουμε τον παραπάνω αλγόριθμο στη δημιουργία σταθερού πολλαπλασιαστή με συντελεστή το 0.922423. Τότε, το σχ. 3.11 ξανασχεδιάζεται ως εξής:



Σχήμα 3.12 Εφαρμογή του αλγορίθμου βέλτιστου δένδρου πολλαπλασιαστή στο σχ. 3.11

γ. Κλιμάκωση και μήκος λέξης σε πολλαπλασιαστές

Ας δούμε τώρα τι συμβαίνει με το θέμα της κλιμάκωσης και του μήκους λέξης της εξόδου ενός πολλαπλασιαστή. Έστω ένας πολλαπλασιαστής με είσοδο ένα σήμα X με (n_X, r_X) , συντελεστή p με (n_P, r_P) , και έξοδο Y , με χαρακτηριστικά (n_Y, r_Y) . Σε συμφωνία με τα παραπάνω, **υιοθετούμε τις παρακάτω αρχές για κάθε πολλαπλασιαστή που θα υλοποιήσουμε:**

- Ορίζουμε $n_P = n_X$ (3.19a)
 - Ορίζουμε $n_Y = n_X$ (3.19b)
 - Από τον αλγόριθμο βέλτιστου δένδρου πολλαπλασιαστή, διαπιστώνουμε άμεσα ότι ισχύει $r_Y = r_X + r_P$. (3.19c)
- Το r_P προκύπτει από χρήση της (3.4) στο P

δ. Κρίσιμο μονοπάτι πολλαπλασιαστή

Εν γένει, ο ακριβής καθορισμός της συνολικής καθυστέρησης σε έναν πολλαπλασιαστή που προκύπτει από την παραπάνω διαδικασία είναι μια μάλλον περίπλοκη υπόθεση λόγω της πιθανότητας κάποια σήματα να μην αθροίζονται στο επόμενο επίπεδο αλλά να προωθούνται αυτούσια στο μεθεπόμενο (βλ. βήμα 2 του αλγορίθμου βέλτιστου δένδρου πολλαπλασιαστή). Ωστόσο, εδώ θα αναπτύξουμε έναν συστηματικό αλγόριθμο καθορισμού ενός αρκετά ικανοποιητικού άνω φράγματος για την καθυστέρηση.

Έστω, λοιπόν, ένας πολλαπλασιαστής με είσοδο X με μήκος λέξης n_X bit και με συντελεστή P , ο οποίος σε csd μορφή έχει k μη-μηδενικά bit. Τότε, ισχύουν τα εξής:

- Το δένδρο των αθροιστών/αφαιρητών που υλοποιεί τον πολλαπλασιαστή έχει $\lceil \log_2 k \rceil + 1$ επίπεδα
- Από το σχ. 3.8 έχω ότι το πρώτο επίπεδο αθροιστών συνεισφέρει (ο κάθε ένας) καθυστέρηση ίση με n_X πλήρεις αθροιστές. Στη συνέχεια, για κάθε επόμενο επίπεδο η καθυστέρηση αυξάνει κατά έναν πλήρη αθροιστή, ενώ για κάθε μετατόπιση των εισόδων προς τα δεξιά πριν την άθροιση επίσης κατά έναν πλήρη αθροιστή. Επειδή μπορούμε, π.χ. στο σχ. 3.12 να δούμε ότι ισχύει η αρχή της επαλληλίας για τις καθυστερήσεις που οφείλονται σε δεξιές

ολισθήσεις, τελικά έχει σημασία η απόσταση του όρου που αντιστοιχεί στο αποτέλεσμα του πρώτου αθροιστή του πρώτου επιπέδου (δηλαδή, αυτού που αθροίζει τους μικρότερης αξίας όρους) από το πιο σημαντικό μη-μηδενικό ψηφίο της csd αναπαράστασης. Έτσι, έχουμε τον εξής αλγόριθμο:

Αλγόριθμος εύρεσης άνω φράγματος για την καθυστέρηση του πολλαπλασιαστή

1. Καθυστέρηση= $n_x + \text{ceiling}(\log k / \log 2)$
2. Μετέτρεψε τον συντελεστή του πολλαπλασιαστή σε μορφή csd. Αν το διάνυσμα αναπαράστασης έχει περιττό αριθμό μη-μηδενικών όρων αφαίρεσε τον πιο απομακρυσμένο από αυτούς.
3. Βρες τον δεύτερο λιγότερο σημαντικό μη μηδενικό όρο του διανύσματος αναπαράστασης. Αν είναι ομόσημος με τον λιγότερο σημαντικό μη-μηδενικό, θέσε $\text{posl} = \langle \text{θέση του όρου} \rangle + 1$, αλλιώς θέσε $\text{posl} = \langle \text{θέση του όρου} \rangle$
4. Βρες τον πιο σημαντικό μη μηδενικό όρο του διανύσματος αναπαράστασης. Αν είναι ομόσημος με τον επόμενο πιο σημαντικό όρο, θέσε $\text{posh} = \langle \text{θέση του όρου} \rangle + 1$, αλλιώς θέσε $\text{posh} = \langle \text{θέση του όρου} \rangle$
5. Καθυστέρηση=Καθυστέρηση+ $\text{posh} - \text{posl}$ (σε πλήρεις αθροιστές)

Πόσο όμως μεγαλύτερο μπορεί να είναι αυτό το άνω φράγμα από την πραγματική τιμή; Στη χειρότερη περίπτωση, το αποτέλεσμα του πρώτου αθροιστή του πρώτου επιπέδου θα προστίθεται κατ'ευθείαν στον όρο μεγαλύτερης αξίας, παρακάμπτοντας τα ενδιάμεσα $\lceil \log_2 k \rceil - 2$ επίπεδα. Επειδή ο συντελεστής είναι σε csd αναπαράσταση, θα ισχύει, λόγω των (3.14), στη χειρίστη περίπτωση $k = n_x / 2$, άρα, το άνω φράγμα που θα υπολογίσαμε πριν είναι μόλις $\lceil \log_2 n_x \rceil - 3$ πλήρεις αθροιστές μεγαλύτερο. Αυτό σημαίνει ότι ακόμα και εάν έχουμε αριθμητική 64bit, η υπερεκτίμηση θα ισούται στη χειρότερη περίπτωση μόλις με 3 (ενώ η συνολική καθυστέρηση θα είναι αρκετά μεγαλύτερη του 64). Έστω t_M η καθυστέρηση που υπολογίζουμε. Από όλα τα προηγούμενα έχω ότι πάντοτε:

$$t_M < 2n_x + \lceil \log_2 n_x \rceil - 2 \tag{3.20}$$

Στην περίπτωση που το σήμα εισόδου του πολλαπλασιαστή προέρχεται από αθροιστή ή άλλο πολλαπλασιαστή, τότε μπορούμε εύκολα να δούμε ότι η μέγιστη επιβάρυνση στο κρίσιμο μονοπάτι θα ισούται με $t_M - n_x + 1$

$$\tag{3.21}$$

4. Στοιχεία καθυστέρησης

Τέλος, θα αναφέρουμε επιγραμματικά τα χαρακτηριστικά των στοιχείων καθυστέρησης, ή καταχωρητές, τα οποία μεταβάλλουν την τιμή της εξόδου τους μόνο στις θετικές ακμές του ρολογιού. Συνήθως ένα τέτοιο στοιχείο αποτελείται από μια κατάλληλου μήκους παράλληλη σειρά D Flip-Flop. Επειδή, ωστόσο, αυτά υλοποιούνται αποδοτικά με διαφορετικό τρόπο για κάθε τεχνολογία, θα αρκεστούμε απλά στην περιγραφή τους σε Verilog επιπέδου συμπεριφοράς. Ο κώδικας, λοιπόν, που υλοποιεί τα D Flip-Flop και τους καταχωρητές είναι ο εξής:

```
module register(out, inp, clk, clr);
  //δημιουργία "καταχωρητή" (στοιχείου καθυστέρησης) από
  //εν παραλλήλω διάταξη DFF's από το module dff
  parameter inp_size=8;
  output [inp_size-1:0] out;
  input [inp_size-1:0] inp;
  input clk, clr;
  dff fdd_i[inp_size-1:0] (out, inp, clk, clr);
endmodule
```

```

endmodule

module dff(out, inp, clk, clr);
    //d flip-flop, σε θετικούς παλμούς
    //του ρολογιού και ενώ το clr είναι
    //0 μεταφέρει την είσοδο στην έξοδο;
    //αλλιώς πάντα 0
    output reg out;
    input inp, clk, clr;
    always@( clr, posedge clk) begin
        if(clr)
            out=1'b0;
        else if(clk)
            out=inp;
    end
endmodule

```

Όπως βλέπουμε και στον παραπάνω κώδικα, τα D Flip-Flop εκτός από την είσοδο του ρολογιού έχουν και μια είσοδο που θέτει τις εξόδους τους ασύγχρονα στο 0. Ένας καταχωρητής είναι σημαντικός κατά την αναζήτηση του κρίσιμου μονοπατιού στο γράφο ενός κυκλώματος, καθώς όλα τα υποψήφια κρίσιμα μονοπάτια ή ξεκινάνε ή/και τερματίζουν σε αυτούς.

Δ. Υλοποίηση σε επίπεδο πύλης

1. Εισαγωγή

Όπως αναφέραμε και στον πίνακα 2.10, στο κεφάλαιο αυτό θα υλοποιήσουμε ως το επίπεδο πύλης τις πέντε βέλτιστες αρχιτεκτονικές του προηγούμενου κεφαλαίου με σταθερούς πολλαπλασιαστές. Λόγω των σταθερών πολλαπλασιαστών θα είμαστε σε θέση να επιτύχουμε σημαντικότερη οικονομία υλικού σε σύγκριση με τη χρήση γενικών πολλαπλασιαστών: συνεπώς μπορούμε εδώ να εφαρμόσουμε άμεσα τα αποτελέσματα της ενότητας Γ3 αυτού του κεφαλαίου.

Οι διάφορες αρχιτεκτονικές θα υλοποιηθούν με αθροιστές/αφαιρέτες carry-save, καθώς και με πολλαπλασιαστές, τα κυκλώματα και τον κώδικα verilog των οποίων παρουσιάσαμε στην προηγούμενη ενότητα. Επιπλέον, ακολουθώντας τις συμβάσεις σχετικά με το μήκος λέξης και την ακρίβεια δεκαδικών στην έξοδο κάθε στοιχείου του κυκλώματος, θα έχουμε μεταβλητά (αλλά σταθερό στο χρόνο!) μήκος λέξης και κλιμάκωση στα διάφορα σήματα. Η περιγραφή και υλοποίηση θα γίνει με χρήση της γλώσσας verilog σε επίπεδο που να χρησιμοποιεί τα παραμετροποιήσιμα εξαρτήματα που ορίσαμε (επίσης σε verilog) πριν: έτσι, κατ' ουσίαν ο τελικός κώδικας υλοποιεί το φίλτρο σε επίπεδο πύλης-πλήρους αθροιστή.

Τέλος, θα συγκρίνουμε με ειδικά test benches τις εξόδους της κάθε αρχιτεκτονικής με είσοδο ένα δεδομένο σήμα, με τις εξόδους που θα μας έδινε *ιδανικά* ο μετασχηματισμός wavelet του CDF 9/7 (από το matlab) για το ίδιο σήμα εισόδου.

2. Περιβάλλον δημιουργίας και προσομοίωσης κυκλωμάτων

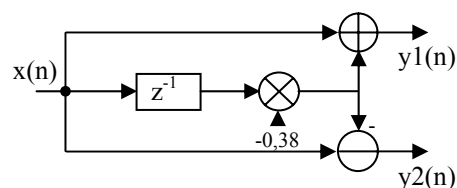
α. Δημιουργία κώδικα verilog: το πρόγραμμα sfc.c

Η μεταφορά καθεμιάς από τις επιλεγείσες αρχιτεκτονικές του πίνακα 2.10 σε κώδικα verilog που να σέβεται τις αρχές που θέσαμε στην ενότητα Γ και υλοποιεί τους σταθερούς πολλαπλασιαστές είναι μια μάλλον άχαρη δουλειά για να γίνει «χειροκίνητα». Επιπλέον, η δημιουργία των φίλτρων «με το χέρι» ενέχει δύο σημαντικούς κινδύνους: πρώτον, είναι εύκολο να γίνουν σφάλματα λόγω της επαναληπτικότητας της εργασίας (που είναι δύσκολο να εντοπιστούν, ακολούθως) και επιπλέον, δεν υπάρχει καθόλου ευελιξία. Για παράδειγμα, εάν διαπιστώσουμε ότι κάναμε λάθος κατά την αντιγραφή ενός συντελεστή, θα πρέπει ο πολλαπλασιαστής που τον χρησιμοποιεί να ξανασχεδιαστεί από την αρχή, σύμφωνα με τον αλγόριθμο βέλτιστου δένδρου πολλαπλασιαστή.

Για όλους αυτούς τους λόγους, λοιπόν, και με δευτερεύοντα στόχο την παραγωγή γενικότερων αποτελεσμάτων από αυτή τη διπλωματική, δημιουργήσαμε στα πλαίσια της εργασίας το απλό εργαλείο **sfc** (simple filter creator), το οποίο παράγει, με αυτόματο τρόπο, τον κώδικα verilog οποιουδήποτε μη-αναδρομικού γραμμικού δικτύωματος που αποτελείται από αθροιστές/αφαιρέτες, πολλαπλασιαστές και στοιχεία καθυστέρησης από μια απλούστατη, netlist περιγραφή, η οποία, για λόγους τυποποίησης, σώζεται σε ένα αρχείο κειμένου με κατάληξη .sf.

Το αρχείο .sf θα πρέπει να περιέχει το όνομα του φίλτρου, τις εξόδους (με προαιρετικά τα πόσα πιο σημαντικά bits επιθυμούμε να λάβουμε από αυτές), τις εισόδους (με το μήκος λέξης τους, και τη μέγιστη απόλυτη τιμή τους) και στη συνέχεια τις δηλώσεις των στοιχείων σε γλώσσα παρόμοια με αυτή του προγράμματος spice. Το πρόγραμμα sfc διαβάζει το αρχείο αυτό, υλοποιεί τους σταθερούς πολλαπλασιαστές ως βέλτιστους πολλαπλασιαστές δυαδικού δένδρου carry-save, (τους οποίους αναλύσαμε πριν, και με τον συντελεστή να έχει όσα bits έχει και το σήμα εισόδου του πολλαπλασιαστή, σε μορφή συμπληρώματος ως προς 2 ή carry-save, ανάλογα με τα ορίσματα που καλείται από τον χρήστη). Επίσης, υπολογίζει το μήκος λέξης και την κλιμάκωση κάθε σήματος του κυκλώματος ανάλογα με τις τιμές των εισόδων, τους συντελεστές και τις πράξεις που συναντά (λαμβάνοντας υπ' όψιν όσα αναφέραμε ως τώρα σε αυτό το κεφάλαιο), και στο τέλος εκτυπώνει τον κώδικα verilog που υλοποιεί αυτό το κύκλωμα, παραθέτοντας (στα σχόλια) την κλιμάκωση των εισόδων/εξόδων και εάν υπάρχει αποκοπή στα bits των εξόδων (και πόση). Επιπλέον, σε μορφή σχολίων εκτυπώνει και το κρίσιμο μονοπάτι του κυκλώματος (σε αριθμό πλήρων αθροιστών), τους πλήρεις αθροιστές, τις πύλες NOT και τα D Flip-Flop που αυτό χρειάζεται.

Για παράδειγμα, ας πούμε ότι θέλουμε να υλοποιήσουμε το παρακάτω απλό δίκτυωμα:



Σχήμα 3.13 Ένα απλό δίκτυωμα

όπου η είσοδος x παίρνει τιμές μεταξύ $(-1, 1)$ και έχει πλάτος 16 bit, ενώ θέλουμε από την έξοδο $y1$ να λάβουμε μόνο τα 16 πιο σημαντικά bits. Ο πολλαπλασιαστής θέλουμε να υλοποιηθεί με συντελεστή σε μορφή csd. Τότε, δημιουργούμε το παρακάτω αρχείο κειμένου (ascii):

my_filter.sf

```
name my_filter;

output y1_n 16;
output y2_n;

input x_n 16 0.999999999;

delay x_n x_delayed;
mult x_delayed x_coef -0.38;
add x_n x_coef y1_n;
sub x_coef x_n y2_n;
```

και δίνουμε στη γραμμή εντολών:

```
sfc csd my_filter.sf my_filter.v
```

και το sfc, αφού μας εκτυπώσει στην οθόνη διάφορες πληροφορίες για τους πολλαπλασιαστές και την κατάσταση λειτουργίας του, τερματίζει και δημιουργεί το αρχείο my_filter.v το οποίο είναι ως εξής:

myfilter.v

```
module mult00 (out, inp);
    //coefficient given: -0.380000000000000000
    //csdform: -1010_000-1_0-100_-1001 resynth to: -0.3799896240234375
    //coef scaling is: -1

    output [15:0] out;
    input [15:0] inp;
    wire auxil;
    wire [16:0] w00_00, w00_01, w00_02,
                w01_00, w01_01;

    sub2c_cp #(.inp_size(16)) sub00_00 (w00_00,
        {inp[15:00]},
        {inp[15], inp[15], inp[15], inp[15:03]} );

    add2c_cp #(.inp_size(16)) add00_01 (w00_01,
        {inp[15:00]},
        {inp[15], inp[15], inp[15:02]} );

    sub2c_cp #(.inp_size(16)) sub00_02 (w00_02,
        {inp[15:00]},
        {inp[15], inp[15], inp[15:02]} );

    sub2c_cp #(.inp_size(16)) sub01_00 (w01_00,
        {w00_01[16:01]},
        {w00_00[15], w00_00[15], w00_00[15], w00_00[15], w00_00[15],
        w00_00[15], w00_00[15:06]} );

    add2c_cp #(.inp_size(16)) add02_00 ({auxil, out},
        {w00_02[15:00]},
        {w01_00[15], w01_00[15], w01_00[15], w01_00[15], w01_00[15],
        w01_00[15], w01_00[15:06]} );

endmodule

module my_filter (y1_n, y2_n, x_n, clk, clr);
```



```

//full adders: 112, not gates: 77
//dff: 16, critical path: 32

//output wires:
output [15:0] y1_n; //maxvalue: 1.379999998620
                //scaling: 1 uses: 16 MSB of 17
output [16:0] y2_n; //maxvalue: 1.379999998620
                //scaling: 1 uses: 17 MSB of 17

//input wires:
input [15:0] x_n; //maxvalue: 0.999999999000, scaling: 0
input clk, clr;

//signal-carrying wires:
wire [15:0] x_delayed, x_coef;
wire [16:0] y1_n_, y2_n_;

//dummy adder lsb wires:

//dummy carry wires:

//output signal assignments:
assign y1_n=y1_n_[16:1];
assign y2_n=y2_n_[16:0];

//multiplier instances:
mult00 mult00i(.out(x_coef), .inp(x_delayed));

//adder instances:
add2c_cp #(.inp_size(16)) add00({y1_n_[16:0]},
    {x_n[15:0]},
    {x_coef[15], x_coef[15:1]});
sub2c_cp #(.inp_size(16)) add01({y2_n_[16:0]},
    {x_coef[15], x_coef[15:1]},
    {x_n[15:0]});

//register instances:
register #(.inp_size(16)) reg00(x_delayed, x_n, clk, clr);

endmodule

```

οπότε, με βάση τα εξαρτήματα register, add2c_cp και sub2c_cp που ορίσαμε πριν, έχουμε μια πλήρως προσομοίωση/συνθέσιμη περιγραφή του φίλτρου σε gate-level verilog.

Λόγω του σχετικά μεγάλου μεγέθους του κώδικα του προγράμματος sfc (περίπου 2200 γραμμές σε 5 διαφορετικά αρχεία) δεν τον παραθέτουμε στη διπλωματική αυτή. Ωστόσο, τόσο το πρόγραμμα όσο και ο πηγαίος κώδικας είναι διαθέσιμα σε κάθε ενδιαφερόμενο από τον επιβλέποντα καθηγητή ή το συγγραφέα. Επιπλέον, λόγω πάλι του όγκου του (και επειδή παράγεται κατά αυτόματο τρόπο) δε θα παραθέσουμε τον κώδικα verilog που υλοποιεί τις υπό μελέτη αρχιτεκτονικές μας παρακάτω, παρά μόνο το περιεχόμενο του αντίστοιχου αρχείου .sf που δίδεται ως είσοδος στο sfc.

β. Πορεία προσομοίωσης και εξαγωγής αποτελεσμάτων

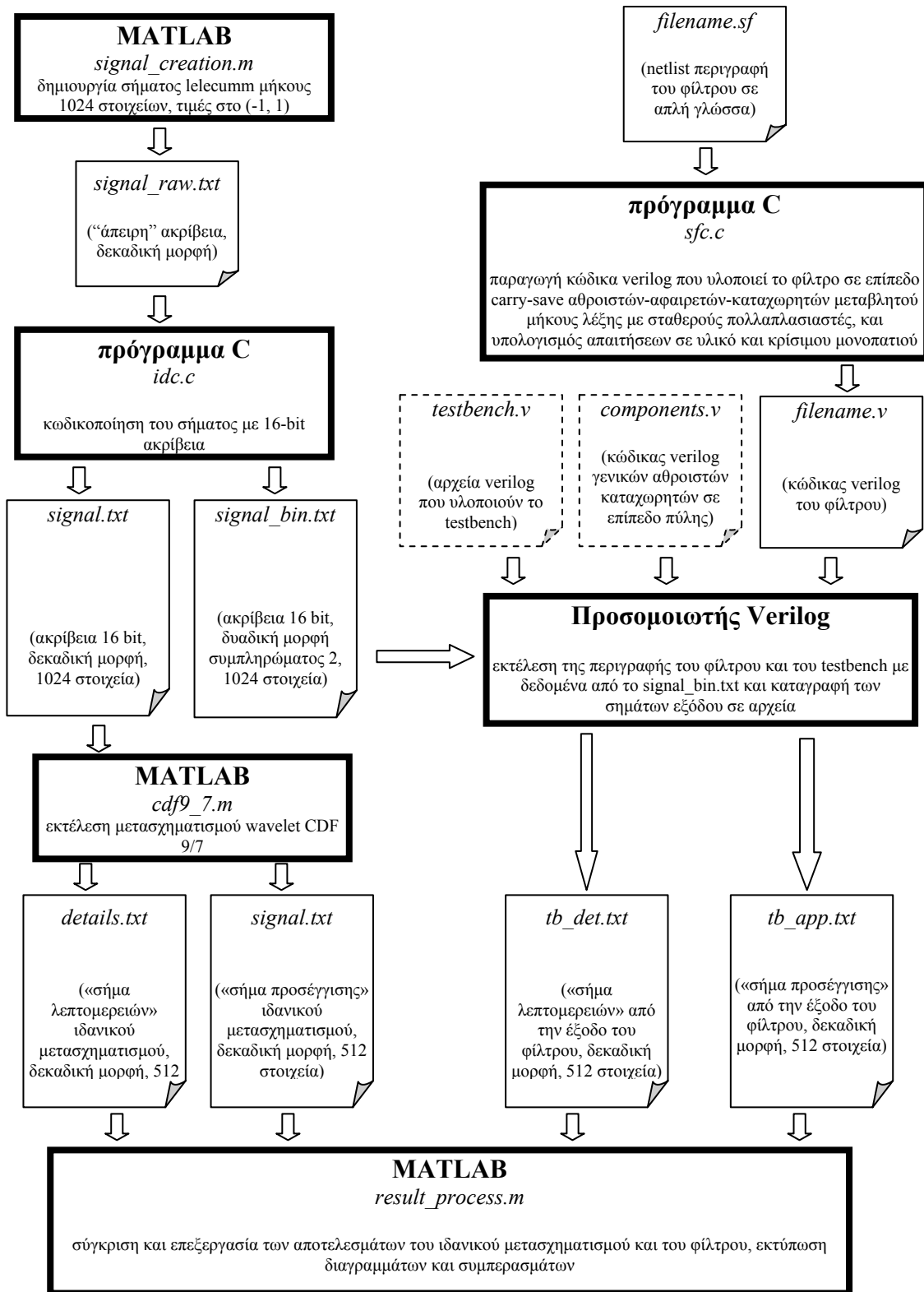
Ο κώδικας verilog που υλοποιεί ένα ψηφιακό αριθμητικό κύκλωμα και μόνο δεν επαρκεί για τη σωστή επαλήθευση της λειτουργίας του. Είναι απαραίτητο να γραφεί το testbench σε verilog, το οποίο θα παρέχει τα δεδομένα εισόδου και θα διαβάζει τα δεδομένα εξόδου από το κύκλωμα, καθώς και κάποια προγράμματα σε c/matlab, τα οποία θα επεξεργάζονται τα αποτελέσματα, τυπώνουν γραφικές παραστάσεις κ.ο.κ.

Η προσομοίωση γίνεται ως εξής: αρχικά, ένα πρόγραμμα σε matlab παράγει το υπό μελέτη σήμα, το οποίο στη συγκεκριμένη περίπτωση είναι τα πρώτα 1024 δείγματα του σήματος `leleccum` που περιέχεται στη βιβλιοθήκη του matlab. Στη συνέχεια, κανονικοποιεί το σήμα (έτσι ώστε να παίρνει τιμές στην (-1, 1)) και το εκτυπώνει σε ένα αρχείο εξόδου, σε δεκαδική προσημασμένη μορφή.

Όλες μας οι αρχιτεκτονικές έχουν μήκος λέξης στις εισόδους τους 16 bits, συνεπώς, θα πρέπει το σήμα να κωδικοποιηθεί (και σε δεκαδική, και σε δυαδική μορφή) με τόση ακρίβεια. Έτσι, δημιουργήσαμε σε γλώσσα c το πρόγραμμα `idc`, το οποίο διαβάζει την έξοδο του matlab και περιορίζει την ακρίβεια των δεδομένων σε 16 bit (με scaling 0, ή 15bit «δεκαδικά»). Το πρόγραμμα αυτό τυπώνει τη νέα εκδοχή του σήματος εισόδου σε δύο αρχεία, στο ένα σε προσημασμένη δεκαδική μορφή, και στο άλλο σε δυαδική μορφή συμπληρώματος ως προς 2.

Το πρώτο από τα δύο αυτά αρχεία το διαβάζει ένα άλλο πρόγραμμα matlab, το οποίο εκτελεί τον «ιδανικό» (από τις βιβλιοθήκες του matlab) μετασχηματισμό CDF 9/7 επί των δεδομένων εισόδου, και καταγράφει τα σήματα (εξόδου) λεπτομερειών και προσέγγισης σε δύο αρχεία κειμένου, σε δεκαδική προσημασμένη μορφή. Το δεύτερο από τα αρχεία αυτά διαβάζεται από τον εξομοιωτή verilog ο οποίος, χρησιμοποιώντας τον κώδικα του φίλτρου που δημιούργησε το `sfc`, βοηθητικά αρχεία verilog που υλοποιούν τον διαχωριστή άρτιων-περιττών, τους αθροιστές/καταχωρητές και υπό την καθοδήγηση του testbench που έχουμε γράψει για το σκοπό αυτό, υπολογίζει (μέσω του φίλτρου) εκ νέου τα σήματα λεπτομερειών και προσέγγισης και καταγράφει τα αποτελέσματα (σε δεκαδική προσημασμένη μορφή) σε δύο αρχεία. Λόγω του όγκου του, ο πηγαίος κώδικας verilog που υλοποιεί το testbench και τα βοηθητικά εξαρτήματα δεν παρατίθεται, αλλά είναι πάντοτε διαθέσιμος στον κάθε ενδιαφερόμενο από τον επιβλέποντα καθηγητή ή τον συγγραφέα. Το ίδιο ισχύει και για τα προγράμματα matlab.

Τέλος, ένα πρόγραμμα matlab αναλαμβάνει να διαβάσει τα τέσσερα αρχεία εξόδου και συγκρίνει τα αποτελέσματα. Πιο συγκεκριμένα, εκτυπώνει τις γραφικές παραστάσεις των «ιδανικών» (δηλαδή, αυτών που προήλθαν από το matlab) σημάτων προσέγγισης/λεπτομερειών έναντι των σημάτων που δημιουργήθηκαν από το φίλτρο, τη γραφική παράσταση του απόλυτου σφάλματος υπολογισμού κάθε σήματος εξόδου και τέλος υπολογίζει το μέσο απόλυτο σφάλμα υπολογισμού τόσο στο σήμα λεπτομερειών όσο και στο σήμα προσέγγισης. Συνοπτικά, όλη η διαδικασία υλοποίησης-προσομοίωσης φαίνεται στο σχήμα 3.14 της επόμενης σελίδας:



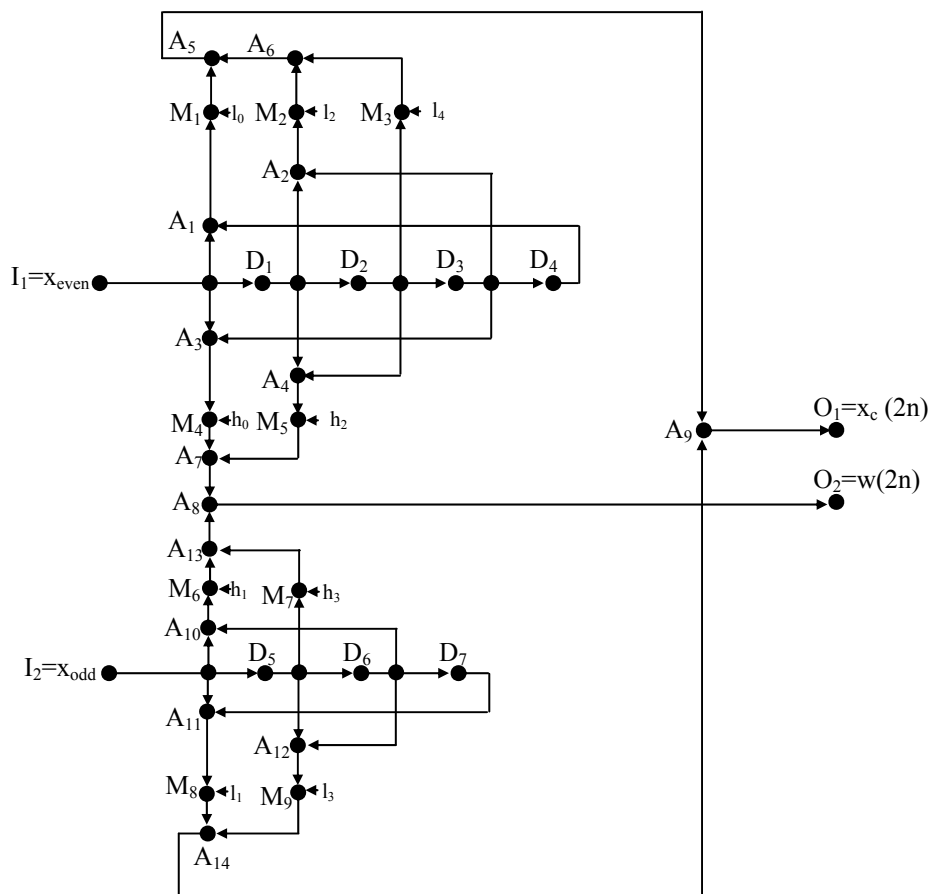
Σχήμα 3.14 Διάγραμμα του περιβάλλοντος δημιουργίας, προσομοίωσης και ελέγχου των ψηφιακών φίλτρων που υλοποιούν το μετασχηματισμό wavelet CDF 9/7

3. Υλοποίηση

Υπενθυμίζουμε εδώ ότι δεν ασχολούμαστε με τον διαχωριστή άρτιων-περιττών, τα δεδομένα εισόδου έχουν εύρος 16 bit και λαμβάνουν τιμές στην $(-1, 1)$, ενώ από τα δεδομένα εξόδου λαμβάνουμε πάντοτε τα 16 σημαντικότερα bits (με αποκοπή). Τέλος, όλοι οι πολλαπλασιαστές που δεν είναι δύναμη του 2 υλοποιούνται όπως προαναφέρθηκε με τον συντελεστή σε μορφή csd. Ακολουθούν η υλοποίηση και τα αποτελέσματα της προσομοίωσης της κάθε αρχιτεκτονικής. Ο σχολιασμός των αποτελεσμάτων θα γίνει συνολικά στο τέλος.

α. Αρχιτεκτονική βασισμένη στη συνέλιξη, direct μορφή

Η τοπολογία του κυκλώματος δίδεται στο σχήμα 2.14. Συνεπώς, παραλείποντας το διαχωριστή άρτιων-περιττών, έχουμε τον εξής γράφο:



Σχήμα 3.15 Γράφος του Direct Half-Rate του CDF 9/7

όπου, με M_X αναπαριστούμε τους πολλαπλασιαστές, με D_X τους καταχωρητές, με A_X τους αθροιστές, με S_X (δεν υπάρχουν στο παραπάνω σχήμα) τους αφαιρέτες, με I_X τις εισόδους και με O_X τις εξόδους. **Θα ακολουθήσουμε αυτή τη σύμβαση και στα επόμενα διαγράμματα.** Συνεπώς, στο κύκλωμα αντιστοιχεί το εξής αρχείο sf:

direct fir.sf

```

name dwt_inner;

output xc_2n 16;
output w_2n 16;

input xeven 16 0.9999999;
input xodd 16 0.9999999;

delay xodd xodd1;
delay xodd1 xodd2;
delay xodd2 xodd3;
delay xodd3 xodd4;
delay xeven xeven1;
delay xeven1 xeven2;
delay xeven2 xeven3;

add xodd xodd4 sum10;
add xodd1 xodd3 sum12;
add xodd xodd3 sumh0;
add xodd1 xodd2 sumh2;
add xeven xeven2 sumh1;
add xeven xeven3 sum11;
add xeven1 xeven2 sum13;

mult sum10 p_10 0.037828455507264;
mult sum12 p_12 -0.110624404418437;
mult xodd2 p_14 0.852698679008894;
mult sumh0 p_h0 0.06453888262870;
mult sumh2 p_h2 -0.41809227322162;

mult sumh1 p_h1 -0.04068941760916;
mult xeven1 p_h3 0.78848561640558;
mult sum11 p_l1 -0.0238494650195568;
mult sum13 p_l3 0.377402855612831;

add p_14 p_12 sum1;
add sum1 p_10 sum2;
add p_l1 p_l3 sum5;
add sum2 sum5 xc_2n;

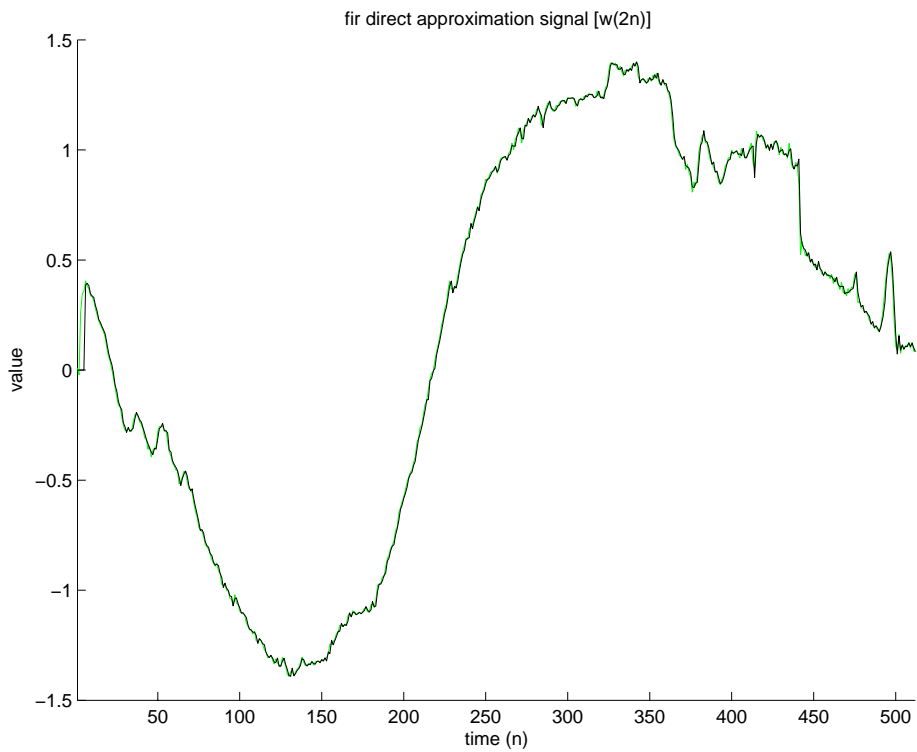
add p_h0 p_h2 sum3;
add p_h1 p_h3 sum4;
add sum3 sum4 w_2n;

```

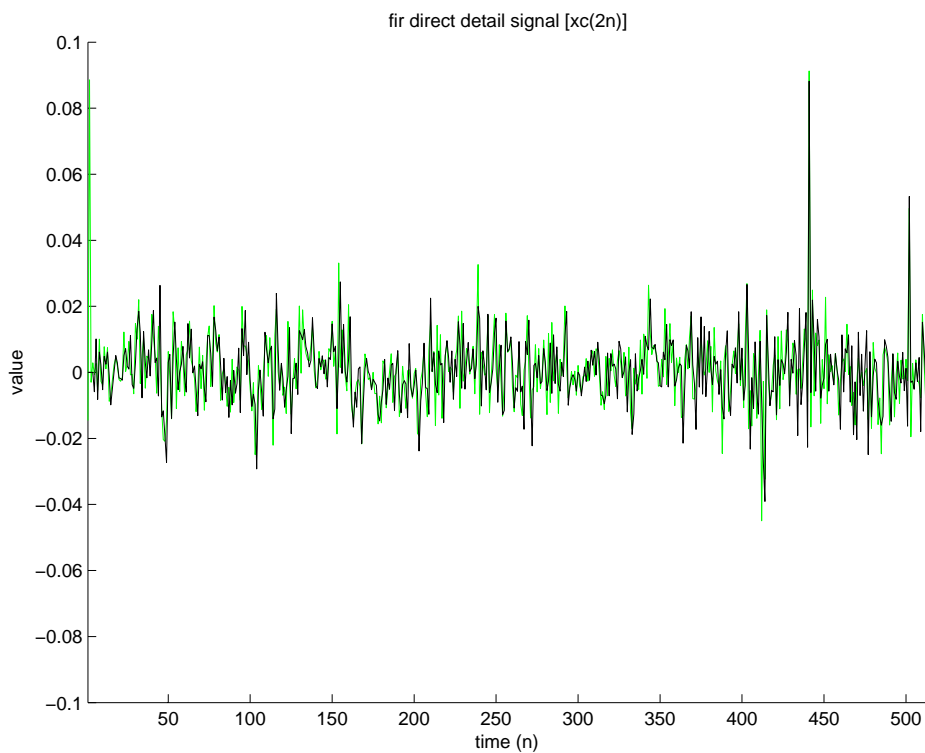
Το πρόγραμμα `sfc`, διαβάζοντας το αρχείο αυτό, μας υλοποιεί το φίλτρο με τα παρακάτω αποτελέσματα:

- Αριθμός πλήρων αθροιστών: **1015**
- Αριθμός πυλών NOT: **464**
- D-Flip-Flops: **112**
- Κρίσιμο μονοπάτι (σε πλήρεις αθροιστές): **37**
- Έξοδος σήματος προσέγγισης [$w(2n)$]: **αρχικό μήκος 17 bit, κλιμάκωση 1**
- Έξοδος σήματος λεπτομερειών [$x_c(2n)$]: **αρχικό μήκος 17 bit, κλιμάκωση 1**

Προσομοιώνοντας τη λειτουργία του κυκλώματος και συγκρίνοντας τα σήματα προσέγγισης και λεπτομερειών στις εξόδους του, με τα πρότυπα αντίστοιχα σήματα του `matlab`, παίρνω τις παρακάτω γραφικές παραστάσεις:



Σχήμα 3.16 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στη συνέλιξη, direct μορφή



Σχήμα 3.17 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στη συνέλιξη, direct μορφή

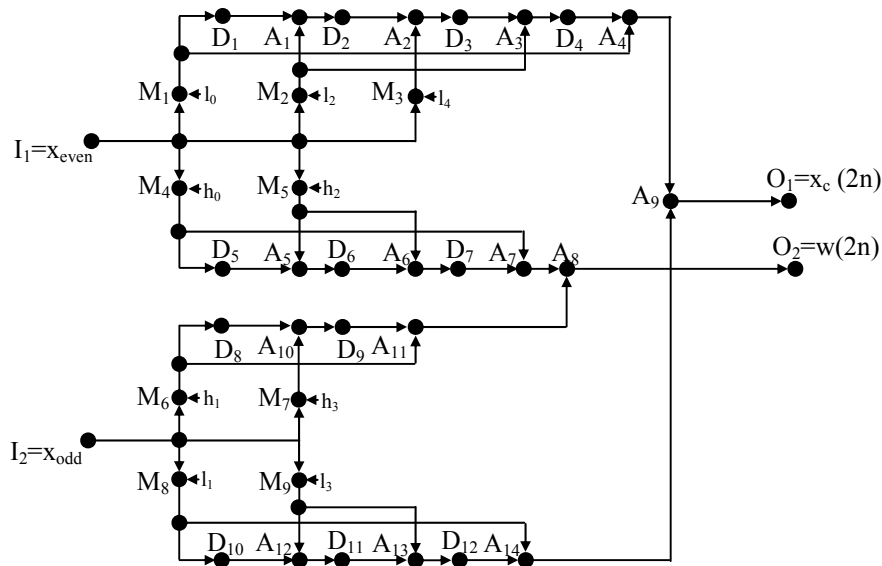
Στις προηγούμενες γραφικές παραστάσεις, με μαύρη γραμμή αναπαρίσταται το σήμα εξόδου του κυκλώματος, και με ανοιχτόχρωμη το «ιδανικό» αντίστοιχο σήμα

που μας δίνει το matlab. Η σύμβαση αυτή θα ακολουθηθεί και στις επόμενες γραφικές παραστάσεις. Επιπλέον, από το τελικό πρόγραμμα επεξεργασίας των δεδομένων έχουμε ότι:

- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος προσέγγισης $[w(2n)]$: **0.0127**
- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος λεπτομερειών $[x_c(2n)]$: **0.0069**

β. Αρχιτεκτονική βασισμένη στη συνέλιξη, transpose μορφή

Εδώ, το σχήμα 2.16 μας δίνει τον εξής γράφο:



Σχήμα 3.18 Γράφος του Transpose Half-Rate του CDF 9/7

ο οποίος αντιστοιχεί στο αρχείο .sf:

transpose_fir.sf

```
name dwt_inner;

output xc_2n 16;
output w_2n 16;

input xeven 16 0.9999999;
input xodd 16 0.9999999;

mult xodd p_10 0.037828455507264;
mult xodd p_12 -0.110624404418437;
mult xodd p_14 0.852698679008894;
mult xodd p_h0 0.06453888262870;
mult xodd p_h2 -0.41809227322162;

mult xeven p_h1 -0.04068941760916;
mult xeven p_h3 0.78848561640558;
mult xeven p_11 -0.0238494650195568;
mult xeven p_13 0.377402855612831;

delay p_10 p_10d;
add p_10d p_12 sum1;
delay sum1 sum1d;
add sum1d p_14 sum2;
delay sum2 sum2d;
add sum2d p_12 sum3;
delay sum3 sum3d;
add sum3d p_10 sum4;
```

```

delay p_l1 p_l1d;
add p_l1d p_l3 sum10;
delay sum10 sum10d;
add sum10d p_l3 sum11;
delay sum11 sum11d;
add sum11d p_l1 sum12;

add sum4 sum12 xc_2n;

delay p_h0 p_h0d;
add p_h0d p_h2 sum5;
delay sum5 sum5d;
add sum5d p_h2 sum6;
delay sum6 sum6d;
add sum6d p_h0 sum7;

delay p_h1 p_h1d;
add p_h1d p_h3 sum8;
delay sum8 sum8_d;
add sum8_d p_h1 sum9;

add sum7 sum9 w_2n;

```

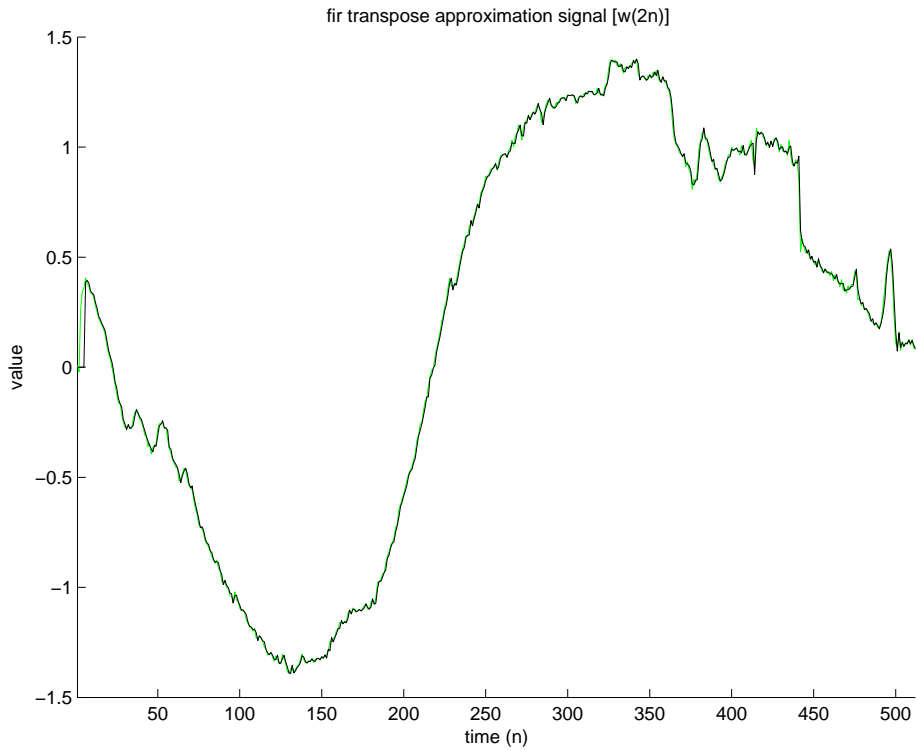
Το πρόγραμμα sfc μας δίδει, αφού εκτελέσει το παραπάνω αρχείο, ότι τα χαρακτηριστικά της αρχιτεκτονικής αυτής είναι:

- Αριθμός πλήρων αθροιστών: **961**
- Αριθμός πυλών NOT: **446**
- D-Flip-Flops: **190**
- Κρίσιμο μονοπάτι (σε πλήρεις αθροιστές): **37**
- Έξοδος σήματος προσέγγισης $[w(2n)]$: **αρχικό μήκος 17 bit, κλιμάκωση 1**
- Έξοδος σήματος λεπτομερειών $[x_c(2n)]$: **αρχικό μήκος 17 bit, κλιμάκωση 1**

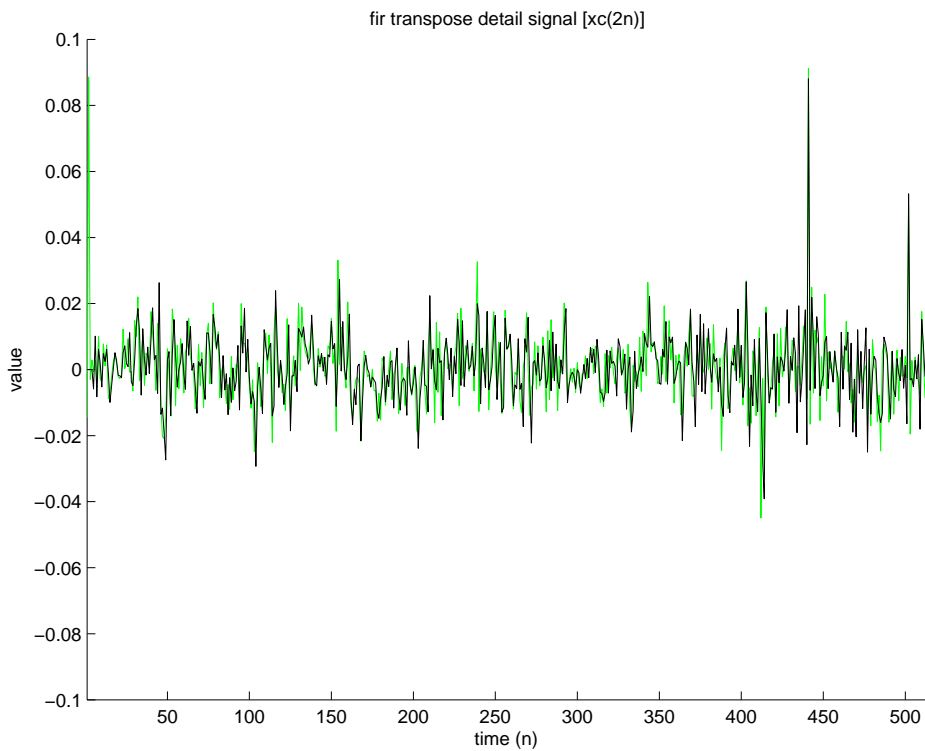
Τέλος, η προσομοίωση της λειτουργίας του κυκλώματος και η επεξεργασία των αποτελεσμάτων στο Matlab όπως πριν, μας δίδει τις εξής μέσες τιμές για τα σφάλματα εξόδου:

- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος προσέγγισης $[w(2n)]$: **0.0127**
- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος λεπτομερειών $[x_c(2n)]$: **0.0069**

ενώ, όσον αφορά τις εξόδους αυτές καθαυτές έχουμε τις ακόλουθες γραφικές παραστάσεις:



Σχήμα 3.19 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στη συνέλιξη, transpose μορφή



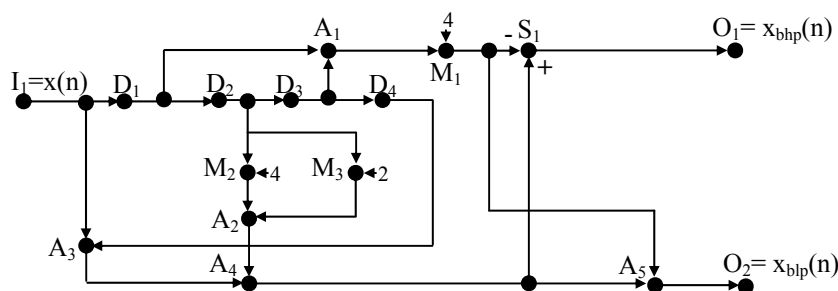
Σχήμα 3.20 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στη συνέλιξη, transpose μορφή

γ. Αρχιτεκτονική βασισμένη στη b-spline παραγοντοποίηση, direct b-spline τμήμα

Στην ενότητα Δ του κεφαλαίου 2 αποδείξαμε ότι σε όλες τις αρχιτεκτονικές που βασίζονται στην b-spline παραγοντοποίηση, η βέλτιστη πρακτική είναι να υλοποιήσουμε το καταναμημένο τμήμα (distributed part) σε transpose μορφή, δηλαδή με την αρχιτεκτονική του σχ. 2.22. Συνεπώς, τόσο σε αυτή την περίπτωση, όπου έχουμε το

b-spline τμήμα σε ευθεία μορφή, όσο και στην επόμενη, όπου το έχουμε σε transpose μορφή, το καταναμημένο τμήμα θα υλοποιηθεί σύμφωνα με το σχ. 2.22. Επειδή, όμως, γνωρίζουμε το μήκος λέξης και το εύρος τιμών **μόνο** για τις εισόδους του μετασχηματισμού, η ανάλυση θα ξεκινήσει υποχρεωτικά από το b-spline τμήμα

Το b-spline τμήμα για το μετασχηματισμό CDF 9/7 σε ευθεία μορφή, δίδεται από το σχήμα 2.23, το οποίο μας δίνει τον εξής γράφο:



Σχήμα 3.21 Γράφος της Direct υλοποίησης του b-spline τμήματος του CDF9/7

και ο οποίος κωδικοποιείται στο παρακάτω .sf αρχείο:

bp_direct.sf

```
name bspline_part;

output xbhp_n;
output xblp_n;

input x_n 16 0.9999999;

delay x_n x_n1;
delay x_n1 x_n2;
delay x_n2 x_n3;
delay x_n3 x_n4;

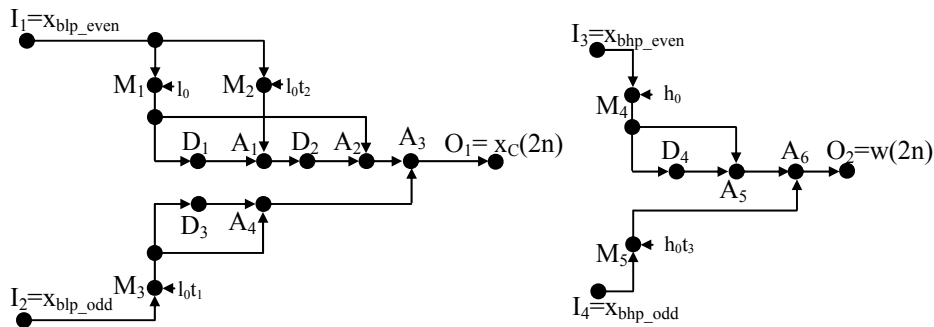
mult x_n2 shift4 4;
mult x_n2 shift2 2;
add shift4 shift2 sum1;
add x_n x_n4 sum2;
add sum2 sum1 sum3;
add x_n1 x_n3 sum4;
mult sum4 prod4 4;
add prod4 sum3 xblp_n;
sub prod4 sum3 xbhp_n;
```

Η εκτέλεση του παραπάνω στο sfc, μας δίνει τα εξής αποτελέσματα:

- Αριθμός πλήρων αθροιστών: **101**
- Αριθμός πυλών NOT: **33**
- D-Flip-Flops: **64**
- Κρίσιμο μονοπάτι (σε πλήρεις αθροιστές): **20**

- Έξοδος $[x_{bhp}(n)]$: μήκος 20 bit, κλιμάκωση 4, μέγιστη απόλυτη τιμή 15.9999984
- Έξοδος $[x_{blp}(n)]$: μήκος 20 bit, κλιμάκωση 4, μέγιστη απόλυτη τιμή 15.9999984

Συνεπώς, μπορούμε πλέον να προχωρήσουμε στην υλοποίηση του κατανεμημένου τμήματος. Η αρχιτεκτονική του δίδεται από το σχ. 2.22, το οποίο μας δίνει τον εξής γράφο:



Σχήμα 3.22 Γράφος του βέλτιστου κατανεμημένου τμήματος (transpose μορφή) του CDF 9/7

Ο παραπάνω γράφος, σε συνδυασμό με τα προηγούμενα αποτελέσματα για τα $x_{bhp}(n)$, $x_{blp}(n)$, μας δίνει το παρακάτω .sf αρχείο:

bspl dist.sf

```

name dist_part;

output xc_2n 16;
output w_2n 16;

input xbhp_even 20 15.9999984;
input xbhp_odd 20 15.9999984;
input xblp_even 20 15.9999984;
input xblp_odd 20 15.9999984;

mult xblp_even p10 -0.0244054551612903;
mult xblp_even p10t2 -0.234230965468;
mult xblp_odd p10t1 0.113008581549677;

mult xbhp_even ph0 -0.06453888263;
mult xbhp_odd ph0t3 -0.2174660884;

delay p10 p10d;
add p10d p10t2 sum1;
delay sum1 sum1d;
add sum1d p10 sum2;
delay p10t1 p10t1d;
add p10t1d p10t1 sum3;
add sum3 sum2 xc_2n;

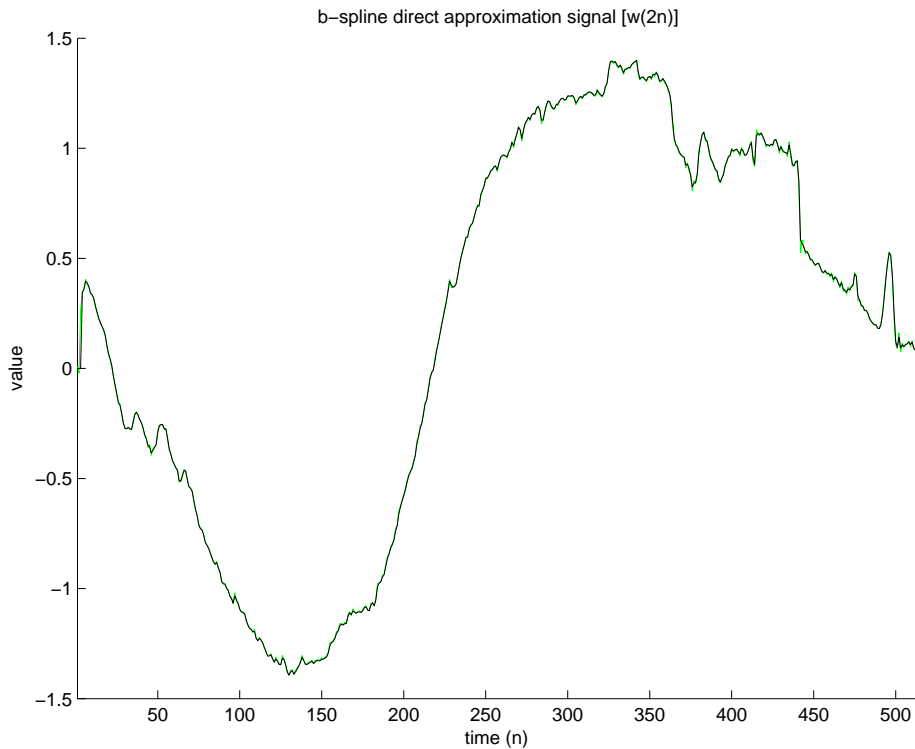
delay ph0 ph0d;
add ph0 ph0d sum4;
add sum4 ph0t3 w_2n;

```

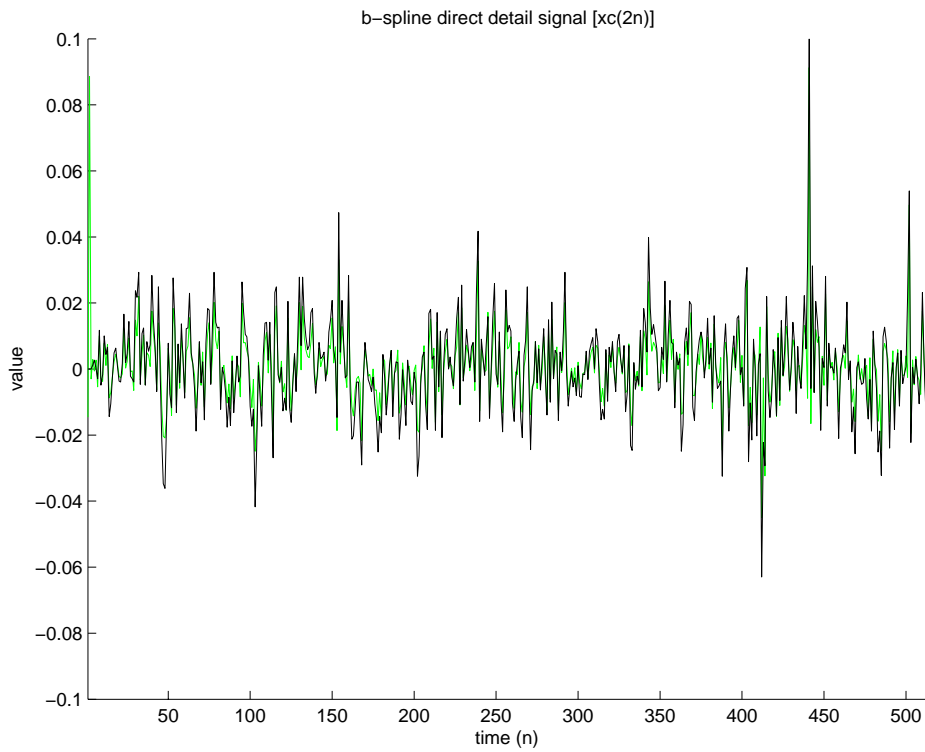
Η εκτέλεση του παραπάνω από το πρόγραμμα sfc μας δίνει τα εξής χαρακτηριστικά (μόνο για το κατανεμημένο τμήμα που περιγράφει ο κώδικας!):

- Αριθμός πλήρων αθροιστών: **656**
- Αριθμός πυλών NOT: **279**
- D-Flip-Flops: **79**
- Κρίσιμο μονοπάτι (σε πλήρεις αθροιστές): **43**
- Έξοδος σήματος προσέγγισης $[w(2n)]$: αρχικό μήκος 19 bit, κλιμάκωση 3

- Έξοδος σήματος λεπτομερειών $[x_c(2n)]$: **αρχικό μήκος 20 bit, κλιμάκωση 4**
- Στη συνέχεια, τροποποιήσαμε ελαφρά το testbench (έτσι ώστε να συνδέει κατάλληλα την έξοδο του b-spline τμήματος στο κατανεμημένο, από το οποίο διαβάζει και τις τιμές των εξόδων). Μετά την προσομοίωση της συνολικής λειτουργίας της αρχιτεκτονικής αυτής, η σύγκριση των αποτελεσμάτων μας έδωσε τις παρακάτω μέσες απόλυτες τιμές σφαλμάτων:
- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος προσέγγισης $[w(2n)]$: **0.0048**
 - Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος λεπτομερειών $[x_c(2n)]$: **0.0041**
- ενώ, έχουμε και τις ακόλουθες γραφικές παραστάσεις:



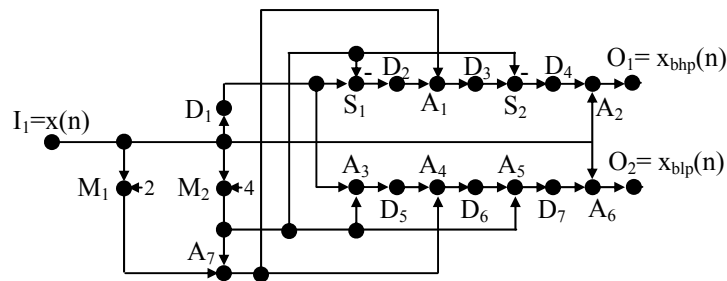
Σχήμα 3.23 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στην b-spline παραγοντοποίηση, με direct b-spline part



Σχήμα 3.24 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στην b-spline παραγοντοποίηση, με direct b-spline part

δ. Αρχιτεκτονική βασισμένη στη b-spline παραγοντοποίηση, transpose b-spline τμήμα

Όπως και πριν, και εδώ ξεκινάμε από το b-spline τμήμα. Η αρχιτεκτονική του b-spline τμήματος σε transpose μορφή δίδεται από το σχ. 2.24 το οποίο, κατόπιν μερικών τροποποιήσεων για εξοικονόμηση υλικού, μετασχηματίζεται στον παρακάτω γράφο:



Σχήμα 3.25 Γράφος της Transpose υλοποίησης του b-spline τμήματος του CDF9/7

Το .sf αρχείο που αντιστοιχεί στον παραπάνω γράφο είναι το εξής:

bp_transpose.sf

```
name bspline_part;

output xbhp_n;
output xblp_n;

input x_n 16 0.9999999;

mult x_n shift2 2;
mult x_n shift4 4;
add shift2 shift4 sum1;
delay x_n x_n1;

add x_n1 shift4 sum21;
delay sum21 sum211;
add sum211 sum1 sum31;
delay sum31 sum311;
add sum311 shift4 sum41;
delay sum41 sum411;
add sum411 x_n xblp_n;

sub shift4 x_n1 sum2h;
delay sum2h sum2h1;
add sum2h1 sum1 sum3h;
delay sum3h sum3h1;
sub shift4 sum3h1 sum4h;
delay sum4h sum4h1;
add sum4h1 x_n xbhp_n;
```

και η εκτέλεσή του με το sfc μας δίδει τα εξής:

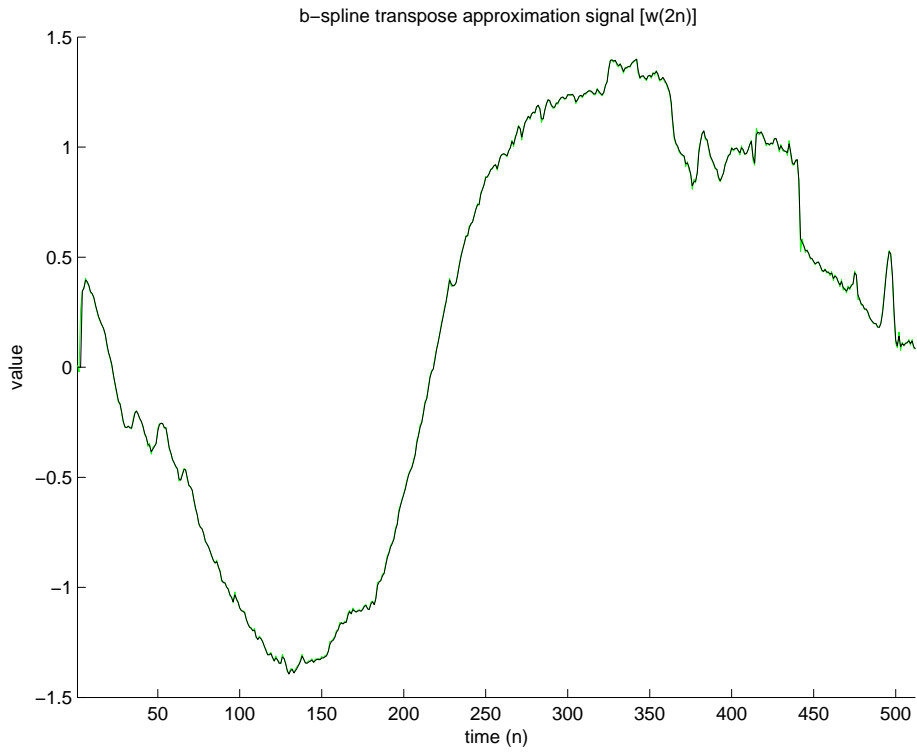
- Αριθμός πλήρων αθροιστών: **162**
- Αριθμός πυλών NOT: **57**
- D-Flip-Flops: **134**
- Κρίσιμο μονοπάτι (σε πλήρεις αθροιστές): **20**
- Έξοδος $[x_{bhp}(n)]$: **μήκος 20 bit, κλιμάκωση 4, μέγιστη απόλυτη τιμή 15.9999984**
- Έξοδος $[x_{blp}(n)]$: **μήκος 20 bit, κλιμάκωση 4, μέγιστη απόλυτη τιμή 15.9999984**

Παρατηρούμε άμεσα (εκτός από τις μεγαλύτερες απαιτήσεις σε υλικό απ'ότι πριν) ότι οι προδιαγραφές για τα σήματα $x_{bhp}(n)$, $x_{blp}(n)$ είναι οι ίδιες με την προηγούμενη αρχιτεκτονική, συνεπώς, μπορούμε να χρησιμοποιήσουμε ακριβώς το ίδιο καταναεμημένο τμήμα από πριν.

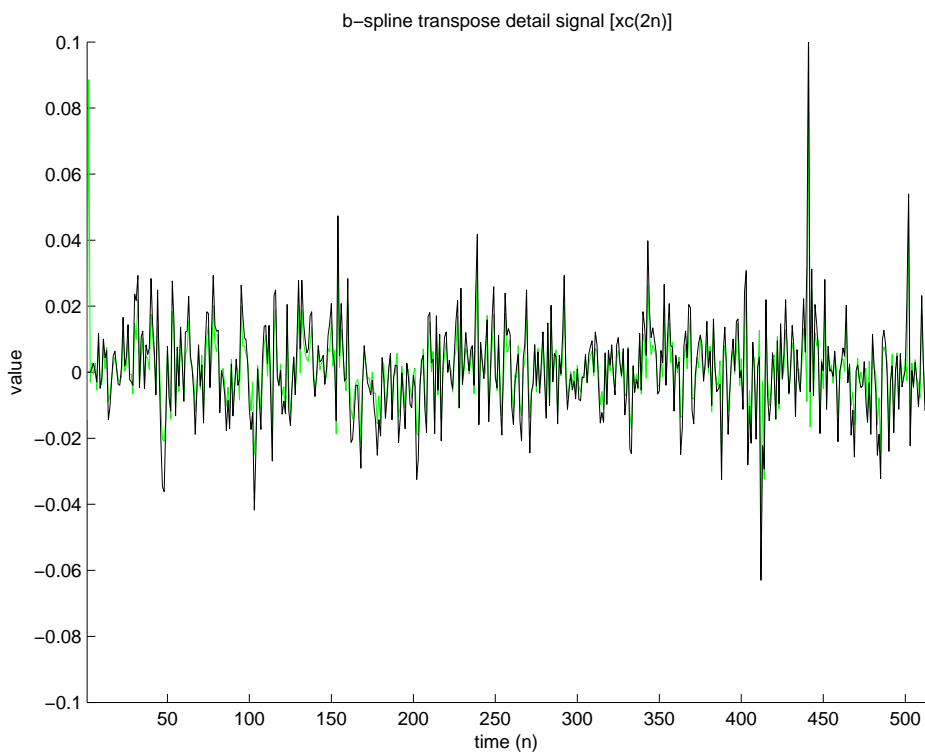
Έτσι, αλλάζοντας μόνο τον κώδικα του b-spline τμήματος, προσομοιώνουμε εκ νέου το συνολικό κύκλωμα, οπότε, έχουμε:

- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος προσέγγισης $[w(2n)]$: **0.0048**
- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος λεπτομερειών $[x_c(2n)]$: **0.0041**

Οι γραφικές παραστάσεις των σημάτων εξόδου εν συγκρίσει με τα «πρότυπα» σήματα εξόδου είναι οι παρακάτω:



Σχήμα 3.26 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στην b-spline παραγοντοποίηση, με transpose b-spline part



Σχήμα 3.27 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στην b-spline παραγοντοποίηση, με transpose b-spline part

ε. Αρχιτεκτονική βασισμένη στο lifting scheme

Στην ενότητα Ε του κεφαλαίου 2, κατά την ανάλυση του flipping structure, αναφέραμε ότι κατά τη μετακίνηση των πολλαπλασιαστών για τη συντόμευση του κρίσιμου μονοπατιού, δεν είμαστε υποχρεωμένοι να πολλαπλασιάσουμε τους παράλληλους κλάδους με τον αντίστροφο του συντελεστή (βλ. σχ. 2.28 a-c) αλλά με οποιοδήποτε γινόμενο του τελευταίου επί μια δύναμη του 2, αφού οι διαιρέσεις που θα γίνουν στον κλάδο όπου βρίσκεται ο αρχικός πολλαπλασιαστής, θα γίνουν χωρίς κόστος.

Στο [C6] αποδεικνύεται ότι πετυχαίνεται ελαχιστοποίηση του σφάλματος υπολογισμού, αν κάθε μία από τις «περιοχές διαίρεσης» στο σχ. 2.28b πολλαπλασιαστεί με τον εξής παράγοντα:

- Η πρώτη πολλαπλασιάζεται πάλι με $1/a = -0.630463621$ (3.22a)

- Η δεύτερη (από αριστερά προς τα δεξιά) με $1/16b$. Τελικά, $1/16ab = 0.743750255$ (3.22b)

- Η τρίτη με $1/2c$. Τελικά, $1/32bc = -0.668067178$ (3.22c)

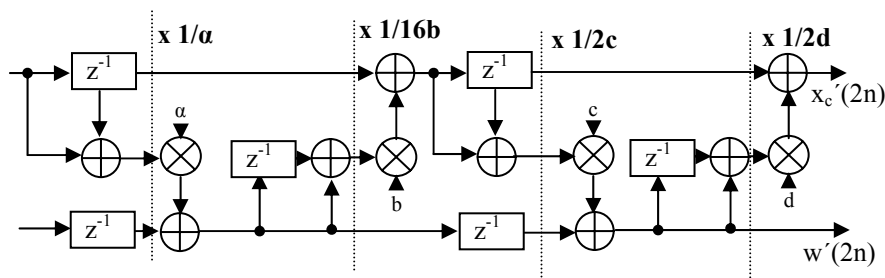
- Η τέταρτη με $1/2d$. Τελικά, $1/4cd = 0.638443853$ (3.22d)

Εξ' αιτίας αυτών των αλλαγών θα πρέπει να αλλάξουν και οι πολλαπλασιαστές κανονικοποίησης στο τέλος:

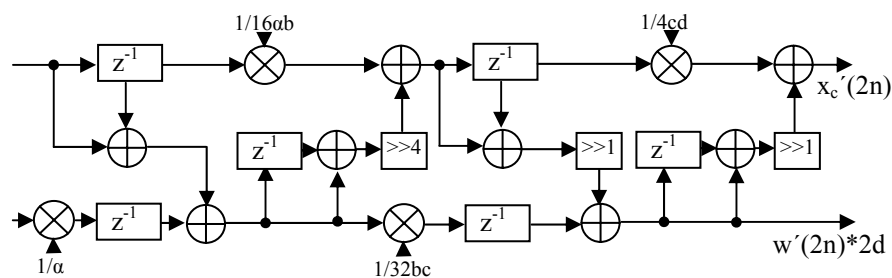
- Ο παράγοντας του πολλαπλασιαστή κανονικοποίησης της εξομαλυμένης μορφής του σήματος (x_c) θα γίνει $64Kabcd = 2.421021123$ (3.22e)

- Αντίστοιχα, ο παράγοντας πολλαπλασιαστή κανονικοποίησης του σήματος λεπτομερειών θα γίνει 2.065244222 (3.22f)

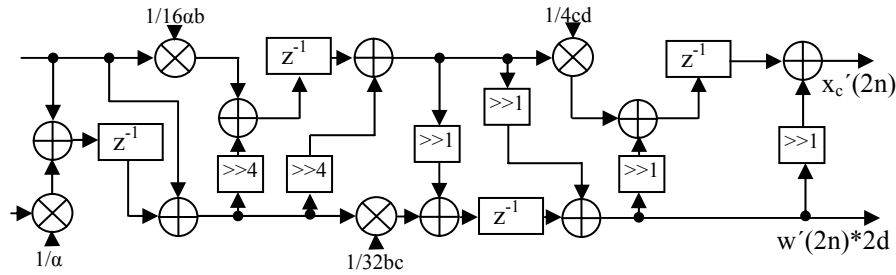
Προφανώς, λόγω των ολισθήσεων που θα πρέπει να συμβούν, δεν μπορούμε να αντικαταστήσουμε απ' ευθείας στο σχ. 2.28e τα 3.22 και να λάβουμε το σωστό αποτέλεσμα. Έτσι, ξεκινώντας από το σχ. 2.28b έχουμε (χωρίς τους πολλαπλασιαστές κανονικοποίησης και το κύκλωμα διαχωρισμού αρτίων-περιττών για λόγους ευκρίνειας):



Σχήμα 3.28a Καθορισμός «περιοχών διαίρεσης»

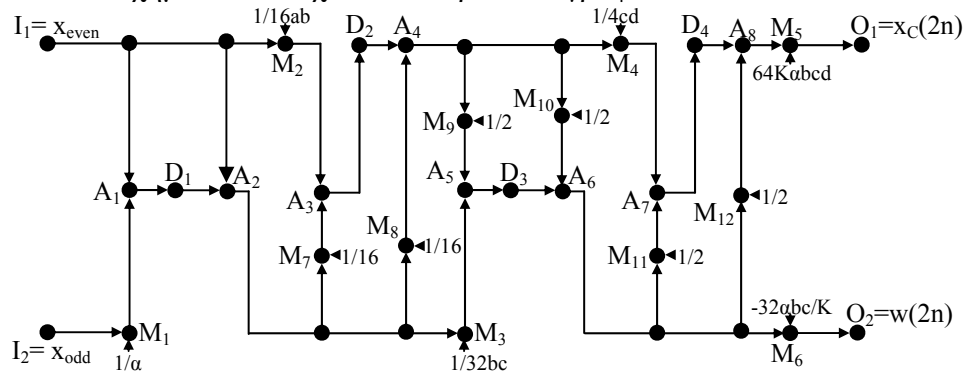


Σχήμα 3.28b Εκτέλεση πράξεων, συγχώνευση πολλαπλασιαστών, μετακίνηση καταχωρητών



Σχήμα 3.28c Τελική μορφή με λιγότερους καταχωρητές και μικρότερο critical path με αλλαγή της σειράς των αθροίσεων

Το τελευταίο σχήμα αντιστοιχεί στον παρακάτω γράφο:



Σχήμα 3.29 Γράφος της αρχιτεκτονικής βασισμένης στο Lifting Scheme

ο οποίος, μαζί με τις (3.22) δίνει το παρακάτω αρχείο .sf:

lifting_scheme.sf

```

name dwt_inner;

output xc_2n 16;
output w_2n 16;

input xeven 16 0.9999999;
input xodd 16 0.9999999;

mult xodd prod1 -0.630463621;
add prod1 xeven sum1;
delay sum1 n1;
add xeven n1 sum2;
mult xeven prod2 0.743750255;
mult sum2 prod3 0.0625;
mult sum2 prod4 0.0625;
add prod2 prod3 sum3;
delay sum3 n3;
add n3 prod4 sum4;
mult sum2 prod6 -0.668067178;
mult sum4 prod5 0.5;
mult sum4 prod10 0.5;
mult sum4 prod9 0.638443853;
add prod6 prod5 sum5;
delay sum5 n5;
add n5 prod10 sum6;
mult sum6 prod7 0.5;
mult sum6 prod8 0.5;
mult sum6 w_2n -2.065244222;
add prod7 prod9 sum7;
delay sum7 n7;
add n7 prod8 sum8;
mult sum8 xc_2n 2.421021123;

```

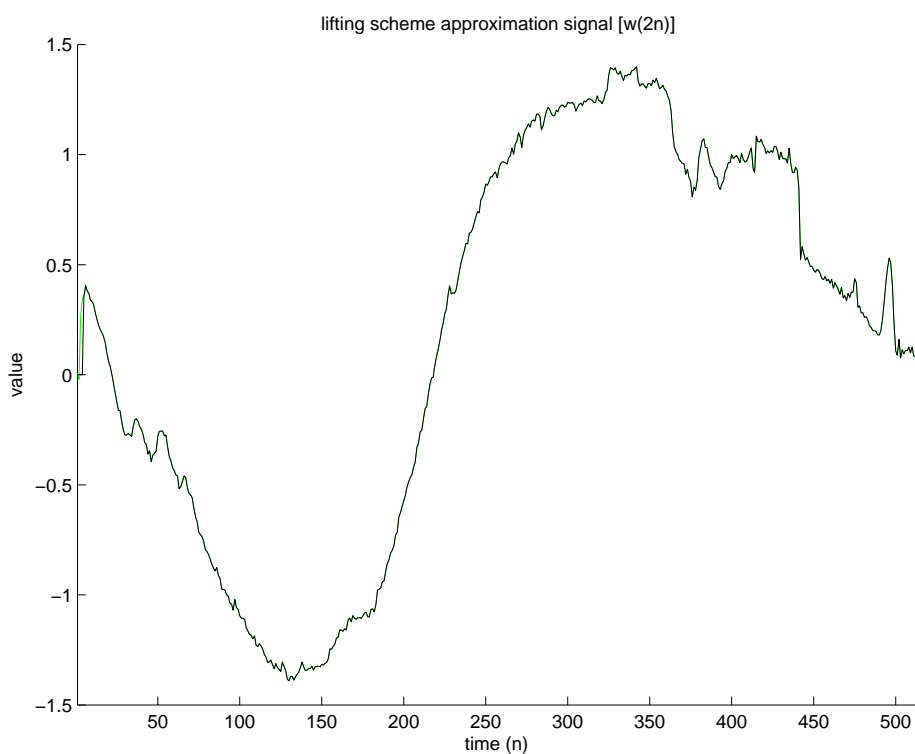
Η αρχιτεκτονική που προκύπτει από την εκτέλεση του παραπάνω αρχείου από το sfc έχει τα παρακάτω χαρακτηριστικά:

- Αριθμός πλήρων αθροιστών: **701**
- Αριθμός πυλών NOT: **349**
- D-Flip-Flops: **69**
- Κρίσιμο μονοπάτι (σε πλήρεις αθροιστές): **39**
- Έξοδος σήματος προσέγγισης $[w(2n)]$: **αρχικό μήκος 18 bit, κλιμάκωση 4**
- Έξοδος σήματος λεπτομερειών $[x_c(2n)]$: **αρχικό μήκος 18 bit, κλιμάκωση 4**

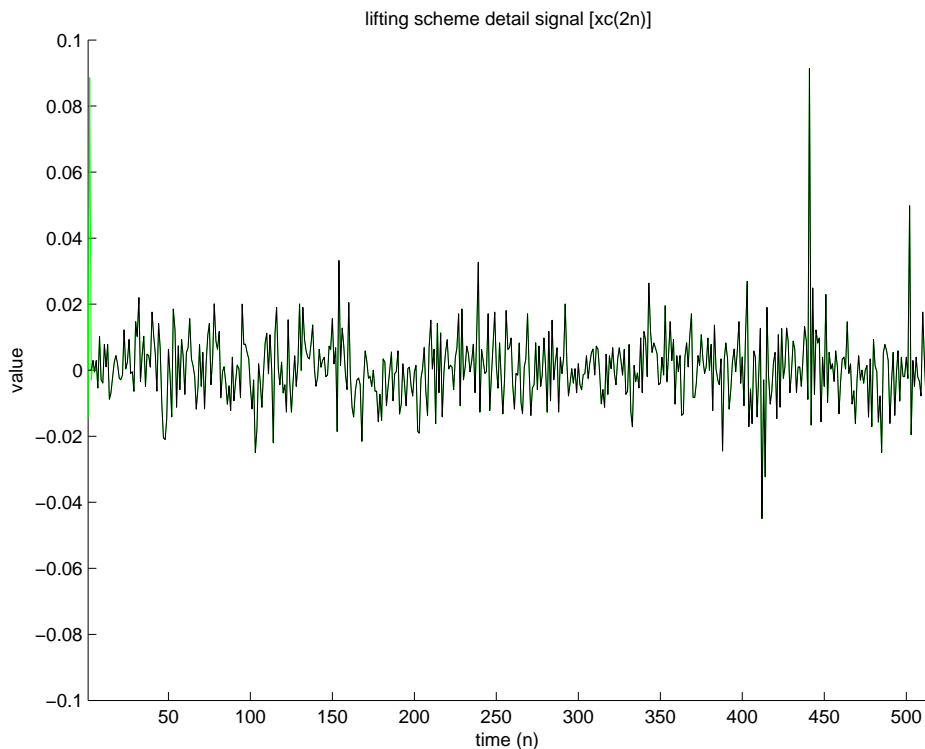
Τέλος, χρησιμοποιώντας το ίδιο ακριβώς testbench με τα κυκλώματα που βασίζονται στη συνέλιξη, έχω για τα σήματα εξόδου:

- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος προσέγγισης $[w(2n)]$: **0.0006**
- Μέση τιμή απόλυτου σφάλματος υπολογισμού του σήματος λεπτομερειών $[x_c(2n)]$: **0.0001**

Η εξέλιξη των σημάτων αυτών στο χρόνο εν συγκρίσει με τα «πρότυπα» σήματα εξόδου δίδεται στις παρακάτω γραφικές παραστάσεις:



Σχήμα 3.30 Σήμα προσέγγισης της αρχιτεκτονικής βασισμένης στο lifting scheme



Σχήμα 3.31 Σήμα λεπτομερειών της αρχιτεκτονικής βασισμένης στο lifting scheme

4. Συμπεράσματα

α. Γενικά

Προτού προχωρήσουμε στο σχολιασμό και τη σύγκριση των αποτελεσμάτων, θα παρατηρήσουμε ότι διαπιστώνουμε εκ του αποτελέσματος και «οπτικά» ότι τα αποτελέσματα των δύο αρχιτεκτονικών βασισμένων στη συνέλιξη, όπως και μεταξύ των δύο αρχιτεκτονικών που βασίζονται στη b-spline παραγοντοποίηση είναι (ανά δύο) **ακριβώς τα ίδια**. Πράγματι, μια εξέταση των αποτελεσμάτων με το matlab μας δείχνει ότι έχουμε για κάθε οικογένεια αρχιτεκτονικών την ίδια επακριβώς έξοδο τόσο για το σήμα των λεπτομερειών, όσο και για το σήμα προσέγγισης.

Ο λόγος που αυτό συμβαίνει είναι διότι (βλ. και προηγούμενη ενότητα) το μόνο στοιχείο του κυκλώματος που εισάγει εκούσια σφάλματα υπολογισμού είναι οι πολλαπλασιαστές: εμπειρικά αποτελέσματα έδειξαν ότι τα παραγόμενα γινόμενα από έναν πολλαπλασιαστή δυαδικού δένδρου αθροιστών/αφαιρετών carry-save έχουν μια απόκλιση από τα αναμενόμενα της τάξης των δύο λιγότερο σημαντικών bits. Επομένως, από τη στιγμή που το σφάλμα εισάγεται από τους πολλαπλασιαστές (και μόνο), με όποιον τρόπο και αν αυτοί λάβουν τις εισόδους τους (η διαφορά transpose-direct μορφής) το σφάλμα στην έξοδο θα είναι πάντα το ίδιο.

Συνεπώς, στον παρακάτω σχολιασμό θα περιοριστούμε στην αναφορά/σύγκριση μόνο των σφαλμάτων από τις “direct” αρχιτεκτονικές που βασίζονται στη συνέλιξη και την b-spline παραγοντοποίηση, ξέροντας ότι τα αποτελέσματα από τις δύο παραλειπούμενες είναι ακριβώς τα ίδια.

β. Σχολιασμός των σφαλμάτων

Μια «οπτική» παρατήρηση στα αποτελέσματα των μετασχηματισμών που βασίζονται στη συνέλιξη και την b-spline παραγοντοποίηση μας δείχνει ότι, ενώ το αποτέλεσμα στο σήμα προσέγγισης $[w(2n)]$ είναι αρκετά κοντά στη σωστή τιμή, στο σήμα λεπτομερειών $[x_c(2n)]$ υπάρχει μια αρκετά μεγαλύτερη απόκλιση. Στην πραγματικότητα, όπως φαίνεται και από τις μέσες απόλυτες τιμές σφάλματος, το σφάλμα στη λεπτομέρεια είναι **μικρότερο** από το σφάλμα στην προσέγγιση, για όλες τις αρχιτεκτονικές. Ωστόσο, επειδή το σήμα λεπτομερειών λαμβάνει αρκετά μικρότερες τιμές σε αυτή την περίπτωση από το σήμα προσέγγισης (το οποίο, φασματικά, σημαίνει ότι το leleccum έχει το μεγαλύτερο τμήμα της ενέργειάς του στις χαμηλότερες συχνότητες), το σφάλμα υπολογισμού γίνεται πιο εμφανές στην περίπτωση αυτή και καταστρέφει, σχεδόν, το σήμα λεπτομερειών. Ωστόσο αυτό δεν είναι απόλυτο: αν το σήμα είχε μεγαλύτερο φασματικό περιεχόμενο στις υψηλές συχνότητες, το σήμα λεπτομερειών θα αναδεικνυόταν πολύ καλύτερα, αν και με παρόμοιας τάξης σφάλματα στον υπολογισμό του.

Όπως αναφέρθηκε και πριν, μοναδικός (σχεδόν) ένοχος για τα σημαντικά αυτά σφάλματα είναι οι πολλαπλασιαστές δένδρου σταθερών αθροιστών και το σφάλμα υπολογισμού που εισάγουν λόγω αποκοπής στα δεδομένα εισόδου των σταθερού μήκους αθροιστών/αφαιρετών που τους αποτελούν. Το μικρότερο σφάλμα που εμφανίζουν οι αρχιτεκτονικές βασισμένες στη b-spline παραγοντοποίηση, έναντι αυτών που βασίζονται στη συνέλιξη, οφείλεται αφενός μεν στους λιγότερους πολλαπλασιαστές που περιέχουν (5 αντί 9), και συνεπώς στη μικρότερη διακύμανση του σφάλματος εξόδου, αφετέρου δε στη μεγαλύτερη αποκοπή που παρατηρείται στο σήμα εξόδου: ενώ στη συνέλιξη αποκόπτεται μόλις 1 bit, στην b-spline παραγοντοποίηση αποκόπτονται 3 ή 4 (3 στο σήμα προσέγγισης, 4 στο σήμα λεπτομερειών), συνεπώς, από τη στιγμή που τα τελευταία bits περιέχουν κατά κύριο λόγο τα σφάλματα υπολογισμού, αυτά χάνονται.

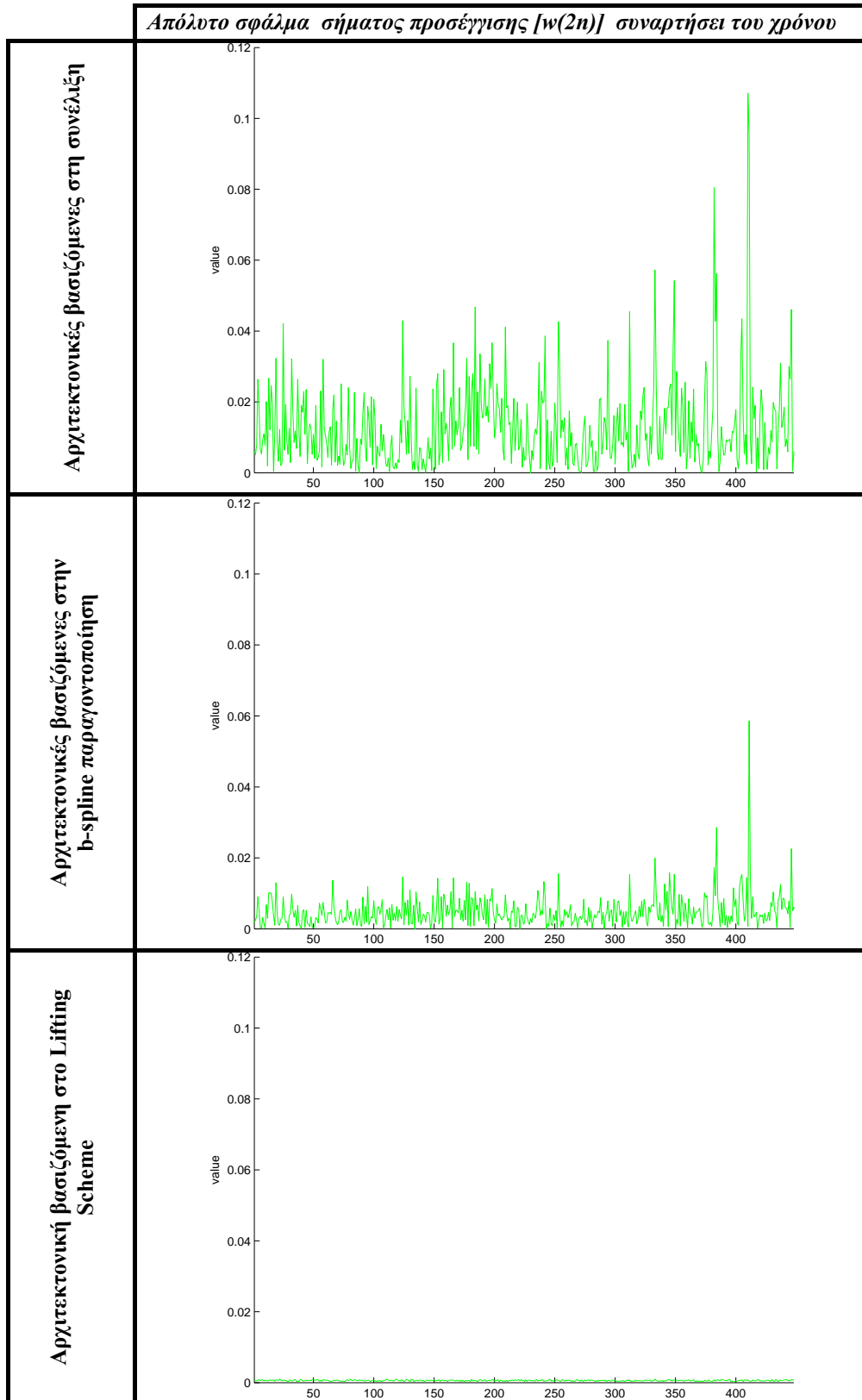
Στην ουσία, αυτό που έχει σημασία είναι ότι οι πολλαπλασιαστές δρουν σε δεδομένα 20 bit αντί σε 16bit που δρουν οι αντίστοιχοι στη συνέλιξη. Επειδή το σφάλμα παρουσιάζεται στα τελευταία, κατά κύριο λόγο, bits και εμπειρικά αποδείχθηκε ότι η εξάπλωσή του είναι σχεδόν αναισθητή στο μήκος λέξης των δεδομένων εισόδου, αυτό σημαίνει ότι μπορεί να αποκοπεί, στη συνέχεια, χωρίς να επηρεάζει τόσο το σήμα εξόδου.

Παρά όλα αυτά, τόσο οι αρχιτεκτονικές βασισμένες στη b-spline παραγοντοποίηση, όσο και αυτές βασισμένες στη συνέλιξη παρουσιάζουν **απαράδεκτα χαμηλή ακρίβεια**. Η μικρότερη μέση απόλυτη τιμή σφάλματος που παρατηρείται και στις τέσσερις είναι 0.0041 στο σήμα λεπτομερειών στις b-spline αρχιτεκτονικές. Αυτό σημαίνει, αν αναμένουμε μέση τιμή σήματος εξόδου κοντά στο 0.5, ότι κατά μέσο όρο το 1% του σήματος θα είναι σφάλμα. Επιπλέον, το «κατά μέσο όρο» είναι μια μάλλον αισιόδοξη ανάλυση: όπως είδαμε και πριν στο σήμα λεπτομερειών, το παραγόμενο σφάλμα είναι ικανό να αποκρύψει εντελώς τη φασματική πληροφορία. Αυτό δεν είναι επιτρεπτό για εξειδικευμένα κυκλώματα που υλοποιούνται με πολλαπλασιαστές 16-bit για σχεδόν όλες τις πιθανές εφαρμογές.

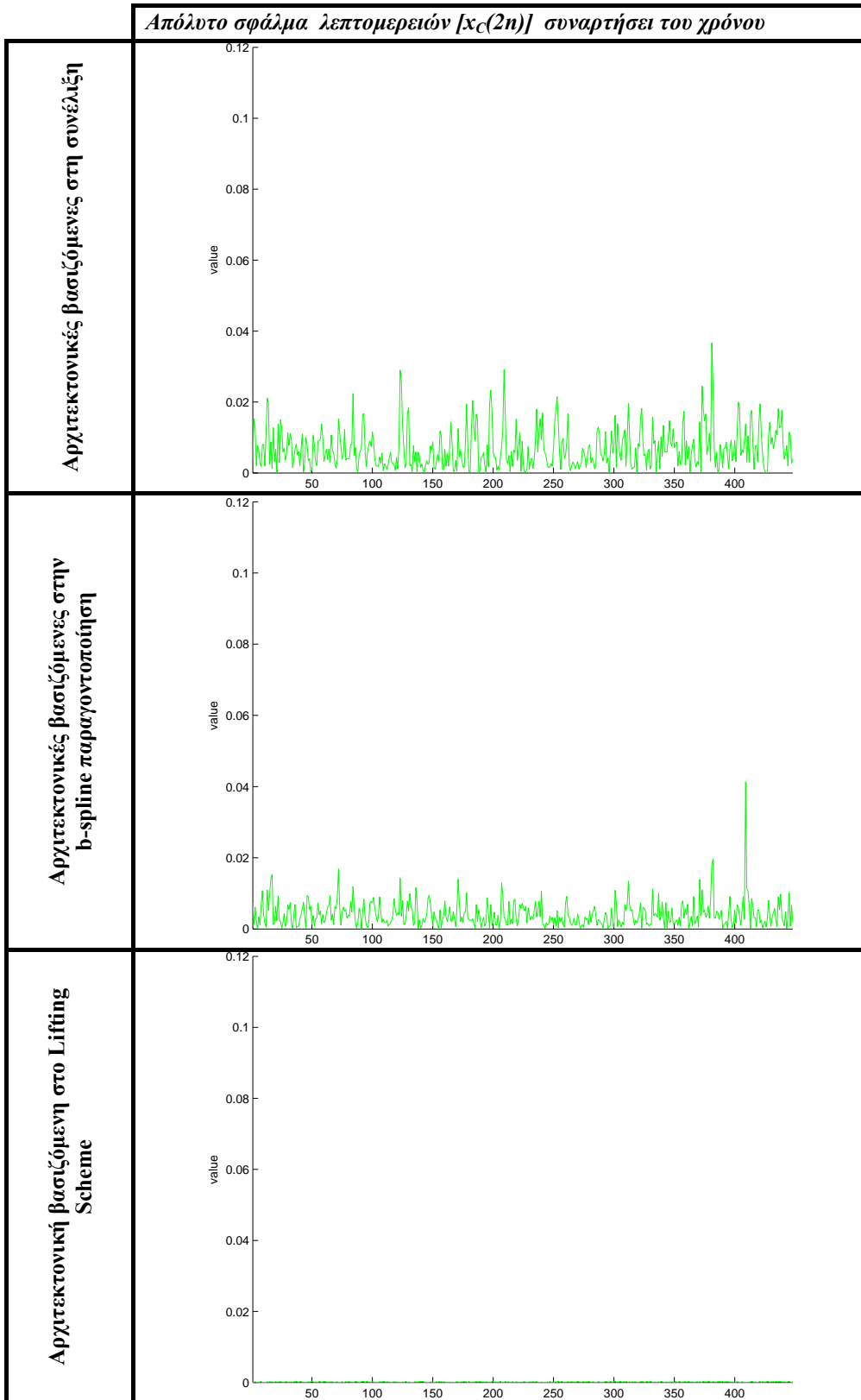
Αυτό που προκαλεί μεγάλη εντύπωση είναι το εξαιρετικά χαμηλό σφάλμα που εμφανίζουν τα σήματα προσέγγισης και λεπτομερειών που προκύπτουν από το μετασχηματισμό που βασίζεται στο lifting scheme: τα αποτελέσματα είναι εξαιρετικά, εποπτικά, ενώ το μέσο απόλυτο σφάλμα είναι 0.0006 για το σήμα προσέγγισης και 0.0001 για το σήμα λεπτομερειών: *τουλάχιστον μια τάξη μεγέθους χαμηλότερα από αυτά των άλλων αρχιτεκτονικών.*

Το ζητούμενο εδώ είναι η εξήγηση αυτού του φαινομένου: δεν εισάγουν σφάλματα οι πολλαπλασιαστές εδώ; Ή οι αριθμοί που χρησιμοποιούνται ως συντελεστές είναι τέτοιοι ώστε να μην δημιουργείται σφάλμα; Η εξήγηση βρίσκεται στο σχ. 3.29 και στην ανάλυση και την προσπάθεια των Huang et. al. στο [C6] να βρουν τους βέλτιστους *αντίστροφους* συντελεστές. Παρατηρούμε ότι η έξοδος σχεδόν όλων των πολλαπλασιαστών, που δεν είναι απλές ολισθήσεις, πριν προστεθεί είτε ολισθαίνεται προς τα δεξιά κατά μία τουλάχιστο θέση, είτε πολλαπλασιάζεται με κάποιον άλλον συντελεστή μεταξύ 0 και 1. Παρατηρούμε ότι σε κάθε περίπτωση, οδηγούμαστε σε αποκοπή ή σε «υποβάθμιση» της αξίας τουλάχιστον του λιγότερου σημαντικού bit του αποτελέσματος του πολλαπλασιαστή: ωστόσο, όπως αναφέραμε πριν, αυτό περιέχει ένα σημαντικό μέρος του σφάλματος υπολογισμού. Έτσι, ελαχιστοποιείται ή εξαλείφεται η διάδοση των σφαλμάτων μέσα στο κύκλωμα. Επιπλέον, στους δύο τελευταίους πολλαπλασιαστές κανονικοποίησης γίνεται αποκοπή των δύο λιγότερο σημαντικών bits στις εξόδους τους, πάλι δηλαδή, εξαλείφουμε το σφάλμα υπολογισμού. Συνεπώς, το lifting scheme είναι η μόνη αρχιτεκτονική όπου η χρήση πολλαπλασιαστών τέτοιου είδους κρίνεται αποδεκτή. Στις επόμενες δύο σελίδες δίνονται συγκριτικοί πίνακες του απόλυτου σφάλματος υπολογισμού των εξόδων όλων των κυκλωμάτων, σε συνάρτηση με το χρόνο. Για αποφυγή των μεταβατικών φαινομένων, έχουμε παραλείψει τα πρώτα και τα τελευταία 32 δείγματα του κάθε σήματος

Πίνακας 3.3 Σύγκριση απόλυτου σφάλματος σήματος προσέγγισης όλων των κυκλωμάτων



Πίνακας 3.4 Σύγκριση απόλυτου σφάλματος σήματος λεπτομερειών όλων των κυκλωμάτων



γ. Σχολιασμός των επιδόσεων

Πίνακας 3.5 Σύγκριση επιδόσεων σε υλικό, ταχύτητα και ακρίβεια

Αρχιτεκτονική	Πλήρεις Αθροιστές	Πύλες NOT	D Flip-Flop	Κρίσιμο μονοπάτι	Μέσο σφάλμα $w(n)$	Μέσο σφάλμα $x_c(n)$
Συνέλιξη, Direct	1015	464	112	37	0.0127	0.0069
Συνέλιξη, Trans.	961	446	190	37	0.0127	0.0069
b-spline, Direct	757	312	143	43	0.0048	0.0041
<i>b-spline τμήμα</i>	<i>101</i>	<i>33</i>	<i>64</i>	<i>20</i>	-	-
<i>καταν. τμήμα</i>	<i>656</i>	<i>279</i>	<i>79</i>	<i>43</i>		
b-spline, Trans.	818	336	213	43	0.0048	0.0041
<i>b-spline τμήμα</i>	<i>162</i>	<i>57</i>	<i>134</i>	<i>20</i>	-	-
<i>καταν. τμήμα</i>	<i>656</i>	<i>279</i>	<i>79</i>	<i>43</i>		
lifting scheme	701	349	69	39	0.0006	0.0001

Παραπάνω δίνουμε συνοπτικά τα χαρακτηριστικά της κάθε μίας αρχιτεκτονικής που υλοποιήσαμε σε επίπεδο πύλης όσον αφορά τις απαιτήσεις σε υλικό, το κρίσιμο μονοπάτι (σε αριθμό πλήρων αθροιστών στη σειρά, αγνοώντας την καθυστέρηση που εμφανίζουν οι πύλες NOT) και την ακρίβεια των αποτελεσμάτων.

Όσον αφορά τις απαιτήσεις σε υλικό, παρατηρούμε, ότι το κύκλωμα που βασίζεται στο lifting scheme, παρά τον μεγαλύτερο αριθμό πολλαπλασιαστών που απαιτεί σε σχέση με αυτά που βασίζονται στη b-spline παραγοντοποίηση, έχει τις μικρότερες από όλα. Ο λόγος είναι διπλός: πρώτον, το b-spline τμήμα κάθε αρχιτεκτονικής βασισμένης στην b-spline παραγοντοποίηση εισάγει μεγάλο αριθμό αθροιστών/αφαιρετών (101 πλήρεις αθροιστές είναι σχεδόν όσοι περιέχονται σε έναν σταθερό πολλαπλασιαστή 16x16). Επιπλέον, λόγω του μεγάλου μήκους λέξης των εξόδων του κατανεμημένου τμήματος στις b-spline αρχιτεκτονικές, όλοι οι πολλαπλασιαστές σε αυτές χειρίζονται δεδομένα 20-bit, έναντι 18 που είναι το μέγιστο για τους πολλαπλασιαστές της lifting-βασισμένης αρχιτεκτονικής.

Το μεγάλο μήκος λέξης στα κατανεμημένα τμήματα των b-spline based αρχιτεκτονικών έχει και άλλες δυσμενείς συνέπειες: εισάγει αφενός μεν πολύ μεγάλο αριθμό D flip-flop σε κάθε καταχωρητή (και, άρα, κάνει τις αρχιτεκτονικές αυτές τις πιο απαιτητικές σε μνήμη, άρα εντελώς ακατάλληλες για low-power εφαρμογές), αφετέρου, δε, αυξάνει και το κρίσιμο μονοπάτι στους αθροιστές/πολλαπλασιαστές.

Ένα ερώτημα που τίθεται είναι εν τέλει πόσο σημαντική είναι η βελτίωση που προτάθηκε στο σχ. 2.28e σε αυτή την εργασία για μείωση του αριθμού των καταχωρητών της lifting-βασισμένης αρχιτεκτονικής. Με δεδομένο ότι αυτοί οι δύο καταχωρητές θα είχαν κόστος τουλάχιστον 32 D Flip-Flop, βλέπουμε ότι εκτός από αύξηση στον αριθμό τους κατά 50%, το κύκλωμα αυτό θα πλησίαζε πολύ την direct αρχιτεκτονική που βασίζεται στη συνέλιξη σε απαιτήσεις μνήμης, ενώ έτσι αποκτά ένα καθαρό πλεονέκτημα.

Όσον αφορά το κρίσιμο μονοπάτι, παρατηρούμε ότι η αρχιτεκτονική που βασίζεται στο lifting scheme είναι μεν πιο «αργή» από αυτές που βασίζονται στη συνέλιξη, αλλά το κρίσιμο μονοπάτι της είναι μόλις κατά δύο πλήρεις αθροιστές πιο μεγάλο και επιπλέον είναι πολύ μικρότερο από αυτό του κατανεμημένου τμήματος των b-spline αρχιτεκτονικών, που «πληρώνουν» το μεγάλο μήκος λέξης των δεδομένων τους. Παρατηρούμε επίσης ότι οι transpose αρχιτεκτονικές δεν δίνουν κανένα ουσιαστικό όφελος σε ταχύτητα, παρά τις αυξημένες απαιτήσεις τους σε υλικό. Αυτό οφείλεται αφενός μεν στο μικρό αριθμό των σημείων των φίλτρων, αφετέρου, δε, στο ότι στην transpose μορφή αθροίζονται περισσότερα γινόμενα, δηλαδή, περισσότερα σήματα που απαιτούν ολίσθηση και επέκταση προσήμου πριν

αθροιστούν, απ'ότι στην direct. Έτσι, οι ολισθήσεις (υπενθυμίζουμε εδώ ότι οι συντελεστές στις αρχιτεκτονικές που βασίζονται στη συνέλιξη είναι όλοι κάτω του 1, μερικοί δε έχουν και κλιμάκωση -5) των προσθετών έχουν μεγαλύτερη επιβάρυνση στο κρίσιμο μονοπάτι απ'ότι μερικοί επιπλέον παράλληλοι αθροιστές στη σειρά.

Τέλος, όσον αφορά την κατανάλωση ισχύος, μόνο εντελώς γενικόλογες εικασίες μπορούν να γίνουν, καθώς η υλοποίηση δεν αφορά κάποια συγκεκριμένη τεχνολογία παρά γενικές, αφαιρετικές, πύλες και καταχωρητές. Ωστόσο, πιθανότατα και εδώ η αρχιτεκτονική που βασίζεται στο Lifting Scheme θα είναι η αποδοτικότερη, αφού όλα τα στοιχεία της δουλεύουν σε μισό ρυθμό ρολογιού (σε αντίθεση π.χ. με τις b-spline, όπου το b-spline τμήμα δουλεύει σε πλήρη ρυθμό ρολογιού) και περιέχει τα λιγότερα στοιχεία.

δ. Συμπεράσματα-Προτάσεις για περαιτέρω έρευνα

Το τελικό συμπέρασμα από τη διπλωματική αυτή εργασία είναι ότι για την υλοποίηση του ευθύ διακριτού μετασχηματισμού wavelet του wavelet CDF 9/7 σε εξειδικευμένο υλικό, η καλύτερη από κάθε άποψη αρχιτεκτονική με σταθερούς πολλαπλασιαστές δυαδικού δένδρου αθροιστών διάδοσης κρατουμένου, και εν γένει με χρήση στοιχείων διάδοσης κρατουμένου, είναι αυτή που βασίζεται στο lifting scheme.

Επιπλέον, προκύπτει ότι η χρήση αυτού του είδους σταθερών πολλαπλασιαστών θα πρέπει να αποφεύγεται σε άλλες αρχιτεκτονικές, και εν γένει σε FIR φίλτρα, καθώς εισάγουν απαράδεκτο σφάλμα υπολογισμού. Ακόμα, φαίνεται καθαρά πόσο μεγάλη σημασία έχει μια σωστή «dsp» ανάλυση ενός γράφου σηματορροής πριν την υλοποίησή της σε υλικό. Μια σωστή θεωρητική σχεδίαση μπορεί και «συγχωρεί» μέχρι κάποιο βαθμό τα σφάλματα στους υπολογισμούς που εισάγουν τα διάφορα στοιχεία του κυκλώματος: συνεπώς, όπως στην περίπτωσή μας, μπορούμε να χρησιμοποιήσουμε άφοβα στοιχεία που εισάγουν ένα μικρό σφάλμα, αλλά έχουν μεγάλα πλεονεκτήματα σε τομείς ταχύτητας, υλικού και ενέργειας.

Ωστόσο, υπάρχει μια πληθώρα, ακόμα, σχετικών θεμάτων που ίσως άξιζε να ερευνηθούν στο μέλλον. Μερικά από αυτά είναι τα εξής:

- Αν αντί για αθροιστές σταθερού μήκους χρησιμοποιούσαμε στους πολλαπλασιαστές αθροιστές μεταβλητού μήκους, έτσι ώστε να περιορίζαμε, ή να εξαλείφαμε τις αποκοπές, πόσο μεγαλύτερο θα ήταν το κόστος σε υλικό; Σε τι κόστος σε υλικό μεταφράζεται η βελτίωση της ακρίβειας σε κάποια αρχιτεκτονική κατά κάποια ποσότητα; Έχει αυτό κάποιο νόημα σε «αυτο-διορθωόμενα» κυκλώματα, όπως αυτό του lifting scheme;
- Εάν αντί για carry-propagate χρησιμοποιούσαμε carry-save στοιχεία, πόσο μεγαλύτερη ταχύτητα αλλά και πόσο μεγαλύτερες απαιτήσεις σε υλικό θα είχαμε;
- Η υλοποίηση των παραπάνω αρχιτεκτονικών σε πραγματικό υλικό (asic/fpga) ανταποκρίνεται στις προβλέψεις μας, ή αναδεικνύει άλλες αρχιτεκτονικές ως καλύτερες;
- Πολλά fpga σήμερα περιλαμβάνουν έτοιμους, γενικούς πολλαπλασιαστές στο κύκλωμά τους. Μήπως λοιπόν οι folded αρχιτεκτονικές που δεν αναπτύξαμε ως το επίπεδο πύλης εδώ, είναι πιο αποδοτικές για χρήση σε αυτά;
- Η εκτέλεση του μετασχηματισμού wavelet από ένα εξειδικευμένο κύκλωμα υλοποιημένο σε fpga είναι τελικά πιο γρήγορη από την εκτέλεση του ίδιου αλγορίθμου σε έναν σύγχρονο ψηφιακό επεξεργαστή σήματος ή σε έναν σύγχρονο επεξεργαστή γενικού σκοπού με ενσωματωμένες SIMD εντολές; (π.χ. MMX της Intel, 3Dnow! της AMD);

Παράρτημα Α:

Βιβλιογραφία

Α. Βιβλία και δημοσιεύσεις για θεωρία των wavelets και επεξεργασία σημάτων

- [A1] J. Proakis and D. Manolakis, *Digital Signal Processing*, 3rd Edition. Prentice Hall, Upper Saddle River NJ, 1996
- [A2] S. Burrus, A. Gopinath and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, Upper Saddle River NJ, 1998
- [A3] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge, Wellesley MA, 1996
- [A4] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic, San Diego CA. 1998
- [A5] M. Vitterli and J. Kovacevic, *Wavelets and Subband Coding*. Prentice Hall, Upper Saddle River NJ, 1995
- [A6] M. Frazier, *An Introduction to Wavelets Through Linear Algebra*. Springer-Verlag, New York NY, 1999
- [A7] I. Daubechies, *Ten Lectures on Wavelets*. SIAM, Philadelphia PA, 1992
- [A8] H.G. Stark, *Wavelets and Signal Processing*. Springer Verlag, Berling Heidelberg, 2005
- [A9] M. Misiti, Y. Misiti, G. Oppenheim and J-M Poggi *Wavelet Toolbox for use with MATLAB*[®]
- [A10] E. Stollnitz, T. DeRose and D. Salesin, *Wavelets for Computer Graphics: A Primer*

Β. Βιβλία και δημοσιεύσεις για γενική υλοποίηση μετασχηματισμών wavelet σε υλικό με αρχιτεκτονικές βασισμένες στη συνέλιξη

- [B1] K. Parhi and T. Nishitani, “VLSI Architectures for Discrete Wavelet Transforms”, *IEEE Transactions on VLSI Systems*, vol. 1, no. 2, pp. 191-202, 1993
- [B2] K. Pekmestzi and P. Bougas, “Systematic Approach to Design Folded Pipelined Architectures for the 1-D Wavelet Transform”
- [B3] D. Sripathi, *Efficient Implementation of Discrete Wavelet Transforms using FPGAs*, MSc Thesis, 2003

- [B4] S. Masud and J. McCanny, “Reusable Silicon IP Cores for Discrete Wavelet Transform Applications”, *IEEE Transactions on Circuits and Systems*, vol. 51, no. 6, pp. 114-124, 2004
- [B5] T. Cooklev, “An Efficient Architecture for Orthogonal Wavelet Transforms”, *IEEE Signal Processing Letters*, vol. 13, no. 2, pp. 77-79, 2006

C. Δημοσιεύσεις για το Lifting Scheme

Θεωρία

- [C1] W. Sweldens and P. Schroder, *Building Your Own Wavelets at Home*
- [C2] W. Sweldens, “The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets”, *Applied and Computational Harmonic Analysis*, vol. 3, no. 15, pp. 186-200, 1996
- [C3] I. Daubechies and W. Sweldens, “Factoring Wavelet Transforms into Lifting Steps”, *The J. of Fourier Analysis and Applications*, vol. 4, pp. 247-269, 1998

Αρχιτεκτονικές

- [C4] T. Acharya and C. Chakrabarti, “A Survey on Lifting-Based Discrete Wavelet Transform Architectures”, *J. of VLSI Signal Processing*, vol. 42, pp. 321-339, 2006
- [C5] W. Jiang and A. Ortega, “Lifting Factorization-Based Discrete Wavelet Transform Architecture Design”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 651-657, 2001
- [C6] C. Huang, P. Tseng and L. Chen, “Flipping Structure: An Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Transform”, *IEEE Transactions on Signal Processing*, pp. 1080-1089, 2004
- [C7] C. Xiong, J. Tian and J. Liu, “A note on “Flipping Structure: An Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Transform” ”, *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 1910-1916, 2006

D. Δημοσιεύσεις για τη B-Spline Παραγοντοποίηση

Θεωρία

- [D1] M. Unser and T. Blu, “Wavelet Theroy Demystified”, *IEEE Transactions on Signal Processing*, vol. 51, no.2, pp. 470-483, 2003

Αρχιτεκτονικές

- [D2] C. Huang, P. Tseng and L. Chen, “VLSI Architecture for Forward Discrete Wavelet Transform Based on B-Spline Factorization”, *Journal of VLSI Signal Processing*, vol. 40, pp. 343-353, 2005
- [D3] C. Huang, P. Tseng and L. Chen, “Analysis and VLSI Architecture for 1-D and 2-D Discrete Wavelet Transform”, *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1575-1586, 2005

Ε. Βιβλία και δημοσιεύσεις για σχεδίαση και γλώσσες περιγραφής υλικού

- [E1] S. White, “Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review”, *IEEE ASSP Magazine*, pp. 4-19, July 1989
- [E2] Xilinx Corporation, *The Role of Distributed Arithmetic in FPGA-Based Signal Processing*
- [E3] U. Meyer-Baese, *Digital Signal Processing with Field-Programmable Gate Arrays*, Springer-Verlag, Berlin-Heildeberg, 2001
- [E4] J-P Deschamps, G. Bioul and G. Sutter, *Synthesis of Arithmetic Circuits*, John Wiley&Sons Inc., 2006
- [E5] Xilinx Corporation, *Synthesis and Simulation Design Guide*
- [E6] Κ. Πεκμεστζή, *Ψηφιακά Συστήματα VLSI*, Έκδοση ΕΜΠ, Αθήνα, 1999
- [E7] M. Clietti, *Advanced Digital Design with the Verilog HDL*, Prentice-Hall, Upper Saddle River NJ, 2003
- [E8] IEEE Computer Society, *IEEE Standard Verilog Hardware Description Language*, 2001
- [E9] Samir Palnitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, SunSoft Press, 1996
- [E10] Behrooz Parhami, *Computer Arithmetic-Algorithms and Hardware Designs*, Oxford University Press, New York, 2004
- [E11] Mi Lu, *Arithmetic and Logic in Computer Systems*, John Wiley and Sons Inc., 2004
- [E12] George A. Constantinides, Peter Y.K. Cheung and Wayne Luk, *Synthesis and Optimization of DSP Algorithms*, Kluwer Academic Publishers, Boston, 2004
- [E13] K.K. Parhi, *VLSI Digital Signal Processing Systems*, John Wiley & Sons, Inc, New York, 1999