

THE UNIVERSITY OF CALGARY

# ROM Based Digital Filtering

by

Michael Sepa

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF

ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

April, 1994

© Michael Sepa 1994



National Library  
of Canada

Acquisitions and  
Bibliographic Services Branch

395 Wellington Street  
Ottawa, Ontario  
K1A 0N4

Bibliothèque nationale  
du Canada

Direction des acquisitions et  
des services bibliographiques

395, rue Wellington  
Ottawa (Ontario)  
K1A 0N4

*Your file    Votre référence*

*Our file    Notre référence*

THE AUTHOR HAS GRANTED AN  
IRREVOCABLE NON-EXCLUSIVE  
LICENCE ALLOWING THE NATIONAL  
LIBRARY OF CANADA TO  
REPRODUCE, LOAN, DISTRIBUTE OR  
SELL COPIES OF HIS/HER THESIS BY  
ANY MEANS AND IN ANY FORM OR  
FORMAT, MAKING THIS THESIS  
AVAILABLE TO INTERESTED  
PERSONS.

L'AUTEUR A ACCORDE UNE LICENCE  
IRREVOCABLE ET NON EXCLUSIVE  
PERMETTANT A LA BIBLIOTHEQUE  
NATIONALE DU CANADA DE  
REPRODUIRE, PRETER, DISTRIBUER  
OU VENDRE DES COPIES DE SA  
THESE DE QUELQUE MANIERE ET  
SOUS QUELQUE FORME QUE CE SOIT  
POUR METTRE DES EXEMPLAIRES DE  
CETTE THESE A LA DISPOSITION DES  
PERSONNE INTERESSEES.

THE AUTHOR RETAINS OWNERSHIP  
OF THE COPYRIGHT IN HIS/HER  
THESIS. NEITHER THE THESIS NOR  
SUBSTANTIAL EXTRACTS FROM IT  
MAY BE PRINTED OR OTHERWISE  
REPRODUCED WITHOUT HIS/HER  
PERMISSION.

L'AUTEUR CONSERVE LA PROPRIETE  
DU DROIT D'AUTEUR QUI PROTEGE  
SA THESE. NI LA THESE NI DES  
EXTRAITS SUBSTANTIELS DE CELLE-  
CI NE DOIVENT ETRE IMPRIMES OU  
AUTREMENT REPRODUITS SANS SON  
AUTORISATION.

ISBN 0-315-99483-5

Name Michael Sepa RAM Based Digital Filtering

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Electronics and Electrical

SUBJECT TERM

Engineering

0544

SUBJECT CODE

U·M·I

## Subject Categories

### THE HUMANITIES AND SOCIAL SCIENCES

#### COMMUNICATIONS AND THE ARTS

Architecture ..... 0729  
Art History ..... 0377  
Cinema ..... 0900  
Dance ..... 0378  
Fine Arts ..... 0357  
Information Science ..... 0723  
Journalism ..... 0391  
Library Science ..... 0399  
Mass Communications ..... 0708  
Music ..... 0413  
Speech Communication ..... 0459  
Theater ..... 0465

#### EDUCATION

General ..... 0515  
Administration ..... 0514  
Adult and Continuing ..... 0516  
Agricultural ..... 0517  
Art ..... 0273  
Bilingual and Multicultural ..... 0282  
Business ..... 0688  
Community College ..... 0275  
Curriculum and Instruction ..... 0277  
Early Childhood ..... 0518  
Elementary ..... 0524  
Finance ..... 0277  
Guidance and Counseling ..... 0519  
Health ..... 0680  
Higher ..... 0745  
History of ..... 0520  
Home Economics ..... 0278  
Industrial ..... 0521  
Language and Literature ..... 0279  
Mathematics ..... 0280  
Music ..... 0522  
Philosophy of ..... 0998  
Physical ..... 0523

Psychology ..... 0525  
Reading ..... 0535  
Religious ..... 0527  
Sciences ..... 0714  
Secondary ..... 0533  
Social Sciences ..... 0534  
Sociology of ..... 0340  
Special ..... 0529  
Teacher Training ..... 0530  
Technology ..... 0710  
Tests and Measurements ..... 0288  
Vocational ..... 0747

#### LANGUAGE, LITERATURE AND LINGUISTICS

Language  
General ..... 0679  
Ancient ..... 0289  
Linguistics ..... 0290  
Modern ..... 0291  
Literature  
General ..... 0401  
Classical ..... 0294  
Comparative ..... 0295  
Medieval ..... 0297  
Modern ..... 0298  
African ..... 0316  
American ..... 0591  
Asian ..... 0305  
Canadian (English) ..... 0352  
Canadian (French) ..... 0355  
English ..... 0593  
Germanic ..... 0311  
Latin American ..... 0312  
Middle Eastern ..... 0315  
Romance ..... 0313  
Slavic and East European ..... 0314

#### PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy ..... 0422  
Religion  
General ..... 0318  
Biblical Studies ..... 0321  
Clergy ..... 0319  
History of ..... 0320  
Philosophy of ..... 0322  
Theology ..... 0469

#### SOCIAL SCIENCES

American Studies ..... 0323  
Anthropology  
Archaeology ..... 0324  
Cultural ..... 0326  
Physical ..... 0327  
Business Administration  
General ..... 0310  
Accounting ..... 0272  
Banking ..... 0770  
Management ..... 0454  
Marketing ..... 0338  
Canadian Studies ..... 0385  
Economics  
General ..... 0501  
Agricultural ..... 0503  
Commerce-Business ..... 0505  
Finance ..... 0508  
History ..... 0509  
Labor ..... 0510  
Theory ..... 0511  
Folklore ..... 0358  
Geography ..... 0366  
Gerontology ..... 0351  
History  
General ..... 0578

Ancient ..... 0579  
Medieval ..... 0581  
Modern ..... 0582  
Black ..... 0328  
African ..... 0331  
Asia; Australia and Oceania ..... 0332  
Canadian ..... 0334  
European ..... 0335  
Latin American ..... 0336  
Middle Eastern ..... 0333  
United States ..... 0337  
History of Science ..... 0585  
Law ..... 0398  
Political Science  
General ..... 0615  
International Law and  
Relations ..... 0616  
Public Administration ..... 0617  
Recreation ..... 0814  
Social Work ..... 0452  
Sociology  
General ..... 0626  
Criminology and Penology ..... 0627  
Demography ..... 0938  
Ethnic and Racial Studies ..... 0631  
Individual and Family  
Studies ..... 0628  
Industrial and Labor  
Relations ..... 0629  
Public and Social Welfare ..... 0630  
Social Structure and  
Development ..... 0700  
Theory and Methods ..... 0344  
Transportation ..... 0709  
Urban and Regional Planning ..... 0999  
Women's Studies ..... 0453

### THE SCIENCES AND ENGINEERING

#### BIOLOGICAL SCIENCES

Agriculture  
General ..... 0473  
Agronomy ..... 0285  
Animal Culture and  
Nutrition ..... 0475  
Animal Pathology ..... 0476  
Food Science and  
Technology ..... 0359  
Forestry and Wildlife ..... 0478  
Plant Culture ..... 0479  
Plant Pathology ..... 0480  
Plant Physiology ..... 0817  
Range Management ..... 0777  
Wood Technology ..... 0746  
Biology  
General ..... 0306  
Anatomy ..... 0287  
Biostatistics ..... 0308  
Botany ..... 0309  
Cell ..... 0379  
Ecology ..... 0329  
Entomology ..... 0353  
Genetics ..... 0369  
Limnology ..... 0793  
Microbiology ..... 0410  
Molecular ..... 0307  
Neuroscience ..... 0317  
Oceanography ..... 0416  
Physiology ..... 0433  
Radiation ..... 0821  
Veterinary Science ..... 0778  
Zoology ..... 0472  
Biophysics  
General ..... 0786  
Medical ..... 0760  
EARTH SCIENCES  
Biogeochemistry ..... 0425  
Geochemistry ..... 0996

Geodesy ..... 0370  
Geology ..... 0372  
Geophysics ..... 0373  
Hydrology ..... 0388  
Mineralogy ..... 0411  
Paleobotany ..... 0345  
Paleoecology ..... 0426  
Paleontology ..... 0418  
Paleozoology ..... 0985  
Palynology ..... 0427  
Physical Geography ..... 0368  
Physical Oceanography ..... 0415

#### HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences ..... 0768  
Health Sciences  
General ..... 0566  
Audiology ..... 0300  
Chemotherapy ..... 0992  
Dentistry ..... 0567  
Education ..... 0350  
Hospital Management ..... 0769  
Human Development ..... 0758  
Immunology ..... 0982  
Medicine and Surgery ..... 0564  
Mental Health ..... 0347  
Nursing ..... 0569  
Nutrition ..... 0570  
Obstetrics and Gynecology ..... 0380  
Occupational Health and  
Therapy ..... 0354  
Ophthalmology ..... 0381  
Pathology ..... 0571  
Pharmacology ..... 0419  
Pharmacy ..... 0572  
Physical Therapy ..... 0382  
Public Health ..... 0573  
Radiology ..... 0574  
Recreation ..... 0575

Speech Pathology ..... 0460  
Toxicology ..... 0383  
Home Economics ..... 0386

#### PHYSICAL SCIENCES

Pure Sciences  
Chemistry  
General ..... 0485  
Agricultural ..... 0749  
Analytical ..... 0486  
Biochemistry ..... 0487  
Inorganic ..... 0488  
Nuclear ..... 0738  
Organic ..... 0490  
Pharmaceutical ..... 0491  
Physical ..... 0494  
Polymer ..... 0495  
Radiation ..... 0754  
Mathematics ..... 0405  
Physics  
General ..... 0605  
Acoustics ..... 0986  
Astronomy and  
Astrophysics ..... 0606  
Atmospheric Science ..... 0608  
Atomic ..... 0748  
Electronics and Electricity ..... 0607  
Elementary Particles and  
High Energy ..... 0798  
Fluid and Plasma ..... 0759  
Molecular ..... 0609  
Nuclear ..... 0610  
Optics ..... 0752  
Radiation ..... 0756  
Solid State ..... 0611  
Statistics ..... 0463  
Applied Sciences  
Applied Mechanics ..... 0346  
Computer Science ..... 0984

Engineering  
General ..... 0537  
Aerospace ..... 0538  
Agricultural ..... 0539  
Automotive ..... 0540  
Biomedical ..... 0541  
Chemical ..... 0542  
Civil ..... 0543  
Electronics and Electrical ..... 0544  
Heat and Thermodynamics ..... 0348  
Hydraulic ..... 0545  
Industrial ..... 0546  
Marine ..... 0547  
Materials Science ..... 0794  
Mechanical ..... 0548  
Metallurgy ..... 0743  
Mining ..... 0551  
Nuclear ..... 0552  
Packaging ..... 0549  
Petroleum ..... 0765  
Sanitary and Municipal ..... 0554  
System Science ..... 0790  
Geotechnology ..... 0428  
Operations Research ..... 0796  
Plastics Technology ..... 0795  
Textile Technology ..... 0994

#### PSYCHOLOGY

General ..... 0621  
Behavioral ..... 0384  
Clinical ..... 0622  
Developmental ..... 0620  
Experimental ..... 0623  
Industrial ..... 0624  
Personality ..... 0625  
Physiological ..... 0989  
Psychobiology ..... 0349  
Psychometrics ..... 0632  
Social ..... 0451



Nom \_\_\_\_\_

*Dissertation Abstracts International* est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

--	--	--	--	--

U·M·I

SUJET

CODE DE SUJET

## Catégories par sujets

### HUMANITÉS ET SCIENCES SOCIALES

#### COMMUNICATIONS ET LES ARTS

Architecture .....	0729
Beaux-arts .....	0357
Bibliothéconomie .....	0399
Cinéma .....	0900
Communication verbale .....	0459
Communications .....	0708
Danse .....	0378
Histoire de l'art .....	0377
Journalisme .....	0391
Musique .....	0413
Sciences de l'information .....	0723
Théâtre .....	0465

#### ÉDUCATION

Généralités .....	515
Administration .....	0514
Art .....	0273
Collèges communautaires .....	0275
Commerce .....	0688
Économie domestique .....	0278
Éducation permanente .....	0516
Éducation préscolaire .....	0518
Éducation sanitaire .....	0680
Enseignement agricole .....	0517
Enseignement bilingue et multiculturel .....	0282
Enseignement industriel .....	0521
Enseignement primaire .....	0524
Enseignement professionnel .....	0747
Enseignement religieux .....	0527
Enseignement secondaire .....	0533
Enseignement spécial .....	0529
Enseignement supérieur .....	0745
Évaluation .....	0288
Finances .....	0277
Formation des enseignants .....	0530
Histoire de l'éducation .....	0520
Langues et littérature .....	0279

Lecture .....	0535
Mathématiques .....	0280
Musique .....	0522
Orientation et consultation .....	0519
Philosophie de l'éducation .....	0998
Physique .....	0523
Programmes d'études et enseignement .....	0727
Psychologie .....	0525
Sciences .....	0714
Sciences sociales .....	0534
Sociologie de l'éducation .....	0340
Technologie .....	0710

#### LANGUE, LITTÉRATURE ET LINGUISTIQUE

Langues	
Généralités .....	0679
Anciennes .....	0289
Linguistique .....	0290
Modernes .....	0291
Littérature	
Généralités .....	0401
Anciennes .....	0294
Comparée .....	0295
Médiévale .....	0297
Moderne .....	0298
Africaine .....	0316
Américaine .....	0591
Anglaise .....	0593
Asiatique .....	0305
Canadienne (Anglaise) .....	0352
Canadienne (Française) .....	0355
Germanique .....	0311
Latino-américaine .....	0312
Moyen-orientale .....	0315
Romane .....	0313
Slave et est-européenne .....	0314

#### PHILOSOPHIE, RELIGION ET THÉOLOGIE

Philosophie .....	0422
Religion	
Généralités .....	0318
Clergé .....	0319
Études bibliques .....	0321
Histoire des religions .....	0320
Philosophie de la religion .....	0322
Théologie .....	0469

#### SCIENCES SOCIALES

Anthropologie	
Archéologie .....	0324
Culturelle .....	0326
Physique .....	0327
Droit .....	0398
Économie	
Généralités .....	0501
Commerce-Affaires .....	0505
Économie agricole .....	0503
Économie du travail .....	0510
Finances .....	0508
Histoire .....	0509
Théorie .....	0511
Études américaines .....	0323
Études canadiennes .....	0385
Études féministes .....	0453
Folklore .....	0358
Géographie .....	0366
Gérontologie .....	0351
Gestion des affaires	
Généralités .....	0310
Administration .....	0454
Banques .....	0770
Comptabilité .....	0272
Marketing .....	0338
Histoire	
Histoire générale .....	0578

Ancienne .....	0579
Médiévale .....	0581
Moderne .....	0582
Histoire des noirs .....	0328
Africaine .....	0331
Canadienne .....	0334
États-Unis .....	0337
Européenne .....	0335
Moyen-orientale .....	0333
Latino-américaine .....	0336
Asie, Australie et Océanie .....	0332
Histoire des sciences .....	0585
Loisirs .....	0814
Planification urbaine et régionale .....	0999
Science politique	
Généralités .....	0615
Administration publique .....	0617
Droit et relations internationales .....	0616
Sociologie	
Généralités .....	0626
Aide et bien-être social .....	0630
Criminologie et établissements pénitentiaires .....	0627
Démographie .....	0938
Études de l'individu et de la famille .....	0628
Études des relations interethniques et des relations raciales .....	0631
Structure et développement social .....	0700
Théorie et méthodes .....	0344
Travail et relations industrielles .....	0629
Transports .....	0709
Travail social .....	0452

### SCIENCES ET INGÉNIERIE

#### SCIENCES BIOLOGIQUES

Agriculture	
Généralités .....	0473
Agronomie .....	0285
Alimentation et technologie alimentaire .....	0359
Culture .....	0479
Élevage et alimentation .....	0475
Exploitation des pâturages .....	0777
Pathologie animale .....	0476
Pathologie végétale .....	0480
Physiologie végétale .....	0817
Sylviculture et faune .....	0478
Technologie du bois .....	0746
Biologie	
Généralités .....	0306
Anatomie .....	0287
Biologie (Statistiques) .....	0308
Biologie moléculaire .....	0307
Botanique .....	0309
Cellule .....	0379
Écologie .....	0329
Entomologie .....	0353
Génétique .....	0369
Limnologie .....	0793
Microbiologie .....	0410
Neurologie .....	0317
Océanographie .....	0416
Physiologie .....	0433
Radiation .....	0821
Science vétérinaire .....	0778
Zoologie .....	0472
Biophysique	
Généralités .....	0786
Médicale .....	0760

Géologie .....	0372
Géophysique .....	0373
Hydrologie .....	0388
Minéralogie .....	0411
Océanographie physique .....	0415
Paléobotanique .....	0345
Paléoécologie .....	0426
Paléontologie .....	0418
Paléozoologie .....	0985
Palynologie .....	0427

#### SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT

Économie domestique .....	0386
Sciences de l'environnement .....	0768
Sciences de la santé	
Généralités .....	0566
Administration des hôpitaux .....	0769
Alimentation et nutrition .....	0570
Audiologie .....	0300
Chimiothérapie .....	0992
Dentisterie .....	0567
Développement humain .....	0758
Enseignement .....	0350
Immunologie .....	0982
Loisirs .....	0575
Médecine du travail et thérapie .....	0354
Médecine et chirurgie .....	0564
Obstétrique et gynécologie .....	0380
Ophtalmologie .....	0381
Orthophonie .....	0460
Pathologie .....	0571
Pharmacie .....	0572
Pharmacologie .....	0419
Physiothérapie .....	0382
Radiologie .....	0574
Santé mentale .....	0347
Santé publique .....	0573
Soins infirmiers .....	0569
Toxicologie .....	0383

#### SCIENCES PHYSIQUES

Sciences Pures	
Chimie	
Généralités .....	0485
Biochimie .....	0487
Chimie agricole .....	0749
Chimie analytique .....	0486
Chimie minérale .....	0488
Chimie nucléaire .....	0738
Chimie organique .....	0490
Chimie pharmaceutique .....	0491
Physique .....	0494
Polymères .....	0495
Radiation .....	0754
Mathématiques .....	0405
Physique	
Généralités .....	0605
Acoustique .....	0986
Astronomie et astrophysique .....	0606
Électronique et électricité .....	0607
Fluides et plasma .....	0759
Météorologie .....	0608
Optique .....	0752
Particules (Physique nucléaire) .....	0798
Physique atomique .....	0748
Physique de l'état solide .....	0611
Physique moléculaire .....	0609
Physique nucléaire .....	0610
Radiation .....	0756
Statistiques .....	0463
Sciences Appliqués Et Technologie	
Informatique .....	0984
Ingénierie	
Généralités .....	0537
Agricole .....	0539
Automobile .....	0540

Biomédicale .....	0541
Chaleur et thermodynamique .....	0348
Conditionnement (Emballage) .....	0549
Génie aérospatial .....	0538
Génie chimique .....	0542
Génie civil .....	0543
Génie électronique et électrique .....	0544
Génie industriel .....	0546
Génie mécanique .....	0548
Génie nucléaire .....	0552
Ingénierie des systèmes .....	0790
Mécanique navale .....	0547
Métallurgie .....	0743
Science des matériaux .....	0794
Technique du pétrole .....	0765
Technique minière .....	0551
Techniques sanitaires et municipales .....	0554
Technologie hydraulique .....	0545
Mécanique appliquée .....	0346
Géotechnologie .....	0428
Matériaux plastiques (Technologie) .....	0795
Recherche opérationnelle .....	0796
Textiles et tissus (Technologie) .....	0794

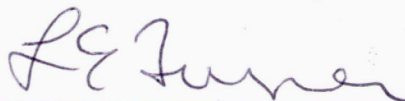
#### PSYCHOLOGIE

Généralités .....	0621
Personnalité .....	0625
Psychobiologie .....	0349
Psychologie clinique .....	0622
Psychologie du comportement .....	0384
Psychologie du développement .....	0620
Psychologie expérimentale .....	0623
Psychologie industrielle .....	0624
Psychologie physiologique .....	0989
Psychologie sociale .....	0451
Psychométrie .....	0632

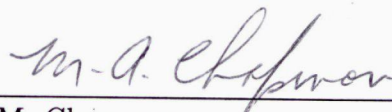


The University of Calgary  
Faculty of Graduate Studies

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled, "ROM Based Digital Filtering" submitted by Michael Sepa in partial fulfillment of the requirements for the degree of Master of Science.



Supervisor, Dr. L. E. Turner  
Electrical and Computer Engineering



Dr. M. Chapman  
Geomatics Engineering



Dr. M. R. Smith  
Electrical and Computer Engineering



Dr. R. A. Stein  
Electrical and Computer Engineering

May 24, 1994

Date

## Abstract

Two new techniques are proposed for the construction of digital filters using read only memory (ROM) based technology. Both implement all components (adders and multipliers), with the exception of shift registers, in ROM. The principal difficulty in the construction of such devices is the large amounts of memory required, and each technique reduces the amount of memory required.

The first technique reduces the memory required by eliminating unreachable states. Since unreachable states in a filter will never be encountered, their elimination will not change the transfer function of the filter.

The second technique uses a difference modulator to reduce the dynamic range of the signal. The reduced dynamic range signal is processed using finite state machines which implement adder and multiplier operations. The processed signal is difference demodulated to construct the resulting waveform.

When compared to standard implementation techniques these implementations have several major advantages: high speed, simple modular component blocks and inexpensive material.

# Contents

<b>Approval Page</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Digital Filters Using ROM Components</b>	<b>7</b>
2.1 Distributed Arithmetic . . . . .	8
2.1.1 An Overview of DA . . . . .	8
2.1.2 The Basic Concept . . . . .	9
2.1.3 An Example . . . . .	11
2.1.4 ROM Implementations . . . . .	13
2.2 Residue Arithmetic Implementations . . . . .	14
2.2.1 The Basics of Residue Arithmetic . . . . .	14
2.2.2 Filter Construction Using Residue Arithmetic . . . . .	16
2.2.3 ROM Implementation Using Residue Arithmetic . . . . .	19
2.3 Conclusions on Look-up Table Implementations . . . . .	20
<b>3 ROM Reduction by Elimination of Unreachable States</b>	<b>22</b>
3.1 Determining the Reachable Set . . . . .	24
3.1.1 Reachable Set for Inputs with Bounded Energy . . . . .	27
3.1.2 Reachable Set for Bounded Amplitude Inputs . . . . .	29
3.1.3 A Direct Form Filter Example . . . . .	30
3.2 The Relationship Between the Fraction of Space Used and the Filter Order. . . . .	34
3.3 Reducing the Unreachable Set . . . . .	35

3.4	Simulation and Analysis . . . . .	38
3.5	Conclusions . . . . .	42
<b>4</b>	<b>Filtering of Delta Modulated Signals</b>	<b>44</b>
4.1	Delta Modulation . . . . .	45
4.2	The Basics of Delta Modulation Systems . . . . .	45
4.3	Delta Modulation Filtering using Analog-digital Hybrid Techniques .	47
4.3.1	Non-Recursive Filtering . . . . .	48
4.3.2	Recursive Filtering . . . . .	49
4.3.3	The Possibility for ROM Implementation . . . . .	49
4.4	Digital Filtering of Delta Modulated Signals . . . . .	51
4.4.1	The Delta Adder . . . . .	51
4.4.2	Delta Multiplier . . . . .	53
4.4.3	Hardware implementation . . . . .	56
4.4.4	Filtering Delta modulated Signals . . . . .	56
4.4.5	Delta doubler . . . . .	58
4.4.6	Method Problems . . . . .	58
4.4.7	ROM Implementation . . . . .	59
4.5	Filtering Ternary Delta Modulation . . . . .	59
4.5.1	Basic Ternary Delta Modulation . . . . .	60
4.5.2	Ternary adder . . . . .	61
4.5.3	Ternary Multiplier . . . . .	62
4.5.4	Ternary Delta modulation Filters . . . . .	63
4.5.5	Ternary Delta Tripler . . . . .	64
4.5.6	Hardware . . . . .	64
4.5.7	Method Problems . . . . .	64
4.6	Filtering of Sigma-Delta Modulated Signals . . . . .	65



4.6.1	The Basics of Sigma-Delta Modulation . . . . .	65
4.6.2	Filtering Sigma-Delta Modulated Signals . . . . .	66
4.6.3	Problems in the Sigma-Delta Approach . . . . .	68
4.6.4	ROM Implementation . . . . .	68
4.7	Conclusions . . . . .	69
<b>5</b>	<b>Filtering Difference Signals</b>	<b>70</b>
5.1	Introduction . . . . .	70
5.2	Difference Signals . . . . .	73
5.3	Signal Coding, Reconstruction and Filtering . . . . .	74
5.3.1	Encoding of Difference Signals . . . . .	76
5.3.2	Decoding of Difference Signals . . . . .	77
5.3.3	Error in the Encoding/Decoding System . . . . .	79
5.3.4	Filtering Difference Signals . . . . .	81
5.4	Components . . . . .	82
5.4.1	Adding Difference Signals . . . . .	82
5.4.2	Multiplying Difference Signals . . . . .	91
5.5	Filter Construction . . . . .	103
5.6	Difference Filter Simulation . . . . .	108
5.6.1	Test Inputs and Results . . . . .	111
5.6.2	Limitations . . . . .	116
5.6.3	Memory Requirements . . . . .	117
5.6.4	Implementations other than ROM . . . . .	119
<b>6</b>	<b>Future Research Directions and Conclusions</b>	<b>121</b>
6.1	Technique Critique . . . . .	121
6.1.1	Elimination of Unreachable States Approach . . . . .	121
6.1.2	Difference Signal Approach . . . . .	123

6.2	Future Research Directions . . . . .	124
6.3	Conclusions . . . . .	126
	<b>Bibliography</b>	<b>127</b>

# List of Tables

2.1	ROM look-up table contents for the example DA implementation. . .	12
2.2	ROM adder space requirements for various wordlengths. . . . .	13
2.3	Unique residue pairs for $k$ when $R_1 = 3$ and $R_2 = 5$ . . . . .	15
4.1	All possible inputs with outputs to the delta adder . . . . .	53
4.2	Delta adder input/output relations rewritten to show correspondence with a bit-serial adder . . . . .	56
4.3	Input/output relations for the addition of ternary signals . . . . .	61
4.4	$Z_n$ as a function of sum, $S_n$ , and carry, $C_n$ . . . . .	62
5.1	Memory requirements for a $5^{th}$ order LDI using look-up tables. . . . .	71
5.2	Memory requirements for a Dmult for various coefficient wordlengths.	101
5.3	Design parameters and multiplier coefficients for the simulated $5^{th}$ order LDI filter . . . . .	109

# List of Figures

1.1	Block ROM implementation of a digital filter. . . . .	5
2.1	A basic distributed arithmetic implementation for four coefficients. . .	11
2.2	Block diagram of a residue arithmetic implementation. . . . .	19
3.1	The direct form digital filter reachable set for $\ \mathbf{u}\ _2^2 \leq 1$ . . . . .	32
3.2	Bounded amplitude reachable set for the direct form filter. . . . .	33
3.3	Maximum fraction of space used as a function of order for elliptical reachable sets. . . . .	36
3.4	Block diagram of the reduced full ROM implementation. . . . .	39
3.5	The ratio of maximum error between the ROM filter and the direct form filter as a function of wordlength, $b$ . . . . .	41
3.6	Maximum error in the FFT of the output as a function of wordlength, $b$ . .	41
4.1	A basic delta modulation system . . . . .	46
4.2	Non-recursive delta modulation filter [10] . . . . .	48
4.3	Recursive delta modulation filter [10]. . . . .	50
4.4	Multiplier for a $0.1011_b$ coefficient constructed from delta adders . . .	55
4.5	Delta adders used to create a tree of multiplication constants . . . . .	55
4.6	The basic ternary delta modulation system . . . . .	60
4.7	A ternary multiplier with a coefficient of $0.12346$ . . . . .	63
4.8	Basic sigma-delta modulation system . . . . .	66
4.9	A sigma-delta modulation FIR filter . . . . .	67
5.1	A $5^{th}$ order LDI structure composed of two input blocks. . . . .	71
5.2	Bit reduction by oversampling for telephone (3 kHz) bandwidth data. . .	75
5.3	Bit reduction by oversampling for audio (22 kHz) bandwidth data. . .	75

5.4	Difference encoder (Dencoder) block diagram. . . . .	77
5.5	Block diagram of the difference adder (Dadder). . . . .	83
5.6	Maximum number of consecutive carry accumulation samples. . . . .	87
5.7	Ideal sum and Dadder sum output waveforms for the random input sequence. . . . .	92
5.8	Error between the output and the ideal response in the Dadder for random waveform inputs. . . . .	92
5.9	Block diagram of the difference multiplier (Dmult). . . . .	93
5.10	The Dmult with quantizers modeled with noise. . . . .	97
5.11	Dmult error as a function of dmult coefficient. . . . .	98
5.12	Dmult output and ideal response for $m = 0.95$ . . . . .	100
5.13	Error in the Dmult output for $m = 0.95$ . . . . .	100
5.14	The crushing of poles and zeros into a small wedge by oversampling. .	104
5.15	Magnitude responses of the ideal and coefficient truncated filters. . .	110
5.16	A 5 <sup>th</sup> order LDI filter constructed from Dmult and Dadder blocks. .	112
5.17	Random waveform test output. . . . .	113
5.18	Random waveform test error. . . . .	114
5.19	Difference filter and ideal filter response when stimulated with a unit Kronecker delta . . . . .	115
5.20	Difference filter and ideal filter magnitude response. . . . .	116
5.21	Size of ROM required as a function of the oversampling ratio . . . . .	119

# Chapter 1

## Introduction

Signals can transmit an endless variety of information, including images, voltages, and sounds. Any signal is composed of a sequence of values, or amplitudes. This sequence can be of finite or infinite length, and the amplitudes may take any one value at any point in the sequence. For example, a voltage signal is composed of a sequence of voltage levels, and an image signal is composed of a sequence of images. In neither case does the signal possess two voltage levels or two images at any single point in the sequence. The consecutive amplitudes in a signal are separated in time thus signals can be considered as functions of time, but the time separating consecutive amplitudes of a signal need not be uniform. When the consecutive amplitudes are separated by zero time, the signal is a continuous function with the amplitude as the dependent variable, and time as the independent variable.

Signals are considered to be functions of time in this thesis, and can be classified into three categories: continuous, discrete and digital. Continuous signals are continuous functions in both time and amplitude. Such signals have a value at all times and can have any finite amplitude. Discrete signals are continuous in one variable, but are discontinuous or quantized in the other. In this thesis, discrete signals are considered to be discrete in time, and continuous in amplitude. Such a discrete signal has a known value only at specific moments in time, but the amplitude of such a signal may take on any specific value. Digital signals are quantized in both amplitude and time. Like discrete signals, digital signals have known amplitudes only at specific times, but, in addition, the amplitude of a digital signal must be one of a number of amplitudes. In practical applications, the number of amplitudes is restricted to a

finite number.

Signal processing is a method of enhancing or attenuating features of signals. Listening to a voice in a crowded room is an example of signal processing. Other voices in the room are attenuated, while the voice of interest is enhanced.

Digital signal processing (DSP) is signal processing applied to signals composed of digital (quantized in both amplitude and time) data. In DSP two main implementation methods have been used: custom hardware, such as integrated circuits (ICs) and field programmable gate arrays (FPGAs), and digital signal processors, such as the Motorola DSP56000 [12].

Digital signal processors (DSP chips) are microprocessors whose instruction set is specially designed for DSP operations. A DSP chip's properties include the following:

1. The algorithm is implemented in a software program (much like in a conventional microprocessor) which can be easily changed during prototyping.
2. Modest facilities are required for programming.
3. Moderate operating speeds can be obtained depending on the algorithm.
4. The cost on a per chip basis is moderately high, requiring large production runs to achieve low costs.

Custom integrated circuits are integrated circuits specially designed for one application. A custom integrated circuit constructed for DSP applications has the following properties:

1. The algorithm is implemented in hardware.
2. Complex (and costly) tools and facilities are required for construction.
3. Very high operating speeds are obtainable depending on the algorithm.

4. The design is fixed at manufacture, so prototyping is expensive.
5. The cost on a per chip basis is very high, requiring large productions to achieve low costs.

Field programmable gate arrays are integrated circuits capable of being programmed on site. An FPGA circuit designed for DSP applications has the following properties:

1. The algorithm is implemented in hardware.
2. Modest facilities are required for construction compared to those required for custom IC construction.
3. High operating speeds are obtainable depending on the algorithm.
4. The reprogrammable structure allows inexpensive prototyping.
5. The cost on a per chip basis is moderately high, requiring large productions to achieve low costs.

All three approaches are unsuitable for low volume, fixed (or static) algorithm designs. Integrated circuits cannot be produced in low volumes. DSP chips and FPGAs are reprogrammable, which is a capability wasted on fixed algorithms. In low volume (several units), low cost production of fixed algorithm designs, a fourth technological alternative is full read only memory based implementation. In this approach all computational components, such as adders, multipliers or larger blocks, are constructed from read only memory (ROM) chips.

ROM chips are commonly used in the computer industry and are inexpensive when purchased as single units and even less expensive per chip when purchased in large quantities. They are difficult (or impossible) to modify after production, so they are suited to static DSP algorithm implementations. ROMs can operate at



very high speeds. As a single state-machine, as shown in Figure 1.1, the maximum sample rate is the access time of the ROM, which can be extremely fast. For typical 1992 technology 20 ns access time is available, allowing potential sample rates of 50 MHz.

The problems encountered in constructing ROMs are greatly simplified compared to those encountered in the design of a custom IC. IC design requires the production of a logic circuit based on a set of input-output relations. In a ROM such a relationship is directly programmed, eliminating the need for logic minimization and timing analysis that is required in IC designs. In addition, electronically programmable read only memories (EPROMs) can be programmed using inexpensive EPROM programmers, while custom ICs require complex tools and facilities for production. For large production runs, ROMs can be fabricated much like ICs, with several ROMs on a common substrate, except that layout is greatly simplified because only ROM programming is required. If each IC is composed of several modular ROM blocks rather than one large ROM block, then only the interconnects between the modular components are required to implement the desired algorithm. Such an approach would be similar to FPGAs, but the ROM function blocks would be designed for DSP applications.

The three technologies encounter different trade-offs as the order of the filter increases. In IC implementations, as the filter order increases the chip area increases because there are more interconnects and components. IC implementations encounter a hardware complexity/order trade-off. In DSP chip implementations, the maximum sample rate decreases as the length of the program increases. Since the order of the filter is roughly proportional to the length of the program, DSP chips encounter a speed/order trade-off. In ROM implementations, the maximum sample rate can be maintained at the cost of an exponentially increasing memory size for

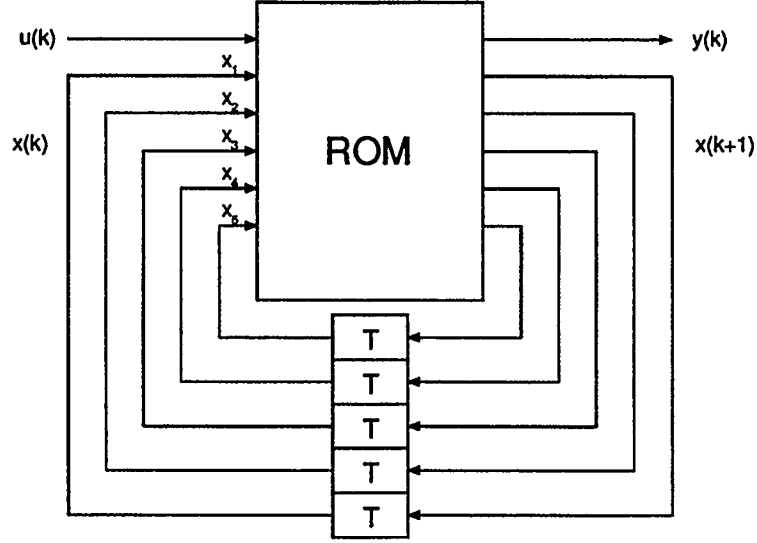


Figure 1.1: Block ROM implementation of a digital filter.

any order. ROM implementations encounter a memory size/order trade-off.

The trade-off between memory size and filter order is the major disadvantage of ROM based filters. ROM based filters, especially those built as block state-machines, require immense amounts of memory. Consider the 5<sup>th</sup> order digital filter based on the difference equations,

$$y(n) = bu(n) + \sum_{k=1}^5 a_k x_k(n) \quad (1.1)$$

$$x_1(n+1) = u(n) \quad (1.2)$$

$$x_2(n+1) = x_1 \quad (1.3)$$

$$x_3(n+1) = x_2 \quad (1.4)$$

$$x_4(n+1) = x_3 \quad (1.5)$$

$$x_5(n+1) = x_4 \quad (1.6)$$

where  $a_k$  and  $b$  are fixed finite precision multiplier coefficients. A block ROM state-machine digital filter is shown in Figure 1.1. If we consider the filter to have a uniform wordlength of 12 bits, the 5<sup>th</sup> order digital filter would require a ROM with six 12 bit inputs (five inputs for the states and one for the input) and six 12 bit

outputs (five for the next filter states and one for the output) which would require

$$12(5 + 1)2^{12(5+1)} = 3.4001 \times 10^{23} \text{ bits}$$

or 0.34 tera-terabits. In 1994, 1 megabit memories are commonly available, and computers often have 16 to 32 megabytes of memory, but 0.34 tera-terabits is an unreasonable large amount of memory for current technologies.

Let us assume that a vast amount of memory is available at a reasonable cost. If the ROM could be programmed at 1 bit per picosecond it would still be unfeasible to build because of the time required:

$$\begin{aligned} 3.4001 \times 10^{23} \times 10^{-12} &= 3.4 \times 10^{11} \text{ seconds} \\ &= 10\,781 \text{ years.} \end{aligned} \tag{1.7}$$

A block ROM implementation is an unsatisfactory approach because it consumes too much memory and requires too much time to program. To allow construction of ROM based digital filters some method of reducing the memory required to a reasonable amount is needed. A reduction in the ROM size will be accompanied by a reduction in the programming time.

This thesis focuses on two new approaches to ROM implementations of digital filters: reduction of the block ROM implementation memory requirements and ROM implementations composed of many small state-machines. The thesis begins with a review of the use of ROM components in digital filters, followed by an examination of block ROM implementations with reduced memory requirements achieved by the elimination of unreachable states. This is followed by a review of delta modulation filtering systems which provides background for a difference signal processing method using small ROM state-machines.

## Chapter 2

### Digital Filters Using ROM Components

Read only memories (ROMs) have been used in digital filtering to increase the speed of an algorithm by pre-computing the result of complex calculations and storing a table of the results in a ROM. This ROM, called a look-up table, can then be used during filter operation to quickly determine the result of the complex calculation rather than require the filter to compute the result each time. Such a scheme allows complex calculations to be conducted in the time required for accessing the ROM. In this chapter, two approaches to digital filtering using ROM components as look-up tables will be examined.

Digital filter implementations are frequently composed of adder and multiplier components. Of these two components the multiplier is the most resource consuming component, because it requires more gates to construct than adder components to attain a similar operation speed. When similar hardware limits are imposed on both components (such as restricting the maximum number of gates), multipliers require more time to operate than adder components. The complex circuitry required for construction make multipliers suitable for ROM or RAM based look-up table implementations because look-up table implementations simplify the circuitry required and allow faster operation. The high speed of the ROM ICs allows multiplexing of the ROM components which reduces the total memory required.

In both distributed arithmetic systems [17] and residue arithmetic systems [7] ROM look-up tables are used to create faster multipliers. These two implementation methods will be outlined and examined for potential full ROM implementation in this chapter.

## 2.1 Distributed Arithmetic

A common form of computation in digital filters is the sum of products (or in vector terms, the inner product). It is this operation that can be effectively performed by distributed arithmetic (DA) [17]. In conventional implementations (direct implementation of the difference equations using multiplier and adder components) the sum of products operation is performed by isolated adder and multiplier components. Each multiplication is performed by a multiplier, while each addition is performed by an adder. The adder and multiplier components are entirely separate, while only the final result of the arithmetic add or multiply operation is passed to other components.

By contrast, DA implementations are not based on isolated adders and multipliers. The inner product is performed by a unified multiplier/adder look-up table (for calculating a partial product) and a parallel accumulator. The parallel accumulator is constructed using conventional logic gates, and the look-up table is implemented in ROM. This approach can reduce the number of gates required for implementation by 50% to 80% over conventional isolated component implementations [17].

In this section, the basics of a DA implementation of a sum of products operation is explained. Full ROM implementations of DA operations are shown to be impractical because of the total memory requirements for such a system.

### 2.1.1 An Overview of DA

In its simplest form, a distributed arithmetic implementation uses a bit-serial computation that evaluates each bit of the inner product of a pair of vectors in a single, direct step [17]. The bit-serial nature of the operation reduces the speed of computation compared to that in conventional implementations, but this restriction is lifted when a DA implementation is expanded into word-serial (serial transmission

of several bits rather than a single bit) at the cost of a larger look-up table. With word-serial DA, speeds comparable to those of conventional implementations are possible.

### 2.1.2 The Basic Concept

An  $N^{th}$  order linear digital filter can be described by its state equations:

$$\mathbf{s}(n+1) = \mathbf{A}\mathbf{s}(n) + \mathbf{b}u(n) \quad (2.1)$$

$$y(n) = \mathbf{c}^T \mathbf{s}(n) + du(n) \quad (2.2)$$

where

$\mathbf{A}$  =  $N \times N$  state matrix

$\mathbf{b}$  =  $N$  dimensional column vector

$\mathbf{c}$  =  $N$  dimensional column vector

$d$  = scalar

$u(n)$  = scalar input at time  $n$

$\mathbf{s}(n)$  =  $N$  dimensional column state vector at time  $n$

$y(n)$  = scalar output at time  $n$ .

The state equations (2.1 and 2.2) can be rearranged as:

$$\begin{bmatrix} \mathbf{s}(n+1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c}^T & d \end{bmatrix} \begin{bmatrix} \mathbf{s}(n) \\ u(n) \end{bmatrix} \quad (2.3)$$

and from this equation it is clear that a direct implementation of a digital filter from its state equations requires  $N + 1$  inner product calculations.

In a DA implementation, each of the  $N + 1$  inner product calculations is implemented separately. Consider an arbitrary inner product calculation:

$$y = \sum_{k=1}^K a_k x_k \quad , \quad (2.4)$$

where  $a_k$  are arbitrary fixed coefficients and  $x_k$  are variable data words. In a filter,  $a_k$  represents the fixed multiplier coefficients, while  $x_k$  is the state and input information.

If each  $x_k$  is a two's complement binary number which is scaled such that  $|x_k| < 1$  (for convenience), then each  $x_k$  is expressed by  $b$  fractional bits,

$$x_k = -b_{k0} + \sum_{n=1}^{b-1} b_{kn} 2^{-n} \quad , \quad (2.5)$$

where  $b_{kn}$  are the bits of  $x_k$ ,  $b_{k0}$  is the sign bit,  $b$  is the number of bits and  $b_{k(b-1)}$  is the least significant bit (LSB).

Using Equations 2.5 and 2.4, the sum of products can be written as

$$y = \sum_{k=1}^K a_k \left[ -b_{k0} + \sum_{n=1}^{b-1} b_{kn} 2^{-n} \right] \quad . \quad (2.6)$$

This represents the conventional form of the inner product. When this equation is implemented directly it defines an isolated arithmetic computation. However, since  $n$  and  $k$  are independent, the order of the summations can be interchanged to form

$$y = \sum_{n=1}^{b-1} \left[ \sum_{k=1}^K a_k b_{kn} \right] 2^{-n} + \sum_{k=1}^K a_k (-b_{k0}) \quad , \quad (2.7)$$

which can be implemented in a distributed arithmetic form [17].

Since  $b_{kn}$  may only take the values of 0 or 1, the bracketed term in Equation 2.7,

$$\left[ \sum_{k=1}^K a_k b_{kn} \right] \quad , \quad (2.8)$$

may take only  $2^k$  possible values. By precalculating and storing these values in a ROM look-up table the input data  $b_k$  can be used to address the memory and the result of the computation can be read directly from the ROM data lines. Then the result can be added with a parallel accumulator. After  $b$  operations the final result of the inner product will be in the accumulator.

### 2.1.3 An Example

To illustrate a DA implementation, let us examine an example [17] where

$$K = 4, \quad a_1 = 0.72, \quad a_2 = -0.30, \quad a_3 = 0.95, \quad a_4 = 0.11 \quad .$$

The ROM look-up table must store all possible combinations of the coefficients and negative coefficients, so that  $2 \times 2^4$  terms are required. The block diagram of a DA implementation for the four coefficients (Figure 2.1) clearly shows the ROM look-up table, whose contents are listed in Table 2.1.

In the operation of the DA implementation, the data words are input into the ROM one bit at a time (1BAAT), from least significant bit (LSB) to most significant bit (MSB). On the MSB (the sign bit) the  $T_s$  signal is set high (1) to accommodate the sign bit. For all other bits the  $T_s$  signal is set low (0).

Beginning with a cleared accumulator, the LSB ( $b_{b-1}$ ) of each of the input states ( $x_1$  to  $x_4$ ) is addressed to the ROM. The parallel output of the ROM represents the sum of all coefficients which received an input bit equal to 1. The output of the ROM is added in parallel to half the value of the zeroed accumulator. The next bit ( $b_{b-2}$ ) of each of the inputs is then addressed to the ROM. The parallel output of the ROM is added to half the value of the accumulator. The process continues until  $b_0$  (the MSB or sign bit) is reached. When the MSB of the input states ( $b_0$  of  $x_1, x_2, x_3, x_4$ ) is addressed to the look-up table, the  $T_s$  line is set high to allow the

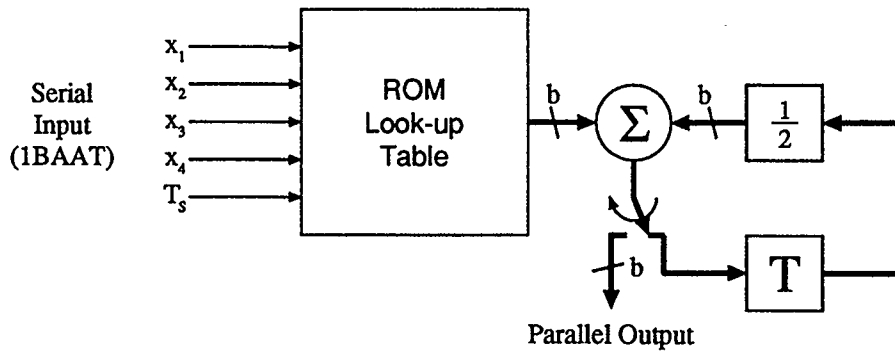


Figure 2.1: A basic distributed arithmetic implementation for four coefficients.



Input Code					32-Word
$T_s$	$b_{1n}$	$b_{2n}$	$b_{3n}$	$b_{4n}$	Memory Contents
0	0	0	0	0	0
0	0	0	0	1	$a_4 = 0.11$
0	0	0	1	0	$a_3 = 0.95$
0	0	0	1	1	$a_3 + a_4 = 1.06$
0	0	1	0	0	$a_2 = -0.30$
0	0	1	0	1	$a_2 + a_4 = -0.19$
0	0	1	1	0	$a_2 + a_3 = 0.65$
0	0	1	1	1	$a_2 + a_3 + a_4 = 0.75$
0	1	0	0	0	$a_1 = 0.72$
0	1	0	0	1	$a_1 + a_4 = 0.83$
0	1	0	1	0	$a_1 + a_3 = 1.67$
0	1	0	1	1	$a_1 + a_3 + a_4 = 1.78$
0	1	1	0	0	$a_1 + a_2 = 0.42$
0	1	1	0	1	$a_1 + a_2 + a_4 = 0.53$
0	1	1	1	0	$a_1 + a_2 + a_4 = 1.37$
0	1	1	1	1	$a_1 + a_2 + a_3 + a_4 = 1.48$
1	0	0	0	0	0
1	0	0	0	1	$-a_4 = -0.11$
1	0	0	1	0	$-a_3 = -0.95$
1	0	0	1	1	$-(a_3 + a_4) = -1.06$
1	0	1	0	0	$-a_2 = +0.30$
1	0	1	0	1	$-(a_2 + a_4) = +0.19$
1	0	1	1	0	$-(a_2 + a_3) = -0.65$
1	0	1	1	1	$-(a_2 + a_3 + a_4) = -0.75$
1	1	0	0	0	$-a_1 = -0.72$
1	1	0	0	1	$-(a_1 + a_4) = -0.83$
1	1	0	1	0	$-(a_1 + a_3) = -1.67$
1	1	0	1	1	$-(a_1 + a_3 + a_4) = -1.78$
1	1	1	0	0	$-(a_1 + a_2) = -0.42$
1	1	1	0	1	$-(a_1 + a_2 + a_4) = -0.53$
0	1	1	1	0	$-(a_1 + a_2 + a_4) = -1.37$
0	1	1	1	1	$-(a_1 + a_2 + a_3 + a_4) = -1.48$

Table 2.1: ROM look-up table contents for the example DA implementation.

negative sum of the coefficients to be computed. When the negative sum is added to half the accumulator, the accumulator will contain the two's complement binary number which represents the sum of products of the inputs and the fixed coefficients. The accumulator's contents can be passed on to other components of the design and, prior to the beginning of the next computation, the accumulator is zeroed.

#### 2.1.4 ROM Implementations

Although much work has been done to reduce the ROM required for the look-up tables in DA implementations [17], it is the adder/accumulator that presents the major difficulty in a full ROM implementation of a DA system. An accumulator structure is difficult to realize in a full ROM implementation because the adder is a parallel  $b$  bit adder and a parallel adder in a full ROM implementation is very expensive:

$$\text{size} = b2^{2b} \text{ bits} = \frac{b2^{2b}}{8} \text{ bytes} \quad (2.9)$$

For a wordlength of 8 bits, one adder requires 65536 bytes, but for larger word lengths the adder quickly becomes unmanageable (Table 2.2). If the DA implementation is such that each state and the output are computed in a DA scheme, one adder will be required for each state the output. Only for filters with small wordlengths is such a scheme possible. Direct full ROM implementation of a DA scheme is not feasible due the the large memory requirements of the parallel adder.

Wordlength	ROM size
10 bits	1.31 Megabytes
12 bits	25.2 Megabytes
16 bits	8.59 Gigabytes

Table 2.2: ROM adder space requirements for various wordlengths.

## 2.2 Residue Arithmetic Implementations

Like distributed arithmetic implementations, residue arithmetic implementations often use ROM components as look-up tables. A principal advantage of residue arithmetic implementations is the reduction of the carry path length in the ripple adder. Reducing the carry path length increases the maximum possible operation speed of the adder.

In a ripple full adder the carry of the addition of the two LSBs is added to the addition of the second most LSBs. The carry from this operation is passed on to the addition of the third most LSBs. In this way, the carry propagates from LSB to MSB.

In residue arithmetic the full data word is divided into several smaller words. The smaller words are coded such that an addition operation requires no carry between the coded words, although carry operations occur within the addition of words. In this manner the length of the carry path is reduced to allow higher computation speed at the cost of more complex hardware.

### 2.2.1 The Basics of Residue Arithmetic

The residue of an integer  $k$  modulo  $R_i$  is defined as the remainder when  $k$  is divided by  $R_i$ . The residue is denoted as  $k_i = \langle k \rangle_{R_i}$ , [6]. For example, the residue of 15 modulo 7 is 1 and the residue of 22 modulo 5 is 2.

The following residue properties exist for integers  $n$  and  $k$  [6]:

$$\begin{aligned} \langle nk \rangle_{R_i} &= \langle \langle n \rangle_{R_i} \langle k \rangle_{R_i} \rangle_{R_i} \\ &= \langle n \langle k \rangle_{R_i} \rangle_{R_i} \\ &= \langle \langle n \rangle_{R_i} k \rangle_{R_i} \quad , \end{aligned}$$

$$\begin{aligned}
\langle n + k \rangle_{R_i} &= \langle \langle n \rangle_{R_i} + \langle k \rangle_{R_i} \rangle_{R_i} \\
&= \langle n + \langle k \rangle_{R_i} \rangle_{R_i} \\
&= \langle \langle n \rangle_{R_i} + k \rangle_{R_i} .
\end{aligned} \tag{2.10}$$

Using these properties, it is possible to multiply and add integer values represented as residues and obtain a result in residue form.

If  $R_1, R_2, \dots, R_P$  are defined as relatively prime integers (integers that share no common factors except 1) then any integer  $k$  in the interval 0 to  $R - 1$ , where  $R = R_1 R_2 \dots R_P$ , can be uniquely represented by the residues [6],

$$\langle \langle k \rangle_{R_1}, \langle k \rangle_{R_2}, \dots, \langle k \rangle_{R_P} \rangle = \langle k_1, k_2, \dots, k_P \rangle \quad . \tag{2.11}$$

For example, consider the case for  $R_1 = 3$  and  $R_2 = 5$ . In such a case, the integers from 0 to

$$R - 1 = R_1 R_2 - 1 = 3 \times 5 - 1 = 14 \tag{2.12}$$

are uniquely represented. In Table 2.3 the unique residue pair for the integers  $k \in [0, 14]$  are shown.

The representation of the integer  $k$  in terms of its residues,

$$\langle k_1 k_2 \dots k_P \rangle \quad , \tag{2.13}$$

is closed under multiplication and addition operations, which means that all residue addition and multiplication operations on the ring  $[0, R - 1]$  will result in values also found on the ring  $[0, R - 1]$ .

By using the unique coding of an integer in residues, addition and multiplication operations can be performed on the residue of the integer using the multiplication

$k$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\langle k \rangle_{R_1}$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
$\langle k \rangle_{R_2}$	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4

Table 2.3: Unique residue pairs for  $k$  when  $R_1 = 3$  and  $R_2 = 5$ .

and addition properties of residues. The result of such operations, or sequences of operations, is a residue which must be converted to a natural integer. This can be accomplished using the Chinese Remainder Theorem [6]:

$$k = \left\langle \sum_{i=1}^P k_i \langle M_i^{-1} \rangle_{R_i} M_i \right\rangle_R, \quad (2.14)$$

where  $M_i = R/R_i$  and  $\langle M_i^{-1} \rangle_R$  is the multiplicative inverse of  $M_i$ :

$$\langle n^{-1} \rangle_R = \{m : \langle \langle n \rangle_R m \rangle_R = 1, 0 \leq m < R\} \quad (2.15)$$

For example, the multiplicative inverse of 3 modulo 5 is 2 because

$$\langle \langle 3 \rangle_5 \times 2 \rangle_5 = \langle 6 \rangle_5 = 1 \quad (2.16)$$

### 2.2.2 Filter Construction Using Residue Arithmetic

The use of residue arithmetic provides a means to divide the computation within a digital filter into multiple parallel paths. By dividing the computation into independent parallel paths, the maximum carry path length within the filter is reduced which allows computation at higher speeds. A similar increase in computation speed can be obtained using look ahead carry for adding operations, but residue arithmetic also allows increased speed in multiplication without great increases in hardware requirements.

To illustrate the construction of a residue arithmetic system, consider an example using the difference equation,

$$y = a_0 u(n) + a_1 u(n-1) \quad (2.17)$$

where the  $a_i$  are constant multiplier coefficients and  $u(n)$  is a variable integer input at discrete time  $n$ . If we choose the moduli set  $\{R_1, R_2, R_3\} = \{17, 19, 31\}$  we can form a ring from 0 to  $R_1 R_2 R_3 - 1 = 10013$ . This ring can represent slightly more

values than 13 bits of data. To allow the representation of negative values for  $u$  the residue operator,  $\langle \cdot \rangle_{R_i}$ , is redefined as [7],

$$\langle k \rangle_{R_i} = \begin{cases} \langle k \rangle_{R_i} & k \in \left[0, \frac{R-1}{2}\right] \\ R_i - \langle k \rangle_{R_i} & k \in \left[-\frac{R-1}{2}, 0\right) \end{cases} \quad (2.18)$$

The new residue operator allows 12 bits of data plus one sign bit. The multiplication coefficients are chosen arbitrarily for the example as

$$a_0 = 127 \quad (2.19)$$

$$a_1 = -201 \quad (2.20)$$

and are converted into the residue representation,

$$\begin{aligned} \langle a_0 \rangle &= \langle \langle 127 \rangle_{17}, \langle 127 \rangle_{19}, \langle 127 \rangle_{31} \rangle \\ &= \langle 8, 13, 3 \rangle \end{aligned} \quad (2.21)$$

$$\begin{aligned} \langle a_1 \rangle &= \langle \langle -201 \rangle_{17}, \langle -201 \rangle_{19}, \langle -201 \rangle_{31} \rangle \\ &= \langle 3, 8, 16 \rangle \end{aligned} \quad (2.22)$$

Using a hardware encoder the input signal  $u$  is converted into the residue representation,

$$\langle u(n) \rangle = \langle \langle u(n) \rangle_{17}, \langle u(n) \rangle_{19}, \langle u(n) \rangle_{31} \rangle \quad (2.23)$$

$$\langle u(n-1) \rangle = \langle \langle u(n-1) \rangle_{17}, \langle u(n-1) \rangle_{19}, \langle u(n-1) \rangle_{31} \rangle \quad (2.24)$$

Once the signals have been encoded into their respective residue representations (composed of three separate residues), each internal residue representation can be used in multiplication and addition operations without reference to any other residue representation. In general, Equation 2.17 becomes

$$\langle y \rangle_{R_i} = \left\langle \left\langle \langle a_0 \rangle_{R_i} \langle u(n) \rangle_{R_i} \right\rangle_{R_i} + \left\langle \langle a_1 \rangle_{R_i} \langle u(n-1) \rangle_{R_i} \right\rangle_{R_i} \right\rangle_{R_i}, \quad (2.25)$$

and for this example,

$$\langle y(n) \rangle_{17} = \langle \langle \langle a_0 \rangle_{17} \langle u(n) \rangle_{17} \rangle_{17} + \langle \langle a_1 \rangle_{17} \langle u(n-1) \rangle_{17} \rangle_{17} \quad (2.26)$$

$$\langle y(n) \rangle_{19} = \langle \langle \langle a_0 \rangle_{19} \langle u(n) \rangle_{19} \rangle_{19} + \langle \langle a_1 \rangle_{19} \langle u(n-1) \rangle_{19} \rangle_{19} \quad (2.27)$$

$$\langle y(n) \rangle_{31} = \langle \langle \langle a_0 \rangle_{31} \langle u(n) \rangle_{31} \rangle_{31} + \langle \langle a_1 \rangle_{31} \langle u(n-1) \rangle_{31} \rangle_{31} \quad (2.28)$$

Since the residues of  $a_0$  and  $a_1$  have been precalculated,

$$\langle y(n) \rangle_{17} = \langle \langle 8 \langle u(n) \rangle_{17} \rangle_{17} + \langle 3 \langle u(n-1) \rangle_{17} \rangle_{17} \rangle_{17} \quad (2.29)$$

$$\langle y(n) \rangle_{19} = \langle \langle 13 \langle u(n) \rangle_{19} \rangle_{19} + \langle 8 \langle u(n-1) \rangle_{19} \rangle_{19} \rangle_{19} \quad (2.30)$$

$$\langle y(n) \rangle_{31} = \langle \langle 3 \langle u(n) \rangle_{31} \rangle_{31} + \langle 16 \langle u(n-1) \rangle_{31} \rangle_{31} \rangle_{31} \quad (2.31)$$

The combination of the three equations operate on the unified ring from  $-\frac{R-1}{2}$  to  $\frac{R-1}{2}$ , and each individual equation operates on an individual ring

$$\left[ -\frac{R_i - 1}{2}, \frac{R_i - 1}{2} \right] \quad (2.32)$$

For this example the rings are:

$$\left[ -\frac{17-1}{2}, \frac{17-1}{2} \right] = [-8, 8] \quad (2.33)$$

$$\left[ -\frac{19-1}{2}, \frac{19-1}{2} \right] = [-9, 9] \quad (2.34)$$

$$\left[ -\frac{31-1}{2}, \frac{31-1}{2} \right] = [-15, 15] \quad (2.35)$$

Using residue arithmetic the computation of the difference equation, Equation 2.17, is split into three independent parallel paths shown in Figure 2.2 where T's represent delay elements. None of the three rings requires information from either of the other rings to compute its residue representation of the final value. It is only the final value that needs to be converted from its residue representation,  $\langle \langle y(n) \rangle_{17}, \langle y(n) \rangle_{19}, \langle y(n) \rangle_{31} \rangle$ , to an integer,  $y(n)$ , and this is accomplished using an implementation of the Chinese Remainder Theorem.

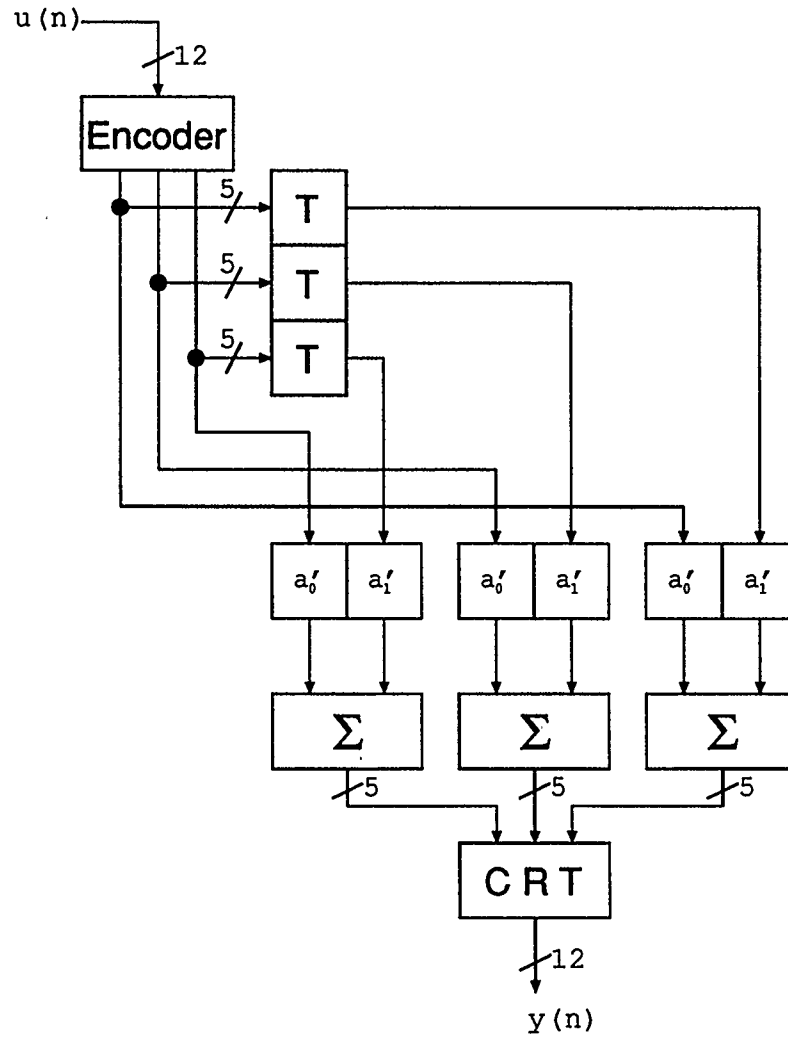


Figure 2.2: Block diagram of a residue arithmetic implementation.

The Chinese Remainder Theorem can be implemented as a hardware look-up table, or as a combination logic decoding device. As an alternative to these two implementations, one can take advantage of the sum of products operation involved in the Chinese Remainder Theorem (Equation 2.14) and implement it in a distributed arithmetic approach [7].

### 2.2.3 ROM Implementation Using Residue Arithmetic

In residue arithmetic implementations, the internal computations, such as addition and multiplication, are not costly in full ROM implementations. In this example,



when each multiplier is implemented as a ROM look-up table, a single ROM look-up table requires five input bits and five output bits for a total of  $5 \times 2^5 = 160$  bits per multiplier. The adder look-up tables are more costly than multiplier look-up tables. In this example, each ROM implementation of an adder look-up table requires ten input bits and five output bits, so the entire table requires  $5 \times 2^{10} = 5\,120$  bits. The implementation of the difference equation (Figure 2.2) requires six multipliers and three adders, so the internal calculations in the difference requires a total of  $6 \times 106 \text{ bits} + 3 \times 5\,120 \text{ bits} = 16\,320 \text{ bits}$ .

The memory required for internal computations is very small compared to the memory required for the conversion to and from the residue representation. The encoder of the system is a 13 bit input ROM with a 15 bit word ( $13 \times 2^{13} = 122\,880$  bits). The decoder (a look-up table implementing the Chinese Remainder Theorem) is a 15 bit input ROM with a 13 bit word ( $15 \times 2^{15} = 425\,984$  bits). The total ROM required for implementing the difference equation is

$$122\,880 + 425\,984 + 16\,320 = 565\,184 \text{ bits.} \quad (2.36)$$

This is the memory required for an implementation of a first order difference equation. If higher order equations are to be implemented many more internal computational components would be required which would increase the memory demands.

### 2.3 Conclusions on Look-up Table Implementations

It is the fundamental difference in the cost of components between ROM digital filter implementations and conventional digital filter implementations that necessitates a different approach to design for full ROM implementations. Conventional approaches concentrate on reducing hardware requirements or increasing the speed of multipliers. In ROM implementations, the hardware cost of implementing an adder is much

greater than the cost of a multiplier because adders require twice as many inputs as multipliers.

The distributed arithmetic and residue arithmetic structures explained in this chapter present the current use of ROM components in digital filters. In both approaches, part of the digital filter is encoded as a look-up table in ROM. A logical extrapolation is to encode the entire digital filter in a ROM look-up table. This approach will be examined in the next chapter.

## Chapter 3

# ROM Reduction by Elimination of Unreachable States

In chapter 1, the memory required to construct a digital filter in a block ROM state machine format was shown to be prohibitive. The construction of ROM based digital filters requires some method of reducing the required memory. One such method is to take advantage of the ROM look-up table's ability to rapidly compute complex calculations. All calculations are performed prior to the ROM implementation, so the look-up table can perform calculations as complex as desired with no speed penalty during operation.

When a filter is constructed as a block ROM state machine (Figure 1.1) the inputs to the ROM are separated into the states of the filter ( $x_1, x_2, x_3 \dots$ ) and the input ( $u$ ), but this is not required. The states can be combined into a super-state which represents the total present state of the filter, while the input ( $u$ ) and output ( $y$ ) remain coded as independent integers to maintain compatibility with the analog-to-digital and digital-to-analog converters.

The super-state is constructed by concatenating all the states together to form a single input word. In this manner, the bit patterns for each state are grouped together and ordered for ease of understanding, but this need not be the case. If the bit orders of the states are scrambled the ROM could still be programmed to function as a filter and the scrambled ROM filter's behaviour would be identical to the unscrambled block ROM implementation.

A ROM look-up table is programmed for input/output relationships. Very complex, or very simple input/output relationships require the same amount of memory,

so the ability to scramble the super-state allows the designer great freedom in the state space representation. The super-states could be placed unevenly throughout the state space of the filter to allow more accurate computation at desired points, but more importantly super-states can be eliminated to reduce the memory required for implementation.

A super-state representation records the present state of the filter. For any bounded input sequence some super-states will never be attainable and the union of these super-states is called the unreachable set [14]. The unreachable set is specific to the particular filter implementation and bound of the inputs.

An unreachable set implies the existence of a reachable set which is the set subtraction of the unreachable set from the state space. The reachable set must be coded into the ROM look-up table representation of the state space to allow the filter to function correctly, but the unreachable set has no impact on the filter's behaviour, since, under normal operating conditions, none of these super-states are ever encountered.

There is no necessity that the state variables in a block ROM implementation be isolated from each other. The states may be combined in a super-state which represents the current state of the filter. If the unreachable set is defined and eliminated from the state space, fewer super-states will be required to represent all the possible filter states. When the number of super-states is reduced, the size of the memory required to implement the filter's look-up table is reduced.

This chapter examines the memory reductions for block ROM filter implementations by eliminating the unreachable set. The results of filter simulations are provided and the memory requirement for a filter as a function of the filter order is determined.

### 3.1 Determining the Reachable Set

An  $N^{th}$  order digital filter constructed with separate adder and multiplier components will have a state space which is an  $N$ -dimensional box, called a hyperbox. Each state variable ( $x_i$ ) in the super-state vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} \quad (3.1)$$

defines an axis in state space from  $+A_i$  to  $-A_i$ , where  $2A_i$  is the dynamic range of the state variable  $x_i$ . Each axis is at right angles to all others, so an  $N^{th}$  order filter (possessing  $N$  states) will define an  $N$ -dimensional hyperbox containing all possible super-state vectors,  $\mathbf{x}$ . This  $N$ -dimensional hyperbox can be divided into two disjoint sets: the reachable set and the unreachable set. The reachable set is the union of all super-state vectors that can be arrived at from a restricted class of inputs, such as bounded energy or bounded amplitude inputs. The unreachable set is the set subtraction of the reachable set from the set of all possible vector states, and it is composed of the union of all super-state vectors that can not be arrived at using the restricted class of inputs.

A linear digital filter has a set of reachable state vectors which depend on the structure of the filter, and this reachable set has been determined for both bounded energy inputs [14],

$$\|\mathbf{u}(k)\|_2 = \left[ \sum_{j>0} u^2(k-j) \right]^{\frac{1}{2}} \quad (3.2)$$

and for bounded amplitude inputs [14],

$$|\mathbf{u}(k)| \leq 1 \quad (3.3)$$

To determine the reachable set, it is necessary to provide some basic equations for a state space description of a filter. The state space equations for a linear discrete time filter are defined as [6]:

$$\mathbf{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ \vdots \\ x_N(k) \end{bmatrix} \quad (3.4)$$

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{b}u(k) \quad (3.5)$$

$$y(k) = \mathbf{c}^T \mathbf{x}(k) + du(k) \quad (3.6)$$

where

$\mathbf{A}$  =  $N \times N$  state matrix

$\mathbf{b}$  =  $N$  dimensional column vector

$\mathbf{c}$  =  $N$  dimensional column vector

$d$  = scalar

$u(k)$  = scalar input at time interval  $k$

$\mathbf{x}(k)$  =  $N$  dimensional column super-state vector at time interval  $k$

$y(n)$  = scalar output at time interval  $k$

The next super-state of the system  $\mathbf{x}(k)$  can be found using knowledge of the previous inputs. Assuming the initial state is  $\mathbf{x}(k-1)$  and the previous input is  $u(k-1)$  then the next super-state  $\mathbf{x}(k)$  is

$$\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k-1) + \mathbf{b}u(k-1) \quad (3.7)$$

The super-state  $\mathbf{x}(k+1)$  can be computed with knowledge of the present input  $u(k)$  and the present super-state  $\mathbf{x}(k)$ ,

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{b}u(k) \quad .$$

By substituting Equation 3.7 into Equation 3.5, the super-state  $\mathbf{x}(k+1)$  can be computed from the prior inputs ( $u(k-1)$  and  $u(k)$ ) and the initial super-state, ( $\mathbf{x}(k-1)$ )

$$\mathbf{x}(k+1) = \mathbf{A} [\mathbf{A}\mathbf{x}(k-1) + \mathbf{b}u(k-1)] + \mathbf{b}u(k) \quad (3.8)$$

If the system begins in an initial rest super-state ( $\mathbf{x} = \mathbf{0}$ ) an arbitrary super-state  $\mathbf{x}(k)$  can be computed using the previous inputs to the system,

$$\mathbf{x}(k) = \mathbf{b}u(k-1) + \mathbf{A}\mathbf{b}u(k-2) + \mathbf{A}^2\mathbf{b}u(k-3) + \mathbf{A}^3\mathbf{b}u(k-4) \dots \quad (3.9)$$

which can be written as

$$\mathbf{x}(k) = [\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \mathbf{A}^3\mathbf{b}, \dots] \begin{bmatrix} u(k-1) \\ u(k-2) \\ u(k-3) \\ u(k-4) \\ \vdots \end{bmatrix} = \mathcal{F}\mathbf{u} \quad (3.10)$$

where  $\mathcal{F}$  is the controllability matrix [13] and  $\mathbf{u}$  is the input vector.

The covariance matrix ( $\mathbf{K}$ ) of the state is an  $N \times N$  matrix and can be written as

$$\mathbf{K} = [\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \dots] \begin{bmatrix} \mathbf{b}^T \\ (\mathbf{A}\mathbf{b})^T \\ (\mathbf{A}^2\mathbf{b})^T \\ \vdots \end{bmatrix} = \mathcal{F}\mathcal{F}^T \quad (3.11)$$

Although  $\mathbf{K}$  can be computed iteratively using Equation 3.11,  $\mathbf{K}$  may also be determined using the set of  $N$  linear equations [14],

$$\mathbf{K} = \mathbf{A}\mathbf{K}\mathbf{A}^T + \mathbf{b}^T\mathbf{b} \quad (3.12)$$

where  $\mathbf{K}$  is an  $N \times N$  matrix.

### 3.1.1 Reachable Set for Inputs with Bounded Energy

For bounded energy inputs the reachable set is easy to calculate [14]. A bounded energy input  $\mathbf{u}$  will have an energy defined by the  $L_2$  norm,

$$\|\mathbf{u}\|_2 = \left[ \sum_{j>0} u^2(k-j) \right]^{\frac{1}{2}} \quad (3.13)$$

Using Equation 3.10, the specific input vector  $\mathbf{u}_x$  required to reach a super-state vector  $\mathbf{x}$  is

$$\begin{aligned} \mathbf{x} &= \mathcal{F}\mathbf{u}_x \\ \mathbf{K}\mathbf{K}^{-1}\mathbf{x} &= \mathcal{F}\mathbf{u}_x \\ \mathcal{F}\mathcal{F}^T\mathbf{K}^{-1}\mathbf{x} &= \mathcal{F}\mathbf{u}_x \\ \mathcal{F}^T\mathbf{K}^{-1}\mathbf{x} &= \mathbf{u}_x \\ \mathbf{u}_x &= \mathcal{F}^T\mathbf{K}^{-1}\mathbf{x} \end{aligned} \quad (3.14)$$

For all input sequences that reach the super-state  $\mathbf{x}$ , the input  $\mathbf{u}_x$  will require the minimum amount of energy. The energy ( $\|\mathbf{x}\|_2^2$ ) required for an arbitrary input sequence  $\mathbf{u}$  is

$$\begin{aligned} \|\mathbf{x}\|_2^2 &= \mathbf{u}^T\mathbf{u} \\ &= \mathbf{x}^T\mathbf{K}^{-1}\mathcal{F}\mathcal{F}^T\mathbf{K}^{-1}\mathbf{x} \\ &= \mathbf{x}^T\mathbf{K}^{-1}\mathbf{K}\mathbf{K}^{-1}\mathbf{x} \\ &= \mathbf{x}^T\mathbf{K}^{-1}\mathbf{x} \end{aligned} \quad (3.15)$$

Vector  $\mathbf{u}_x$  requires the minimum energy to reach super-state  $\mathbf{x}$  because if another vector  $\mathbf{u}$  could reach  $\mathbf{x}$  ( $\mathbf{x} = \mathcal{F}\mathbf{u}$ ) in the same amount of time as  $\mathbf{u}_x$  then

$$\mathbf{u} = \mathbf{u}_x + (\mathbf{u} - \mathbf{u}_x) \quad (3.16)$$

In energy terms

$$\|\mathbf{u}\|_2^2 = \|\mathbf{u}_x + \mathbf{u} - \mathbf{u}_x\|_2^2$$



$$\|\mathbf{u}\|_2^2 = \|\mathbf{u}_x\|_2^2 + \|\mathbf{u} - \mathbf{u}_x\|_2^2 + \mathbf{u}_x^T(\mathbf{u} - \mathbf{u}_x) + (\mathbf{u} - \mathbf{u}_x)^T \mathbf{u}_x \quad (3.17)$$

but since

$$\begin{aligned} \mathbf{u}_x^T(\mathbf{u} - \mathbf{u}_x) &= \mathbf{x}^T \mathbf{K}^{-1} \mathcal{F}(\mathbf{u} - \mathbf{u}_x) \\ &= \mathbf{x}^T \mathbf{K}^{-1} (\mathcal{F}\mathbf{u} - \mathcal{F}\mathbf{u}_x) \\ &= \mathbf{x}^T \mathbf{K}^{-1} (\mathbf{x} - \mathbf{x}) \\ &= 0 \end{aligned} \quad (3.18)$$

and, for similar reasons,

$$(\mathbf{u} - \mathbf{u}_x)^T \mathbf{u} = 0 \quad (3.19)$$

Equation 3.17 becomes

$$\|\mathbf{u}\|_2^2 = \|\mathbf{u}_x\|_2^2 + \|\mathbf{u} - \mathbf{u}_x\|_2^2 \quad (3.20)$$

If this equation is rewritten as

$$\|\mathbf{u}\|_2^2 - \|\mathbf{u}_x\|_2^2 = \|\mathbf{u} - \mathbf{u}_x\|_2^2 \quad (3.21)$$

it is clear that

$$\|\mathbf{u}\|_2^2 \geq \|\mathbf{u}_x\|_2^2 \quad (3.22)$$

since

$$\|\mathbf{u} - \mathbf{u}_x\|_2^2 \geq 0 \quad (3.23)$$

This establishes that  $\mathbf{u}_x$  requires the minimum amount of energy to reach  $\mathbf{x}$ . All other inputs  $\mathbf{u}$  that can reach  $\mathbf{x}$  will require more energy,

$$\|\mathbf{u}\|_2^2 = \|\mathbf{u}_x\|_2^2 + \|\mathbf{u} - \mathbf{u}_x\|_2^2 \geq \|\mathbf{u}_x\|_2^2 \quad (3.24)$$

The knowledge of the minimum energy required to reach any super-state  $\mathbf{x}$  is used to determine which states can be reached by an energy bounded input,

$$\|\mathbf{u}\|_2^2 \geq \|\mathbf{u}_x\|_2^2 = \mathbf{x}^T \mathbf{K}^{-1} \mathbf{x} \quad (3.25)$$

or rearranged,

$$\mathbf{x}^T \mathbf{K}^{-1} \mathbf{x} \leq \|\mathbf{u}\|_2^2 \quad . \quad (3.26)$$

Only if the minimum energy required to reach a super-state  $\mathbf{x}$  is less than, or equal to, the maximum energy of the input  $\|\mathbf{u}\|_2^2$  will that state be reachable.

### 3.1.2 Reachable Set for Bounded Amplitude Inputs

The reachable set for a bounded amplitude input is more difficult to calculate [14]. Suppose  $\|u(k)\| \leq 1$  for all  $k$  (the condition for a bounded input). Using Equation 3.10, for any possible input  $\hat{\mathbf{u}}$

$$\mathbf{x} = \mathcal{F} \hat{\mathbf{u}} \quad . \quad (3.27)$$

By introducing an arbitrary row vector  $\boldsymbol{\rho}$  (representing an arbitrary direction in  $N$ -dimensional space),

$$\boldsymbol{\rho} \mathbf{x} = \boldsymbol{\rho} \mathcal{F} \hat{\mathbf{u}} \leq \|\boldsymbol{\rho} \mathcal{F}\|_1 = \sum_{k=0}^{\infty} |\boldsymbol{\rho} \mathbf{A}^k \mathbf{B}| \quad . \quad (3.28)$$

The bound is obtained by setting

$$u(k-j) = \text{sign} [\boldsymbol{\rho} \mathbf{A}^{j-1} \mathbf{B}] = \pm 1 \quad . \quad (3.29)$$

The set of all vectors  $\mathbf{x}$  satisfying the inequality

$$\boldsymbol{\rho} \mathbf{x} \leq \|\boldsymbol{\rho} \mathcal{F}\|_1 \quad (3.30)$$

is a region of space bounded by the plane

$$\boldsymbol{\rho} \mathbf{x} = \|\boldsymbol{\rho} \mathcal{F}\|_1 \quad . \quad (3.31)$$

There will be one such space for each vector  $\boldsymbol{\rho}$ . The reachable set is the intersection of all these spaces and this calculation is very computationally intensive because the calculation requires the computation of an infinite number of infinite sums.

The reachable set for bounded amplitude inputs is difficult to calculate, while the reachable set for bounded energy inputs is much easier to calculate. Roberts and Mullis [14] demonstrate that the shape of the reachable sets are similar and that the bounded amplitude input reachable set can be approximated by scaling the bounded energy input reachable set. To illustrate the similarity of the two reachable sets a direct form filter is examined in the following section.

### 3.1.3 A Direct Form Filter Example

#### Reachable Set for Bounded Amplitude Inputs

For bounded energy inputs, the reachable set is an  $N$ -dimensional ellipse, also called a hyperellipse. The hyperellipse must be contained by the hyperbox defined by the dynamic range of the inputs, which for  $N = 2$  (a 2-dimensional case) is an ellipse in a box.

Consider the  $\mathcal{Z}$ -domain transfer function of the direct form filter

$$H(z) = \frac{z^2}{z^2 - 2r \cos(\phi)z + r^2} \quad (3.32)$$

If the poles are located at  $r = 0.98$ ,  $\phi = 45^\circ$ ,

$$\begin{aligned} H(z) &= \frac{z^2}{z^2 - 2r \cos(\phi)z + r^2} \\ &= \frac{z^2}{z^2 + m_1 z + m_2} \end{aligned} \quad (3.33)$$

where  $m_1 = -1.96 \cos(45^\circ)$  and  $m_2 = 0.9604$ .

For a direct form realization the state matrix description is

$$\mathbf{A} = \begin{bmatrix} -m_1 & -m_2 \\ 1 & 0 \end{bmatrix} \quad (3.34)$$

$$\mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (3.35)$$

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \mathbf{B}u \quad (3.36)$$

For such a configuration the covariance matrix  $\mathbf{K}$  can be found using Equation 3.12,

$$\begin{aligned}\mathbf{K} &= \mathbf{A}\mathbf{K}\mathbf{A}^T + \mathbf{b}\mathbf{b}^T \\ -\mathbf{b}\mathbf{b}^T &= \mathbf{A}\mathbf{K}\mathbf{A}^T - \mathbf{K} \\ -\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} &= \begin{bmatrix} 0 & -m2 \\ 1 & -m1 \end{bmatrix} \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} -m1 & -m2 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}\end{aligned}\quad (3.37)$$

Since both  $m1$  and  $m2$  are known values, four linear equations with four unknown variables are produced

$$\begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 + (m1)^2 & m1 \cdot m2 & m1 \cdot m2 & (m2)^2 \\ -m1 & -1 & -m2 & 0 \\ -m1 & -m2 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} k_{11} \\ k_{12} \\ k_{21} \\ k_{22} \end{bmatrix}\quad (3.38)$$

which can be solved to determine the covariance matrix

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} = \begin{bmatrix} 25.7521 & 18.2058 \\ 18.2058 & 25.7521 \end{bmatrix}\quad (3.39)$$

Using Equation 3.26 the reachable set for bounded energy inputs is

$$\begin{aligned}\|\mathbf{u}\|_2^2 &\geq \mathbf{x}^T \mathbf{K}^{-1} \mathbf{x} \\ \|\mathbf{u}\|_2^2 &\geq [x_1 \ x_2] \begin{bmatrix} 25.7521 & 18.2058 \\ 18.2058 & 25.7521 \end{bmatrix}^{-1} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \|\mathbf{u}\|_2^2 &\geq 0.0776x_1^2 - 0.1098x_1x_2 + 0.0776x_2^2\end{aligned}\quad (3.40)$$

The boundary of the reachable set defined by Equation 3.40 is shown in Figure 3.1. The interior of the ellipse represents all reachable states for the direct form filter represented by Equation 3.33. When the reachable set is enclosed in the minimum size box containing all possible super-state vectors the reachable set accounts for approximately 32% of the possible super-states defined by the box.

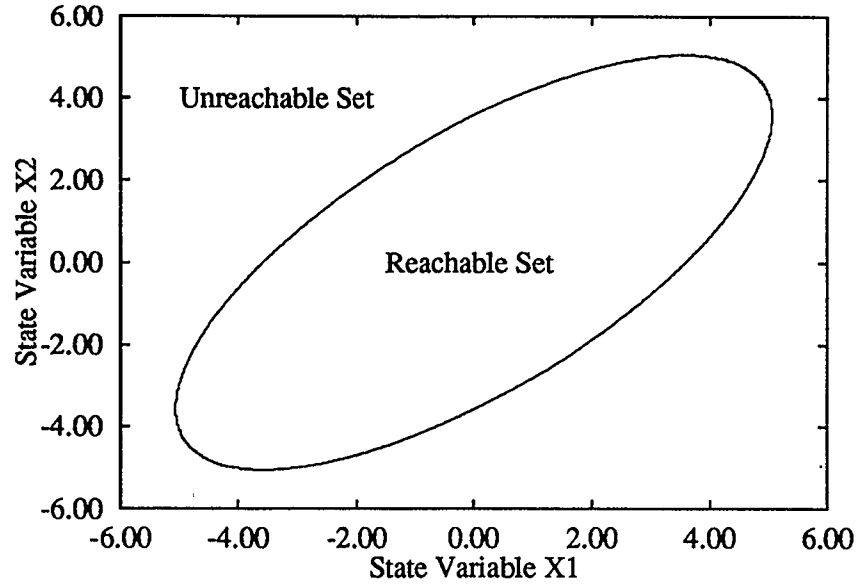


Figure 3.1: The direct form digital filter reachable set for  $\|\mathbf{u}\|_2^2 \leq 1$ .

### Reachable Set for Bounded Amplitude Inputs

An approximation to the bounded amplitude set can be determined using the state equation presented in Equation 3.36,

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \mathbf{B}\mathbf{u} \quad .$$

The state equation is applied to the bounded amplitude reachable set bound defined by Equation 3.31,

$$\begin{aligned} \rho\mathbf{x} &= \|\rho\mathcal{F}\|_1 \\ \rho\mathbf{x} &= \sum_{k=0}^{\infty} |\rho\mathbf{A}^k\mathbf{B}| \end{aligned} \quad (3.41)$$

producing

$$\rho\mathbf{x} = \sum_{k=0}^{\infty} \left| \rho \begin{bmatrix} -m1 & -m2 \\ 1 & 0 \end{bmatrix}^k \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right| \quad . \quad (3.42)$$

Equation 3.42 defines a set of lines which separate the state space into two disjoint sets. The boundary line belongs to the set containing the origin and this set is a

super set of the reachable set. The intersection of all the super sets produced by all possible vectors  $\rho$  is the reachable set.

The direct computation of the exact reachable set is not possible. There are an infinite number of arbitrary row vectors  $\rho$  and each half-space bound defined by Equation 3.42 requires the computation of an infinite sum. The reachable set (shown in Figure 3.2) is approximated by using 784 row vectors equally distributed in direction and truncating the infinite sum to 301 terms.

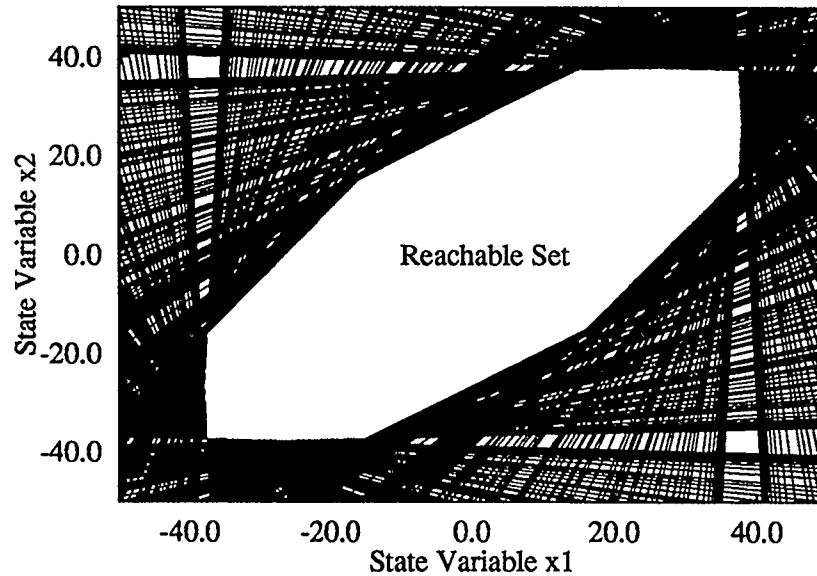


Figure 3.2: Bounded amplitude reachable set for the direct form filter.

When Figure 3.1 and Figure 3.2 are compared, the similarity in shape of the two reachable sets allow the approximation of the bounded amplitude set (which is difficult to calculate) using the bounded energy set (which is easier to calculate). To perform the approximation the bounded energy ellipse is scaled so that it encloses the entire bounded amplitude set. In this approach only the general form of the bounded amplitude set need be calculated. Detailed calculations, which would involve thousands of steps, need not be performed.

The potential for reducing the state space via the elimination of unreachable states has been demonstrated. In the next section the savings as a function of the filter order will be examined.

### 3.2 The Relationship Between the Fraction of Space Used and the Filter Order.

To evaluate the advantage offered by the reduced state space representation the space savings must be calculated. The space saving is a function of the filter order and the filter implementation. In this section, the maximum space used by an  $N^{th}$ -order filter is determined.

It was shown in the previous section that the shape of the reachable set can be approximated by a hyperellipse. The best space filling hyperellipse for a hyperbox is one aligned with the axis [14]. Using this knowledge the maximum percent of used space in an  $N$ -dimensional filter can be easily calculated for any order,  $N$ .

The generalized  $N$ -dimensional volume is called content, and the content of a hyperellipse is [5]

$$V_{hyperellipse} = \frac{2r_1 r_2 \cdots r_N \pi^{\frac{1}{2}N}}{N\Gamma(\frac{1}{2}N)} \quad (3.43)$$

where  $N$  is the dimension of the hyperellipse,  $r_i$  is a semi-axis of the ellipse in direction  $i$  and  $\Gamma(\cdot)$  is the Gamma function [1]:

$$\Gamma(z) = \lim_{n \rightarrow \infty} \frac{n! n^z}{z(z+1) \cdots (z+n)} \quad (3.44)$$

The content of an  $N$ -dimensional hyperbox is

$$V_{hyperbox} = a_1 a_2 \cdots a_N \quad (3.45)$$

where  $a_i$  is the length of one side of the hyperbox.

The fraction of space used by a hyperellipse in a hyperbox is equal to the fraction of super-states which will lie within the reachable set. The fraction of space used by the hyperellipse in a hyperbox decreases as the order of the space increases,

$$\begin{aligned} V_{used} &= \frac{V_{hyperellipse}}{V_{hyperbox}} \\ &= \frac{2r_1 r_2 \cdots r_N \pi^{\frac{1}{2}N} / N\Gamma(\frac{1}{2}N)}{a_1 a_2 \cdots a_N} \end{aligned} \quad (3.46)$$

When the hyperellipse enclosed within the hyperbox is a maximum area the diameter of the hyperellipse in direction  $i$  must be equal to the length of the side of the hyperbox in direction  $i$ . In other words  $a_i = 2r_i$ , so

$$\begin{aligned} V_{used} &= \frac{2r_1 r_2 \cdots r_N \pi^{\frac{1}{2}N} / N\Gamma(\frac{1}{2}N)}{2r_1 2r_2 \cdots 2r_N} \\ &= \frac{2\pi^{\frac{1}{2}N} / N\Gamma(\frac{1}{2}N)}{2^N} \\ &= \frac{\pi^{\frac{1}{2}N}}{N\Gamma(\frac{1}{2}N)2^{(N-1)}} \end{aligned} \quad (3.47)$$

Figure 3.3 shows that as the order of the filter increases the fraction of state space used decreases. By 5<sup>th</sup> order only 16.4% of the available super-states are reachable, and by 7<sup>th</sup> order only 3.70% of the defined super-states are reachable.

This analysis is only valid for aligned hyperellipses in hyperboxes. An arbitrary filter structure will possess a reachable set that is a hyperellipse, but that hyperellipse may not be aligned with the axis of the hyperbox (as the example in Section 3.1.3 demonstrated). The above analysis is the best case, and most filters will have larger unreachable super-state sets. Thus the exact fraction of possible super-state vectors that are reachable depends on the filter structure.

### 3.3 Reducing the Unreachable Set

The previous section demonstrated that large amounts of available state space is unused (the unreachable super-state set). If the reachable super-state set is small



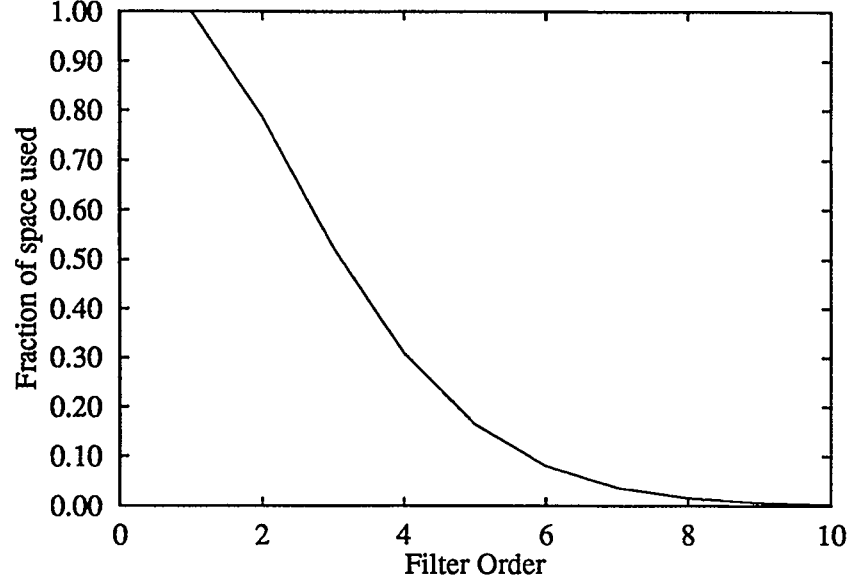


Figure 3.3: Maximum fraction of space used as a function of order for elliptical reachable sets.

then the number of input bits to the filter required to represent the states can be reduced. The number of bits that can be removed for a specific fraction of used state space is

$$b_{\text{removed}} = \left\lfloor \frac{-1 \log(V_{\text{used}})}{\log(2)} \right\rfloor \quad (3.48)$$

where  $\lfloor x \rfloor$  is the floor function of  $x$ :

$$\lfloor x \rfloor = \max\{i \in \mathcal{I} : i \leq x\} \quad (3.49)$$

Removing these bits will remove a large section of the unused state space and still maintain the same accuracy internal to the filter. The number of bits removed ( $b_{\text{removed}}$ ) is the total number of bits removed from the representation of the super-state. For instance, if the filter is originally designed with  $b$  bit wordlengths for all states in an  $N^{\text{th}}$  order filter the total number of bits required to represent the super-state is  $Nb$ . When the unused portion of the state space is eliminated by bit removal, the new number of bits required to represent the super-state is  $Nb - b_{\text{removed}}$ , so the

bit reduction per state variable would be  $(Nb - b_{removed})/N$ . The value of individual state variables is no longer directly accessible, but each reachable super-state of the filter is coded uniquely.

The remaining unused state space can be taken advantage of by remapping the super-state transitions in the filter. By representing only the reachable set, a larger number of super-states within the reachable set are created, which increases the internal accuracy of the filter. An alternative approach is to use the excess super-states to increase the bound of the input.

In a second order system, an elliptical reachable set with axes which are aligned with the axes of the state space hyperbox has a reachable set composed of 78.5% of the total state space. The unused state space (21.5%) is insufficient to reduce the super-state representation by an integer number of bits. The super-states in the unreachable set will be wasted unless the state space is mapped such that the entire represented state space lies within the reachable set. If the wordlength of the two state variables in the second order system is  $b$  the total number of representable vectors in the state space is  $2^{2b}$ . If all  $2^{2b}$  vectors are used to describe super-states within the reachable set (leaving none to represent super-states outside the reachable set) the internal resolution of the filter will be improved because all the available vectors are used to represent a smaller state space: the reachable set.

Alternatively, the unused super-states can be used to increase the amplitude bound of the input. Rather than mapping all super-states into the reachable set, the reachable set is expanded until the reachable set requires all available super-states. Both approaches require a remapping of the super-state variables, but the expansion of the reachable set leaves the distance between states equal to the distance prior to remapping. The new filter retains the accuracy of the original filter, but an input of larger amplitude may be accommodated since the reachable set has been expanded.

### 3.4 Simulation and Analysis

To test the remapping concept, a second order direct form digital filter has been simulated. The filter in this example is the same filter used in Section 3.1.3. Assuming a constant wordlength  $b$  for the input, output and all state variables a total of  $3b$  input lines and  $3b$  output lines to the ROM are required.

The filter's reachable set is not aligned with the axis of the hyperbox, so the reachable set will be less than the maximum reachable set of 78.5%. The unreachable set was determined to be 69% of the total state space. Since only 31% of the state space is used by the filter,

$$b_{removed} = \left\lfloor \frac{-1 \log(0.31)}{\log(2)} \right\rfloor = 1, \quad (3.50)$$

so one input bit can be removed from the representation of the state variables. This leaves  $b$  lines into the ROM representing the quantized input, and  $2b - 1$  lines into (and out of) the ROM to represent the filter state. The output requires  $b$  lines to maintain a standard binary coded decimal representation of the output to allow easy interfacing with a digital-to-analog converter. In total, the ROM look-up table requires  $3b - 1$  addressing lines (inputs) and  $3b - 1$  data lines (outputs).

After bit reduction,  $1 - 0.31/0.50 = 38\%$  of the remaining state space is unused. The vectors representing the super-states in this unreachable space are remapped into the reachable set increasing the internal resolution of the filter. Each square quantization interval in the 2-dimensional state space is reduced by 38%. The higher internal resolution allows a closer approximation of the ideal output. Alternatively, the system can allow larger input/output signals by maintaining the same super-state density and increasing the area of the reachable set.

In the conventional filter, a state variable has a dynamic range from  $-A$  to  $A$ . In this simulation  $A = 10$  and the dynamic range of the filter variables is  $[-10, 10]$ .

The quantization interval of the input, and output is

$$q = \frac{2A}{2^b} \quad (3.51)$$

(where  $b$  is the wordlength), while the quantization interval internal to the remapped filter is

$$q_{internal} = \frac{2A}{2^b} \sqrt{V_{used}} \quad (3.52)$$

For the direct form example:

$$q = \frac{2A}{2^b} = \frac{2 \times 10}{2^b} = \frac{20}{2^b} \quad (3.53)$$

$$q_{internal} = \frac{2A}{2^b} \sqrt{V_{used}} = \frac{2 \times 10}{2^b} \sqrt{0.62} = \frac{15.748}{2^b} \quad (3.54)$$

The tested system, simulated in software, is shown as a block diagram in Figure 3.4. The input  $u$  is the quantized input sequence  $b$  bits wide. The output  $y$  is the quantized output sequence, also  $b$  bits wide. The super-state is coded into  $2b - 1$  bits and is fed back into the ROM through a parallel unit sample delay. The dashed box in the diagram represents the ROM block and inside the ROM block is an equivalent diagram of the ROM programming.

The super-state is decoded into the state variables  $x_1$  and  $x_2$ . In combination with the input, the state variables are fed into the digital filter's algorithm to update the

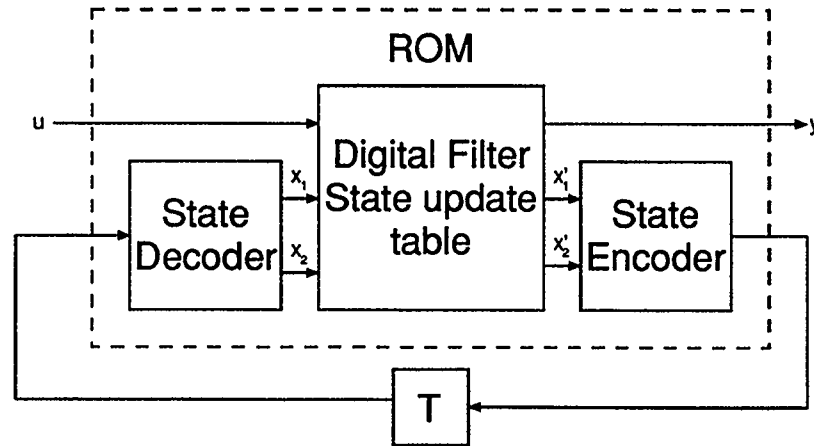


Figure 3.4: Block diagram of the reduced full ROM implementation.

state variables and output. The updated state variables  $x'_1$  and  $x'_2$  are then encoded into a  $2b - 1$  bit super-state. The input and output are not encoded to allow direct connection with standard analog-to-digital and digital-to-analog converters.

The error in the system's output as a fraction of the error in an unmodified digital filter with identical wordlength is shown as a function of wordlength in Figure 3.5. Both systems are stimulated with a Kronecker delta input and their responses are compared with an ideal response (calculated with double precision floating point accuracy). The maximum error encounter in each system for each wordlength is recorded as the ratio,

$$\frac{\text{maximum reduced filter error}}{\text{maximum direct form error}} \quad (3.55)$$

Fractions above 1.0 indicate a higher error in the reduced full ROM implementation than in the standard direct form implementation, while fractions lower than 1.0 indicate a lower error.

For small wordlengths ( $b \leq 3$ ) the two systems have identical maximum errors because the systems have too few states to take advantage of any mapping. For larger wordlengths ( $b \geq 4$ ), the full ROM implementation consistently displays a smaller error. It should be noted that, although, the reduced full ROM implementation seems to have a smaller error, the maximum possible error in the output for both systems is identical because both systems quantize the output in identical manners. The lower error rate of the reduced full ROM implementation reflects slower accumulation of errors from internal calculations. Internal calculations are more accurate because the quantization interval is 38% smaller.

The same error comparison can be done for the fast Fourier transform (FFT) of the impulse response (Figure 3.6). The maximum error between the ROM implementation FFT and the ideal FFT vs. the maximum error between the standard implementation FFT and the ideal FFT is graphed as a ratio. Again ratios above

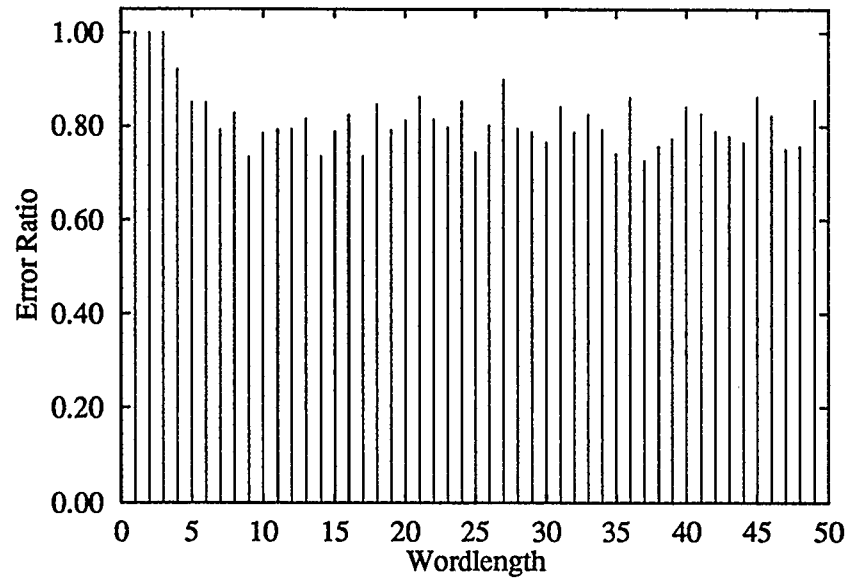


Figure 3.5: The ratio of maximum error between the ROM filter and the direct form filter as a function of wordlength,  $b$ .

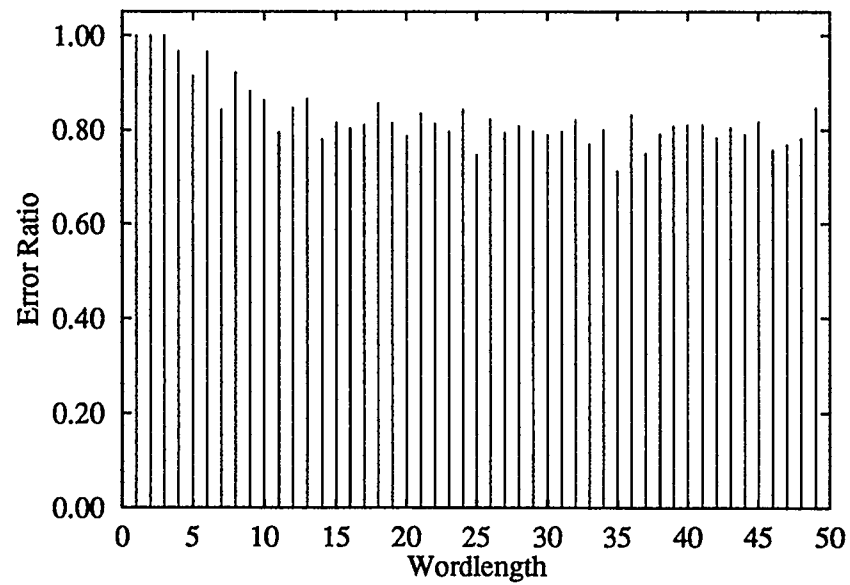


Figure 3.6: Maximum error in the FFT of the output as a function of wordlength,  $b$ .

1.0 indicate the maximum error is greater in the ROM based system, while ratios below 1.0 indicate the maximum error is lower in the ROM based system. The graph indicates that the FFT obtained from the ROM based filter has a lower error with respect to the ideal FFT than the standard implementation. The lower error in the ROM based filter are due to a slower accumulation of errors during operation.

### 3.5 Conclusions

Reducing the number of input bits to the ROM reduces the memory requirements. Every bit removed from the super-state representation will reduce the ROM size by 50%. Unfortunately, as the filter order increases, the amount of unused state space increases much more slowly than the memory requirements. The unreachable super-state set increases with each additional state variable and each state variable will add  $b$  bits to the input of the ROM. The addition of several state variables *may* allow the decrease in the new ROM size by one bit. The technique slows the increase in ROM memory requirements as the filter order increases, but the memory requirements still grow exponentially.

The restriction to representing only the reachable super-state set may be useful for implementations other than ROM, but the computations are difficult because the next state is no longer a linear function of the present state. The high computational power of the ROM enables such complex calculations to be done easily.

This method is a trade-off between memory size and computational complexity. A full ROM implementation requires a great deal of memory, while allowing very complex computations. By taking advantage of this inherent feature, the memory required is reduced and/or the accuracy can be improved.

The restriction of states to only the reachable set reduces the memory required, but does not allow construction of filters because the memory requirements are still very large.



## Chapter 4

### Filtering of Delta Modulated Signals

In the previous chapter, the memory requirement for ROM based digital filter implementations was reduced by eliminating the unreachable super-states from the represented state space. An alternative approach is to reduce the dynamic range of the states to reduce the wordlength of the states. Reductions in wordlength will reduce the memory requirements of the block ROM implementation.

The wordlength of the filter states can be reduced by using delta modulation and sigma-delta modulation. In this chapter, both methods will be explained in detail to provide appropriate background information for the next chapter in which a filtering system based on difference signals is introduced.

Delta modulation and sigma-delta modulation can potentially reduce the system wordlength to one bit. If reduced word length signals can be filtered, substantial reductions in memory requirements for ROM based filters are possible since the wide parallel state inputs to the ROM components are avoided.

Filtering delta modulated signals has been neglected as a method of signal processing due to its late development compared to pulse code modulated systems [10]. In spite of its slow start, several methods of filtering delta modulated signals have been developed [8, 9, 11, 19, 20, 21] and these approaches will be examined in this chapter. In addition to the filtering of delta modulated signals, attempts have been made to digitally filter sigma-delta modulated signals [18], and this approach will also be examined.

## 4.1 Delta Modulation

Delta modulation is an efficient means of encoding analog signals as narrow word-length digital signals. In delta modulation, an analog signal is encoded as a binary pulse stream based on the change in the amplitude of the analog input signal, rather than the amplitude of the signal. The input signal is encoded by a modulator which quantizes the difference between the present input signal and the reconstructed input signal as a  $+\delta$  or a  $-\delta$ , where  $\delta$  is the quantization step interval. Since only two values are required, both can be transmitted on a binary transmission line using 1s and 0s, with 1s representing  $+\delta$  and 0s representing  $-\delta$ . At the recovery end, the signal is reconstructed using a demodulator, which is an integrator. The combination of the delta modulator and the integrator provide an inexpensive method of analog to digital and digital to analog conversion.

## 4.2 The Basics of Delta Modulation Systems

A delta modulation system can be divided into two distinct components: a modulator and a demodulator. The modulator (Figure 4.1) compares the input signal to a reconstruction of the input signal and transmits the error, which is quantized to  $+\delta$  or  $-\delta$ . The value of  $\delta$  depends on the minimum step size allowable, which can be found for bandwidth limited input signals.

The minimum step size allowable is the maximum possible change,  $\Delta_{max}$ , in amplitude of the input signal. The input signal that changes most rapidly is the highest frequency component of the input signal. Let the highest frequency be designated as  $f_{max}$ . The highest frequency signal can be represented as

$$x = A \sin(2\pi f_{max}t) \quad , \quad (4.1)$$

where  $A$  is the amplitude of the input signal and  $t$  is time.

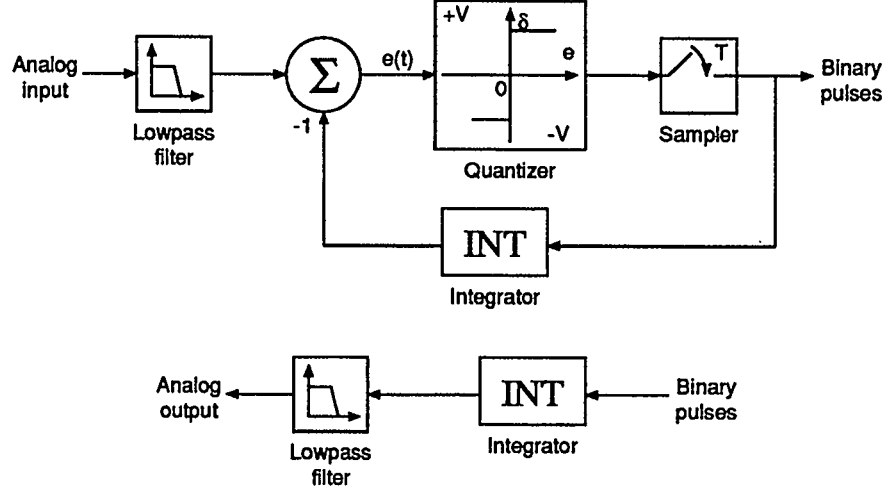


Figure 4.1: A basic delta modulation system .

The change in the signal  $x$  as a function of time is

$$\frac{dx}{dt} = A2\pi f_{max} \cos(2\pi f_{max}t) \quad (4.2)$$

The maximum slope of the input signal is  $A2\pi f_{max}$  since  $\cos(2\pi f_{max}t)$  will only vary between  $-1$  and  $+1$ . The maximum change in the input signal during one sample period  $T_s$  is

$$\begin{aligned} \Delta_{max} &= \max[A2\pi f_{max} \cos(2\pi f_{max}t)T_s] \\ &= \max[A2\pi f_{max}T_s] \max[\cos(2\pi f_{max}t)] \\ &= A2\pi f_{max}T_s \\ &= \frac{A2\pi f_{max}}{f_s} \quad , \end{aligned} \quad (4.3)$$

where the sampling rate  $f_s = \frac{1}{T_s}$ . Equation 4.3 implies that for a bandwidth limited signal there will be a maximum step size for a given sample rate. If the quantization step size  $\delta$  is larger than the maximum change,  $\Delta_{max}$ , in the input signal then the input signal can be encoded using the delta modulator.

Once the signal is encoded it can be transmitted as a binary signal. When the binary signal is received the analog signal must be reconstructed via a demodulator.

In delta modulation, the demodulator is an integrator. The integration of the error signal will reconstruct the input signal. Reconstruction depends on knowledge of the initial conditions. If the system begins in an unknown state, the output may contain a dc error. When both the encoder and decoder begin in a known identical state the dc error is avoided. Provided  $\delta \geq \Delta_{max}$ , the error in the quantized signal will always be less than or equal to  $\delta$ .

To convert the digital signal to an analog signal a digital to analog (D/A) converter is used. After the digital signal is converted to a quantized signal, via the D/A converter, a unity gain, low pass filter is used to smooth the quantized signal into an analog signal.

In the following sections, filtering methods for delta modulation signals are explained. Early attempts with hybrid analog-digital techniques are presented because the inexpensive advantages of analog-to-digital and digital-to-analog conversion are demonstrated. Subsequent techniques use all digital filtering methods with both binary and ternary delta modulation encoding systems.

### 4.3 Delta Modulation Filtering using Analog-digital Hybrid Techniques

Lockhart used a hybrid of analog and digital techniques to process delta modulation signals [10] . Although this technique is not fully digital, it provides insight into the development of delta modulation signal processing, and emphasizes the economy of analog-to-digital and digital-to-analog conversion in delta modulation systems.

The arithmetic operations on the signal are performed in the continuous domain, while the signal is coded at the input in the digital domain.

### 4.3.1 Non-Recursive Filtering

By using the binary transversal filter [11], a non-recursive filter structure for delta modulation can be constructed (Figure 4.2). The binary transversal filter is a hybrid of digital and analog technology. The time delays ( $T$ ) in the filter are achieved using digital flip-flops, while the multiplication and summation operations are achieved in analog technology. The resistors determine the multiplication constants or weights  $h_r$ . The weighted delays are summed, and integrated (using operational amplifier circuits) to produce the output signal.

By using  $M$  resistors and neglecting overload and quantization noise effects, the integrated output is

$$y(nT) = \sum_{r=0}^{M-1} h_r x[(n-r)T] \quad (4.4)$$

In the transfer function of  $z$ -domain Equation 4.4 is

$$H(z) = \sum_{r=0}^{M-1} h_r z^{-r} \quad (4.5)$$

These are standard equations for finite impulse response (FIR) filters and the multiplication coefficients  $h_r$  can be determined by standard methods for FIR filters [11].

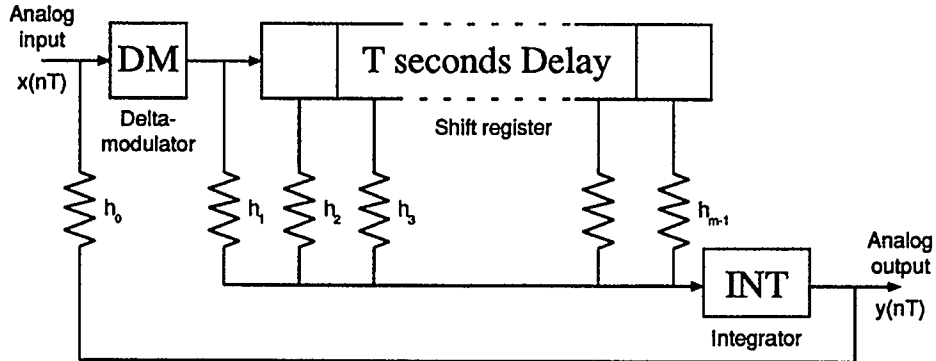


Figure 4.2: Non-recursive delta modulation filter [10] .

### 4.3.2 Recursive Filtering

Recursive filtering is accomplished by adding a feedback loop to the filter (Figure 4.3). The recursive delta modulation filter implements transfer function poles in addition to zeros in the transfer function. For the same selectivity as a non-recursive filter, a recursive filter requires a lower order.

A difficulty with such a recursive structure is that feedback of the weighted signal tends to overload the delta modulator. When the change in the modulator input signal (produced by the subtraction of the feedback signal from the filter input signal) exceeds the maximum allowable change  $\delta$ , the delta modulator will be unable to represent the signal. To avoid the overloading tendency, only small weighting coefficients can be used and this restricts the range of realizable filters.

The  $W_i$  coefficients form low-pass filter structures, whose outputs are the inputs to the main recursive filter structure through  $b_i$ . The lowpass filter must be designed with a gain of less than one, to avoid affecting the overall transfer function of the recursive filter. If such filters are used, the coefficients  $(b_1, b_2, \dots)$  can be specified by conventional recursive design methods [6].

### 4.3.3 The Possibility for ROM Implementation

In hybrid analog-digital circuits, the disadvantages of both the analog and the digital components must be examined. The resistors (used for weighting the digitally time delayed signals) will be affected by temperature and thermal noise. Over time, the resistance may change causing the filter response to change. An all digital implementation has an advantage over an all analog or an analog-digital technique since the components are very temperature insensitive, and aging has no effect.

As the analog components change over time, the circuit may slowly alter until the specifications originally stated no longer apply. By contrast, the digital components

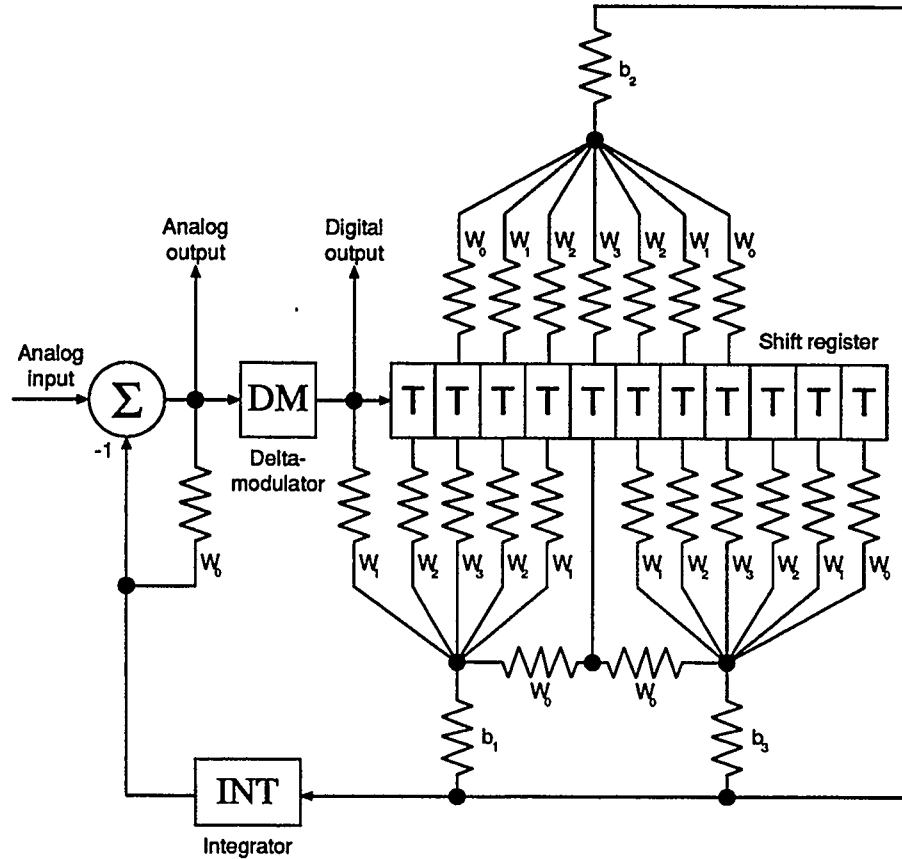


Figure 4.3: Recursive delta modulation filter [10].

fail in a catastrophic manner. If one of the flip-flops (used as time-delay elements for the delta modulated signal) fails the transfer function of the filter will be radically changed. In addition to these disadvantages of the hybrid filter, the structure imposes limitations on the coefficient weights. Only structures with small valued recursive coefficients can be constructed to avoid overloading the input delta modulator. The limits imposed on the coefficient values limits the possible filter transfer functions.

This early attempt at delta modulation filtering has no possibility of integration into ROM since it relies on analog components. The resistors in the delta modulation filter dictate the values of the multipliers. The resistors determine the gain of each time delayed signal in a summing amplifier which converts the digital signal into a continuous signal. The continuous signal is integrated to produce the output analog

waveform.

Since the output of the system is a continuous time signal, a digital-analog converter is not required if an analog signal is desired. This can be a great advantage in filter designs, but if the desired output is a delta modulated signal a second delta modulator would be required to convert the output of the filter to the digital form.

The inherent conversion of the resultant signal to analog is useful in applications where digital-to-analog conversion is necessary, because delta modulation provides an inexpensive conversion to digital form. Then the digital operations and filtering required can occur before conversion to analog form is accomplished. The resulting signal is analog and requires no additional processing.

## 4.4 Digital Filtering of Delta Modulated Signals

A full digital implementation of a filter for delta modulated signals was achieved by Kouvaras [8]. Using simple digital logic components, he created a delta-adder as a basic building component. The delta adder enabled the creation of a multiplier and finally a limited range of digital filters for delta modulated signals.

### 4.4.1 The Delta Adder

Delta modulated signals are composed of  $-1$ s and  $+1$ s, representing  $-\delta$  and  $+\delta$  respectively. When two such signals are summed the result is not representable as a delta modulated signal. Consider two delta modulated signals  $x_1$  and  $x_2$ . Any addition of the two signals will always produce an unrepresentable number:



$$x_1 + x_2 = x_3$$

$$-1 + -1 = -2$$

$$-1 + 1 = 0$$

$$1 + -1 = 0$$

$$1 + 1 = 2$$

The resulting signal  $x_3$  is composed of three states, none of which may be represented by the states used in  $x_1$  and  $x_2$ . Kouvaras solved this problem by creating a delta adder whose output is the sum of the two inputs divided by two:

$$y = \frac{x_1 + x_2}{2} \quad (4.6)$$

This allows the addition of two identical inputs to be expressed without error,

$$\frac{-1 + -1}{2} = -1 \quad (4.7)$$

$$\frac{1 + 1}{2} = 1 \quad , \quad (4.8)$$

but an addition that sums to zero,

$$\frac{-1 + 1}{2} = 0 \quad , \quad (4.9)$$

will introduce an error. Only  $-1$  and  $1$  may be used to represent the result of the addition, so a sum of  $0$  must be approximated with either a  $1$  or a  $-1$  which represents a significant error. If such an approximation occurred and the delta modulated signal was reconstructed the entire signal after the error point would have an offset error of  $\delta$ . If several errors occurred, the reconstructed signal could have large offsets errors.

To avoid such errors a carry is introduced and the adder is implemented according to,

$$S_n = 2^{-1}(X_n + Y_n - (1 - X_n Y_n)C_{n-1}) \quad (4.10)$$

$$C_n = X_n Y_n C_{n-1} \quad (4.11)$$

$$C_{n-1} = +1 \text{ or } -1 \quad (4.12)$$

where  $n = \dots, -1, 0, 1, \dots$ ;  $S_n$  is an approximation to the half sum and  $C_n$  is the carry. In such an adder, an error is compensated for at the following interval by storing the error in the carry and including the carry in the following addition operation. The entire input-output relationship is summarized in Table 4.1.

By introducing the carry, the error at the output of the adder is greater than the error in the input delta modulated signal, but the possibility of an accumulation of long term errors is eliminated. Such a system allows the error in the output of the delta adder ( $\epsilon_S$ ) to be [8],

$$\epsilon_S = \frac{\epsilon_Y + \epsilon_X}{2} + \varphi \quad (4.13)$$

$$\varphi \in \{-\delta, 0, +\delta\} \quad , \quad (4.14)$$

where  $\delta$  is the step quantization interval and  $\epsilon_i$  is the error during the quantization of input signal  $i$ , and  $\varphi$  is an error introduced by the carry. By using rounding, the error will always be less than  $2\delta$  [8].

#### 4.4.2 Delta Multiplier

Since the delta adder incorporates a multiplication by two components, the adder may be used as a building block to construct a multiplier, provided the multiplier coefficient is between 0 and 1. Any multiplier coefficient between 0 and 1 can be thought of as the summation of the signal multiplied by various  $2^{-i}$  terms. For

$X_n$	$Y_n$	$C_{n-1}$	$S_n$	$C_n$
-1	-1	-1	-1	-1
1	-1	-1	-1	1
-1	1	-1	-1	1
1	1	-1	1	-1
-1	-1	1	-1	1
1	-1	1	1	-1
-1	1	1	1	-1
1	1	1	1	1

Table 4.1: All possible inputs with outputs to the delta adder .

example, if the coefficient is  $m = 0.1011_b$  (in binary) or 0.6875 (decimal), then the multiplication can be accomplished by successive add-divide-by-two operations. Consider  $X$  to be an arbitrary delta modulated signal. The multiplication of  $X$  by  $m$  is accomplished by

$$X \times m = X \times 0.6875 \quad (4.15)$$

$$= X \times 0.1011_b \quad (4.16)$$

$$= X \left( \frac{1}{2} + \frac{1}{2^3} + \frac{1}{2^4} \right) \quad (4.17)$$

$$= X \frac{1}{2} + X \frac{1}{2^3} + X \frac{1}{2^4} \quad (4.18)$$

$$= \frac{1}{2} \left( X + X \frac{1}{2^2} + X \frac{1}{2^3} \right) \quad (4.19)$$

$$= \frac{1}{2} \left( X + \frac{1}{2} \left( X \frac{1}{2} + X \frac{1}{2^2} \right) \right) \quad (4.20)$$

$$= \frac{1}{2} \left( X + \frac{1}{2} \left( 0 + \frac{1}{2} \left( X + \frac{1}{2} (X + 0) \right) \right) \right) \quad (4.21)$$

The zero signal is approximated using an alternating signal called an idling sequence,

$$I = \dots, -1, 1, -1, 1, -1, \dots \quad (4.22)$$

Using the idling sequence, Equation 4.21 becomes

$$X \times m = \frac{1}{2} \left( X + \frac{1}{2} \left( I + \frac{1}{2} \left( X + \frac{1}{2} (X + I) \right) \right) \right) \quad (4.23)$$

Equation 4.23 demonstrates how the multiplication of a delta modulated sequence  $X$  by the coefficient  $m$  can be implemented as a sequence of half additions. By using this scheme, a delta multiplier can be constructed from delta adders (Figure 4.4).

The derived circuit can be used for multiplications by positive coefficients only, but a negative multiplication can be accomplished by first multiplying the signal by a positive coefficient then inverting the signal via a logical inverting gate. Delta modulated signals are composed of 1s and 0s which represent 1s and  $-1$ s. Since the two states of a delta modulated signal (1,0) are the logical inverses of each other

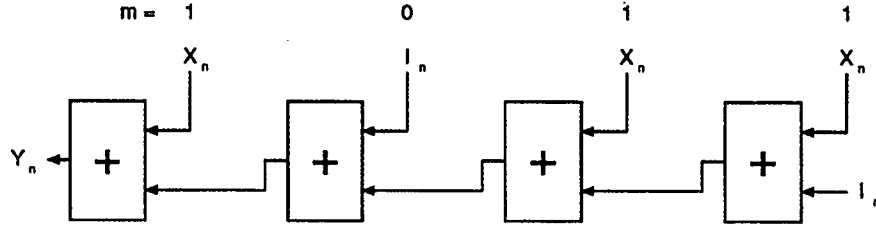


Figure 4.4: Multiplier for a  $0.1011_b$  coefficient constructed from delta adders .

and the two represented states  $(1, -1)$  are negatives of each other, logical inversion of the signal is equivalent to a negation.

An interesting consequence of the construction of the multiplier using successive half additions is the construction of a multiplier tree (Figure 4.5). In this structure, all possible multiplications (of a specific multiplier coefficient word length) are represented. Various multiplication coefficients may be chosen (or changed) by multiplexing the output of the tree. The error at the output of any branch of the tree of multipliers is always less than three times the error of the input signal [8].

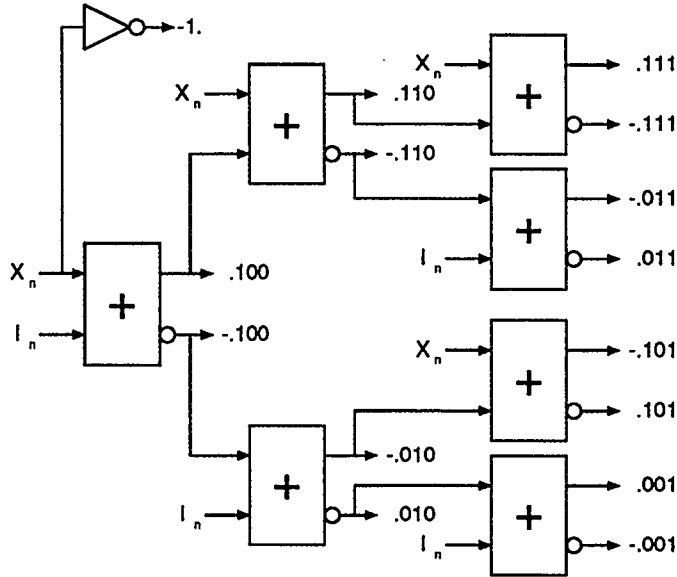


Figure 4.5: Delta adders used to create a tree of multiplication constants .

#### 4.4.3 Hardware implementation

The delta multiplier is based on a system of multiple delta adder components, so only the delta adder needs to be constructed. By converting Table 4.1 to a binary logic representation, where a 0 represents a  $-1$  and a 1 represents a 1, the logic circuit required to construct the system can be developed. The binary logic input/output relationship is shown in Table 4.2. If this input/output relationship is compared with the input/output relationship of a bit-serial full adder the similarity becomes apparent. The delta sum ( $S_n$ ) is the carry of a bit-serial adder, while the delta carry ( $C_n$ ) is the sum of a bit-serial adder.

$X_n$	$Y_n$	$C_{n-1}$	Delta sum, $S_n$	Delta carry, $C_n$	Bit serial sum	Bit serial carry
0	0	0	0	0	0	0
1	0	0	0	1	1	0
0	1	0	0	1	1	0
1	1	0	1	0	0	1
0	0	1	0	1	1	0
1	0	1	1	0	0	1
0	1	1	1	0	0	1
1	1	1	1	1	1	1

Table 4.2: Delta adder input/output relations rewritten to show correspondence with a bit-serial adder .

#### 4.4.4 Filtering Delta modulated Signals

By using the delta adder as a basic component, Kouvaras constructed several digital filters for delta modulated signals [8]. This method has one major drawback: true addition is not possible. Only the add-divide-by-two function is possible, so only filters that allow a multiplier to be encapsulated into the adder can be constructed. For instance, the equation

$$y = 0.25x_1 + 0.5x_2 \quad (4.24)$$

could be implemented as

$$y = 0.5(0.5x_1 + x_2) \quad (4.25)$$

because the divide-by-two nature of the delta adder allows construction of nested additions. On the other hand, an equation such as

$$y = 0.5x_1 + x_2 \quad (4.26)$$

can not be implemented since the delta adder does not allow true additions.

Finite impulse response (FIR) filters can be constructed easily from the delta adder and delta multiplier components. The divide-by-two property of the delta adder can be compensated for by modifying the multiplication coefficients in the filter. The maximum length of the cascade of two input adders needed to add all the outputs of an FIR filter is  $(N - 1)/2$ , where  $N$  is the order of the filter. Since each addition divides the input by two, the additions can be compensated for by multiplying the input to the adder sections by  $2^{(N-1)/2}$  in odd order filters. In even order filters one tap must be multiplied by  $2^{(N-1)/2}$ , while all others are multiplied by  $2^{N/2}$ . For some filters the additional multiplication can be assimilated by increasing the value of the multiplier coefficient. Unfortunately, the delta multiplier can be used only with coefficients less than unity, so the number of possible filter transfer functions is restricted.

Recursive filters can also be built with delta adders and delta multipliers, but the filter coefficients are restricted due to the add-divide-by-two nature of the delta adder. Any addition operation within a filter must include a divide-by-two operation. Thus the maximum coefficient within any addition loop within a filter is  $2^{-1}$ , and the more addition operations that occur within a filter loop the lower the maximum coefficient. The inherent divide-by-two nature of the delta adder can be compensated for by increasing the multiplier coefficient within a filter loop, but since all coefficients must be less than unity the number of possible transfer functions is restricted.

#### 4.4.5 Delta doubler

To alleviate the difficulties incurred by the divide-by-two nature of the delta adder, Kouvaras presented the delta doubler [9]. Essentially, it is a circuit to multiply a delta modulated signal by two. Using the doubler in series with an adder allows true addition at the cost of higher errors. The doubler increases the quantization noise to twice the input quantization noise at low frequencies, and three times the input quantization noise at high frequencies [9]. By using a more complex version of the delta doubler, the quantization noise can be reduced to a 50% increase over the input noise. By using the delta doubler, the class of realizable digital filters for delta modulated signals is extended to include recursive filters.

#### 4.4.6 Method Problems

Kouvaras's approach to digital filtering of delta modulated signals has two drawbacks. The first is the limited number range of filters that can be constructed using the delta adder and delta multiplier. This restriction is lifted by introducing the delta doubler, but the doubler increases the quantization error in the filter. A true delta adder constructed from a delta adder and a delta doubler will have errors in the range of four to six times the error in the input (or three times when using the more complex, modified delta doubler).

Kouvaras suggests that by decreasing the quantization interval the error can be reduced to a reasonable level. To decrease the quantization interval in a delta modulated system the sample frequency must be increased. The increase in the sample frequency is limited by the components used, and any increase in the sample frequency will effect the required accuracy of the digital filter.

#### 4.4.7 ROM Implementation

The construction of delta modulation filters with delta adders and delta multipliers requires only one component: the delta adder. The delta multiplier is constructed from multiple delta adders.

A hardware implementation of a delta adder is straightforward. The delta adder requires a small block of combinational logic or ROM for its computations. This block has three inputs and two outputs requiring  $2 \times 2^3 = 16$  bits for a look-up table implementation.

When constructing a delta modulated filter using delta adders, many small ROM components will be required rather than a few large ROM blocks. The delta adders, which compose all the arithmetic parts of the system, require many small ROM blocks which would be difficult to construct. Multipliers require many delta adder blocks so the entire system is constructed from many small ROM blocks.

ROM components are not well organized as small memory blocks. Memories are available as large blocks and an implementation using ROM components should use large blocks of memory rather than small blocks.

### 4.5 Filtering Ternary Delta Modulation

In delta modulation, only two states are used to represent the change of the input. This is convenient for a minimal, binary representation, but there is no necessity to encode a signal as only two values. A second possible difference coding technique, based on ternary signals has been presented [19, 20, 21]. Instead of encoding the error signal as a sequence of  $+\delta$  and  $-\delta$  signals, the error signal is encoded as  $+\delta$ , 0, or  $-\delta$ . Following a similar path to the development of the delta adder, a ternary adder has been constructed allowing implementations of a limited class of digital filters [19].



#### 4.5.1 Basic Ternary Delta Modulation

The ternary delta modulation system is very similar to the previously presented delta modulation method. The major difference between these two techniques is the choice of quantization scheme. In ternary delta modulation the quantizer is a three state device as opposed to the two state device used in delta modulation. The overall ternary delta modulation scheme is shown in Figure 4.6.

The quantization scheme quantizes the error signal to three states:  $+\delta$ , 0, or  $-\delta$ , which are represented by +1, 0, or -1, respectively. The addition of a zero representation in the quantizer reduces the overall error in the represented signal by 50% over binary delta modulation quantization. The higher signal to noise ratio is desirable, but the ternary signals can no longer be processed using the delta adder presented earlier.

Arithmetic operations on ternary signals require the introduction of a new set of components. In an approach similar to the delta adder method, a ternary adder has been created [19]. Multiple ternary adders are combined to form ternary multipliers, which are used in combination with ternary adders to create a limited class of digital filters. The digital filters require ternary delta modulated inputs and produce ternary

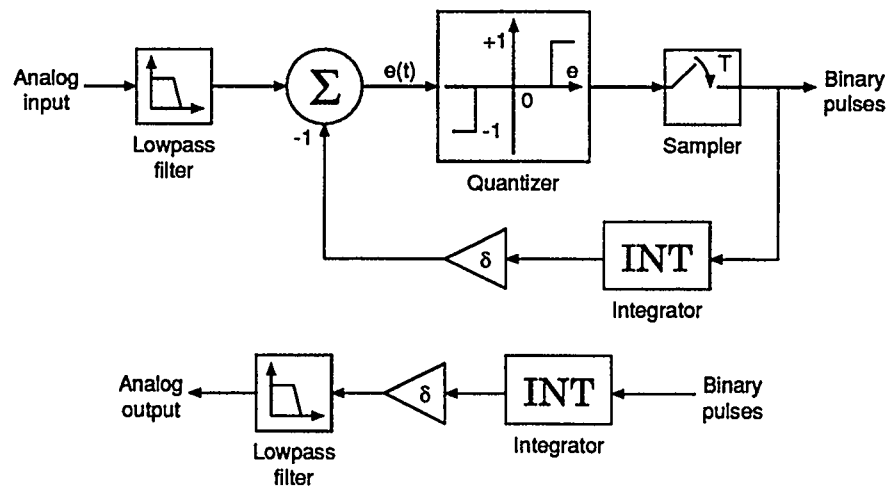


Figure 4.6: The basic ternary delta modulation system .

delta modulated outputs.

#### 4.5.2 Ternary adder

Unlike in delta modulation, the sum of two ternary delta modulation signals ( $X_n$  and  $Y_n$ ) will not always add to an unrepresentable number. Table 4.3 shows that the sum,

$$S_n = X_n + Y_n \quad , \quad (4.27)$$

possesses only two unrepresentable values:  $-2$  and  $2$ .

If an output carry term is added to allow accurate representations of the sum, a carry must be added as an input term of the system. Again this would leave two unrepresentable states:  $-3$  and  $3$ . To eliminate the difficulty in these representations, Zrilić[19] proposed an adding component called the ternary adder.

The ternary adder is not a true addition operation, but an addition with a divide by 3 in series. The output of the adder ( $S_n$  and  $C_n$ ) as a function of the input  $Z_n$  is found in Table 4.4, where  $Z_n$  represents the true sum plus carry,

$$Z_n = X_n + Y_n + C_{n-1} \quad . \quad (4.28)$$

The ternary adder allows all possible inputs to be represented, although some output states must be approximated over several iterations.

$X_n$	$Y_n$	$S_n$
-1	-1	-2
-1	0	-1
-1	+1	0
0	-1	-1
0	0	0
0	+1	+1
+1	-1	0
+1	0	+1
+1	+1	+2

Table 4.3: Input/output relations for the addition of ternary signals .

$Z_n$	3	2	1	0	-1	-2	-3
$S_n$	+1	+1	0	0	0	-1	-1
$C_n$	0	-1	+1	0	-1	+1	0

Table 4.4:  $Z_n$  as a function of sum,  $S_n$ , and carry,  $C_n$ .

The values for the carry,  $C_n$ , and the one-third sum,  $S_n$ , are calculated from the equations:

$$C_n = \frac{(Z_n^3 - 9Z_n)(3Z_n^2 - 8)}{40} \quad (4.29)$$

$$S_n = \frac{Z_n - C_n}{3} \quad (4.30)$$

$$S_n = \frac{1}{3} [X_n + Y_n - (C_n - C_{n-1})] \quad (4.31)$$

$$C_{n-1} = +1, 0 \text{ or } -1 \quad (4.32)$$

where  $n = \dots, -3, -2, -1, 0, 1, 2, 3, \dots$ . Using this approach, the error in the ternary adder was shown to be less than  $\frac{4}{3}\delta$  where  $\delta$  is the quantization interval of the ternary encoder [19].

### 4.5.3 Ternary Multiplier

The ternary multiplier is constructed in much the same manner as the delta multiplier explained earlier. Since the ternary adder is an addition and divide-by-three operation, successive ternary additions of the signal with zero or itself will perform multiplications for coefficients between 0 and  $\frac{1}{2}$ . For example, to multiply an arbitrary ternary signal  $X$  by a multiplication coefficient  $m = 0.12346$ ,

$$X \times m = X \times 0.12346 \quad (4.33)$$

$$= X \left( \frac{1}{3^2} + \frac{1}{3^4} \right) \quad (4.34)$$

$$= X \frac{1}{3^2} + X \frac{1}{3^4} \quad (4.35)$$

$$= \frac{1}{3} \left( 0 + X \frac{1}{3} + X \frac{1}{3^3} \right) \quad (4.36)$$

$$= \frac{1}{3} \left( 0 + \frac{1}{3} \left( X + X \frac{1}{3^2} \right) \right) \quad (4.37)$$

$$= \frac{1}{3} \left( 0 + \frac{1}{3} \left( X + \frac{1}{3} \left( 0 + X \frac{1}{3} \right) \right) \right) \quad (4.38)$$

$$= \frac{1}{3} \left( 0 + \frac{1}{3} \left( X + \frac{1}{3} \left( 0 + \frac{1}{3} (0 + X) \right) \right) \right) \quad (4.39)$$

By using Equation 4.39, and the ternary adder previously explained, a ternary multiplier with a coefficient of  $m$  can be constructed (Figure 4.7). Negative coefficients are implemented by using positive multipliers followed by ternary inversion, where the  $-1$  state becomes 1, the 1 state becomes  $-1$  and the zero state is unchanged by inversion. The error in the ternary multiplier is less than  $2\delta$ , where  $\delta$  is the quantization interval [19].

#### 4.5.4 Ternary Delta modulation Filters

By using the ternary delta adder and the ternary multiplier (constructed from adders), a limited class of filters is realizable [19]. Realizable filter structures must allow all addition operations to incorporate a multiplication by  $\frac{1}{3}$  and the multiplication coefficients are restricted to values between  $-0.5$  and  $0.5$ . Since the multiplier coefficients in ternary filters are restricted to a narrower range of values, the number of possible realizable ternary delta modulation filter structures is less than the number of realizable delta modulation filter structures. This disadvantage is compensated for by lower errors in the ternary system since ternary delta modulation has a 50% lower quantization than standard delta modulation.

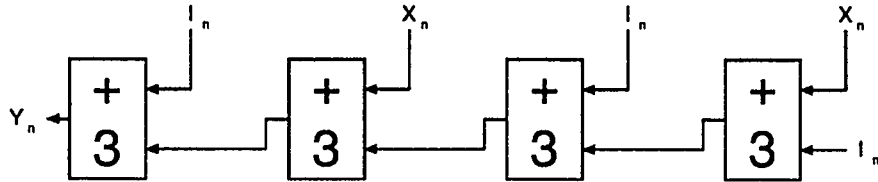


Figure 4.7: A ternary multiplier with a coefficient of 0.12346.

#### 4.5.5 Ternary Delta Tripler

The restrictions on the number of realizable filter structures is lifted by introducing a ternary delta tripler, which triples the input signal [20]. By using a ternary tripler in series with a ternary adder, true addition becomes possible, at the cost of increased errors. Multiplication by coefficients above 0.5 become possible as well by a combination of ternary multipliers, ternary adders, and ternary delta triplers.

The tripler introduces a larger error in the addition operation when it is cascaded with a ternary adder [20]. The increased error can be reduced by increasing the sampling frequency and reducing the step quantization interval.

#### 4.5.6 Hardware

The ternary delta adder is constructed using three-state logic on custom integrated circuit chips [19]. The approach allows maximum use of the available routing, since only one multilevel data line is required for data transmission. Such an approach is reasonable for large production runs, but for smaller production runs it is prohibitive. The use of multivalued logic (three states in this case) prevents the use of standard logic design tools. A new set of logic blocks, encompassing the three state nature of the system, must be designed from analog components, which increases the cost of a filter construction using ternary logic.

#### 4.5.7 Method Problems

The greatest drawback of the ternary delta modulation system is the fabrication of custom IC chips for the ternary logic. An alternative implementation is to use a two bit parallel word in the designed system. This would allow the representation of three states, yet allow implementation using standard logic devices. In this scheme, one bit in four (25%) of the states in the system are unused, although this may be tolerated since a two bit wordlength is considerably easier to route on a chip than a

12 or 15 bit parallel word.

In either method of fabrication, the range of filters that can be implemented is limited. The 0.5 maximum coefficient on a multiplier is a very harsh restriction, especially when the multiplier is expected to compensate for the divide by 3 operation of the adder. The ternary delta tripler alleviates the problem, but increases the error. Direct conversion to ROM based systems is awkward since the method is based on the assumption of many small units, rather than on large state-machine blocks more suited to the ROM available today.

## 4.6 Filtering of Sigma-Delta Modulated Signals

An alternative structure to delta modulation is sigma-delta modulation. Sigma-delta modulation is a difference encoding method similar to delta modulation, but the integrator (decoder) is placed prior to the difference modulator. This allows sigma-delta modulation to be more robust against channel or coding errors than is delta modulation [18].

### 4.6.1 The Basics of Sigma-Delta Modulation

The delta-sigma modulator (Figure 4.8) consists of a delta modulator preceded by an integrator. The delta modulator encodes the integrated input signal. The quantizer within the delta modulator differs from previous quantizers in that it quantizes to the full dynamic range instead of a small quantization step.

Let  $A$  represent the maximum absolute dynamic range, so the full dynamic range is between  $-A$  and  $A$ . To avoid distortions in encoding the input signal, designated as  $x(t)$ , the input must always lie within the dynamic range,

$$-A \geq x(t) \geq A \quad (4.40)$$

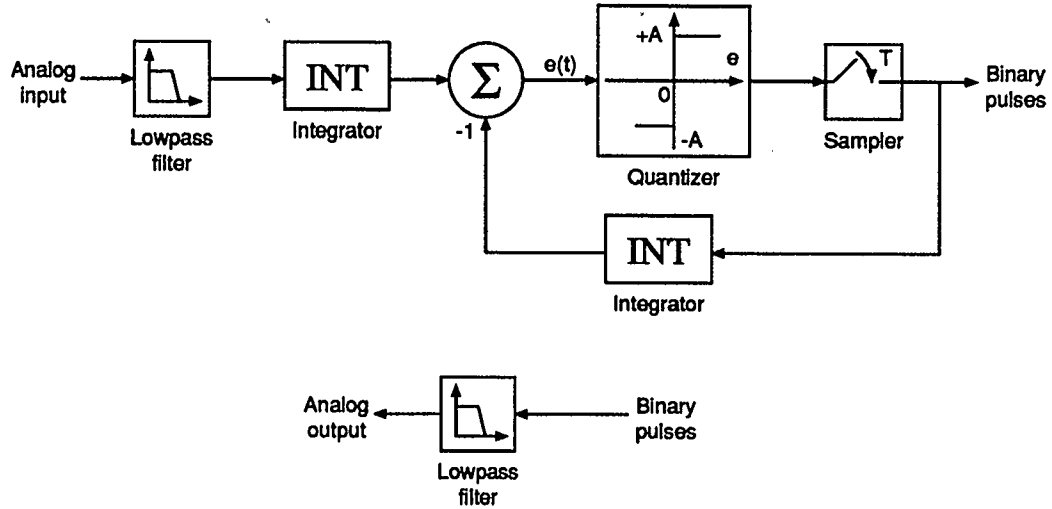


Figure 4.8: Basic sigma-delta modulation system .

In a sigma-delta system the quantizer will quantize the input to either  $A$  or  $-A$  which can be encoded using one bit.

To convert the delta-sigma signal to a multilevel signal a decoder is required, which is a sharp cut-off lowpass filter. The lowpass filter averages the delta-sigma signal (made up of  $A$  and  $-A$  impulses) and produces a multilevel output signal. This multilevel signal can then be converted to analog if required.

The overall transfer function of the system remains unity because only the order of the components has been changed from the delta modulation system. In a linear system, the transfer function is independent of the order of operations, assuming the final averaging filter has a passband gain of unity.

#### 4.6.2 Filtering Sigma-Delta Modulated Signals

Filtering sigma-delta modulation encoded signals has been accomplished in a fully digital FIR implementation [18]. The FIR filter (Figure 4.9) is a direct implementation of the convolution of the desired impulse response of the filter and the input. The input signal  $x(t)$  is sampled at  $R$  times the Nyquist rate and the oversampled signal is encoded as a sigma-delta sequence  $v_n$ . The impulse response of the  $K^{th}$

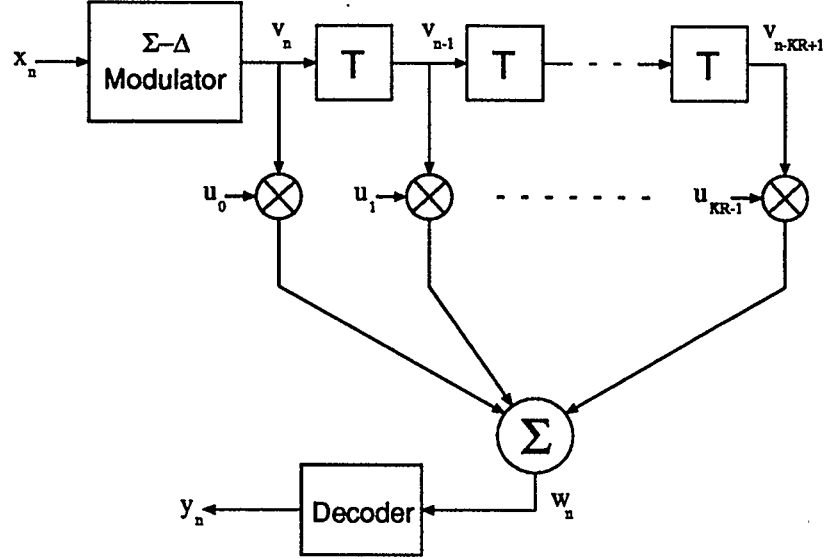


Figure 4.9: A sigma-delta modulation FIR filter .

order filter,

$$a_n = a_0, a_1, \dots, a_{K-2}, a_{K-1} \quad (4.41)$$

is up-sampled by  $R$  and encoded in a sigma-delta sequence

$$v_n = v_0, v_1, \dots, v_{RK-1} \quad (4.42)$$

The sequence  $v_n$  is undefined before the filter begins operation, and this leads to unknown initial conditions within the filter. It is assumed that the filter begins in a zero state, but any initial state may be chosen. Non-zero initial conditions will lead to a transitory error during the first  $RK - 1$  values, but the initial condition would not effect long term behaviour [18].

The convolution of the input signal  $u_n$  and the impulse response of the filter  $v_n$  is

$$w_n = \sum_{i=0}^{RK-1} u_i v_{n-i} \quad (4.43)$$

and this multibit signal is the output of the filter. After low-pass filtering the resulting analog signal represents the filtered input signal.

Since both  $u_i$  and  $v_i$  are binary signals (composed of 1s and 0s) so that their



multiplication can be carried out by a simple logic circuit. This leads to considerable reduction in the complexity of the circuit design. Although the multipliers required are considerably simplified the adder has increased in complexity. The adder is implemented as a full parallel addition of  $KR - 1$  one bit words.

#### 4.6.3 Problems in the Sigma-Delta Approach

The filter requires very high order to function with reasonable accuracy. Since the implementation is restricted to FIR filters, the filter order is higher than would be required for an recursive implementation. This high order impulse response is upsampled by  $R$  times to produce  $KR - 1$  coefficients which necessitate a  $KR - 1$  input adder to produce  $w_n$ .

The adder can be implemented as a multiplexed adder, or a counter (since all inputs are binary), but for either case the adder must operate at higher speeds than the sigma-delta filter. Since the filter is already oversampled by  $R$ , the adder must operate at  $(KR - 1)R$  times the Nyquist rate to be implemented as a counter with one input bit. The high operating speed of the adder reduces the maximum sample rate of the sigma-delta filter to a value below the maximum sample rate of a similar delta modulation filter.

#### 4.6.4 ROM Implementation

The main difficulty in ROM based implementation is the large parallel adder in the filter. All states must be added together at the output and this necessitates a very large parallel adder. Reducing the ROM to a reasonable size by multiplexing the adder will reduce the maximum Nyquist frequency of the input.

The difficulty in implementation is the assumption of inexpensive large adders. In most technologies, the adder is an inexpensive (both in chip area and delay time) component while the multiplier is expensive. In a ROM based implementation, which

focuses on state machine implementations, the adder has double the number of inputs required for a multiplier leading to a memory requirement which is the square of the memory required for a ROM multiplier. In the delta-sigma filter, the adder at the output of the filter has  $KR - 1$  inputs and implementing this in ROM is difficult. Since  $R$  is typically in the range of 64 to 1024 [18] a full parallel implementation for 64 input bits is impractical with present technology. A adder with 64 input lines leads to a ROM requiring  $2^{64} = 1.844 \times 10^{19}$  bits of storage.

## 4.7 Conclusions

In the previous sections, three approaches to filtering reduced wordlength signals have been examined. The examination demonstrates that both approaches are inappropriate for ROM based implementations. In the following chapter, the author presents a filtering technique for reduced wordlength signals called difference signals.

## Chapter 5

### Filtering Difference Signals

In Chapter 1, the memory requirements for a block ROM implementation of a 5<sup>th</sup> order, 12-bit wordlength filter structure was determined to be 0.34 tera-terabytes. Chapter 2 explains the previous use of ROM components in digital filters. In chapter 3, the memory requirements for a block ROM filter was reduced by encoding only the reachable states of the filter. Although the state space of the filter can be reduced to a fraction of its previous size, the memory requirements of the approach are still too large to allow construction. In the previous chapter a second approach, where the input, output and states are reduced in wordlength using delta modulation, ternary delta modulation, or sigma delta modulation, was examined. The chapter provides the background for the author's method which is explained in this chapter.

#### 5.1 Introduction

Chapters 2 and 3 outline ROM filter implementations based on single ROM blocks, but when the implementation is partitioned into individual look-up tables for each adder and multiplier, a 5<sup>th</sup> order lossless discrete integrator (LDI) structure (chosen for its low sensitivity to coefficient wordlengths [3]) would require much less memory.

If one look-up table is implemented for each multiplier and adder in the 5<sup>th</sup> order LDI structure with a 12-bit wordlength, the entire structure would require  $2.77 \times 10^8$  bytes or 277 megabytes (Table 5.1). Although the memory required for such an implementation is much smaller than a block ROM state machine implementation, it is still very large and can be further reduced by combining multiplication blocks

Components	Memory Required	
	in bits	in bytes
11 adders	$2.21 \times 10^9$	$2.77 \times 10^8$
11 multipliers (including $\times -1$ s)	$5.41 \times 10^5$	$6.76 \times 10^4$
total memory	$2.22 \times 10^9$	$2.77 \times 10^8$

Table 5.1: Memory requirements for a 5<sup>th</sup> order LDI using look-up tables.

with the adder block.

When an adder block is realized as a ROM look-up table, it is realized as an arbitrary dual input function:  $y = f(x_1, x_2)$ . If all inputs and outputs have the same dynamic range ( $-A$  to  $A$ ) then the function  $f(x_1, x_2)$  is accurate provided  $-A \geq f(x_1, x_2) \geq A$ . By using an arbitrary two input function, all the multipliers can be combined with the adder components with no loss in accuracy. The combined adder/multiplier function is a two input block representing

$$y = f(x_1, x_2) = m_1x_1 + m_2x_2 \quad , \quad (5.1)$$

where  $m_i$  are the arbitrary multiplier coefficients and  $x_i$  are the input signals. By using such a set of programmed two input blocks, the entire filter can be constructed. The partitioning of the two input blocks for the 5<sup>th</sup> order LDI structure is shown in Figure 5.1 where dashed lines designate each two input block (labeled A to L) .

By Using 11 dual input blocks, the memory required for the 5<sup>th</sup> order LDI filter is reduced by the amount of ROM required for the multipliers ( $5.41 \times 10^5$  bits).

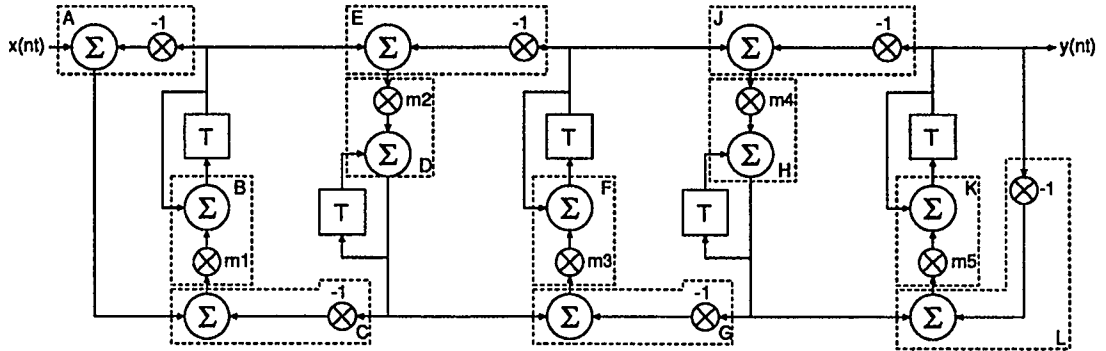


Figure 5.1: A 5<sup>th</sup> order LDI structure composed of two input blocks. .

The reduction is small compared to the overall memory requirements because the multiplier components eliminated require very little space compared to the adder components, which comprise greater than 99% of the memory requirements. The small space requirements for multiplier components compared to adder components illustrates a fundamental difference between a conventional implementation and full ROM based implementations.

In conventional designs, the majority of resources are consumed by multipliers. Multipliers require more chip area, more gates and are often the slowest components in a filter design. Most techniques that use ROM based look-up tables to increase the speed of the multiplier assume inexpensive adders (Chapter 2). When all components are implemented with ROM look-up tables the assumption of inexpensive adders is false. In full ROM look-up table implementations, the majority of the hardware is consumed by adder components rather than multiplier components because adder components require twice the number of inputs. The doubling of the number of inputs causes an adder to require the square of the memory required for a multiplier block of the same wordlength. This shifts the focus of design work from multiplier components to adder components.

The shift in focus from multipliers to adders requires a novel design strategy. In Chapter 3, the number of possible super-states that the machine may lie in is reduced. This reduced the number of bits required to represent the super-state, and thus reduced the memory requirements of a ROM based implementation of the filter.

An alternative approach is the reduction of the length of the word used to represent each of the individual states and input. This approach has the advantage of reducing the memory requirements faster than the super-state reduction method since more bits can be removed from the representation of the state.

In this chapter, the basic premise for state wordlength reduction by oversampling

and encoding the change in the input signal rather than the amplitude of the signal is explained. Since arithmetic operations on the difference signal can not be performed by existing components, novel difference multiplier and difference adder components are defined. Finally a simulation of a 5<sup>th</sup> order LDI filter is presented.

## 5.2 Difference Signals

Highly oversampled, bandlimited signals have a high correlation between the present sample value and the previous sample value. The difference between these two samples decreases as the sample rate increases. The maximum change of an input (with a maximum amplitude of the full dynamic range  $A$  and a frequency  $f$ ) between two samples (separated by  $T_s$ ) was found in Equation 4.3 and is

$$\Delta_{max} = 2\pi A f T_s = \frac{2\pi A f_{max}}{f_s} \quad .$$

If the frequency of the input is set to the highest unaliased input frequency which is half the Nyquist frequency,

$$f_s = 2Rf_{max} \quad , \quad (5.2)$$

where  $R$  is the oversampling rate, Equation 4.3 can be rewritten as

$$\Delta_{max} = \frac{2A\pi f_{max}}{2Rf_{max}} = \frac{A\pi}{R} \quad . \quad (5.3)$$

The maximum change  $\Delta_{max}$  of the input is inversely related to the ratio of oversampling  $R$ . Every doubling of the oversampling ratio  $R$  will halve the maximum change in the signal  $\Delta_{max}$ . By reducing the maximum change in the signal the required dynamic range of the filter is reduced which reduces the the total number of levels required to represent a quantized signal. When the number of levels is reduced to half the original number of levels one bit can be removed from the binary word used to represent the signal. In general the number of bits removed,  $\beta$ , is

$$\beta = \left\lfloor \log_2 (R) \right\rfloor \quad . \quad (5.4)$$

For a ROM access speed of 25 MHz (40 ns), telephone data ( $f_{max} = 3000$  Hz) potentially can be reduced by 12 bits (Figure 5.2). Larger reductions are possible by using faster ROM components.

A similar study of bit reduction for audio range data ( $f_{max} = 22$  kHz) leads to a maximum potential reduction of 8 bits (Figure 5.3), but larger bit reductions are possible by using faster ROM components.

Both Figure 5.2 and Figure 5.3 demonstrate that by oversampling a bandlimited input signal it is possible to significantly reduce the wordlength of the data. For sufficiently small initial wordlengths, or sufficiently large oversampling rates, the input can be reduced to two bits representing three states:  $+\delta$ ,  $0$ ,  $-\delta$ , where  $\delta$  is the step quantization interval.

The similarity to delta modulation and especially to ternary delta modulation is clear, but unlike either approach the maximum number of levels to which the input will be quantized is not defined. The number of quantization levels depends on the step quantization interval  $\delta$ , the oversampling ratio  $R$  and the maximum change of the input signal  $\Delta_{max}$ .

### 5.3 Signal Coding, Reconstruction and Filtering

In the previous section, the possibility of reducing the wordlength necessary to represent the input signal by encoding it as a difference signal was demonstrated. In this section, the methods by which the difference signal is created and the output signal is reconstructed are discussed. The error introduced by the encoding and decoding of the signals is determined and the potential for filtering difference signals before reconstruction is examined.

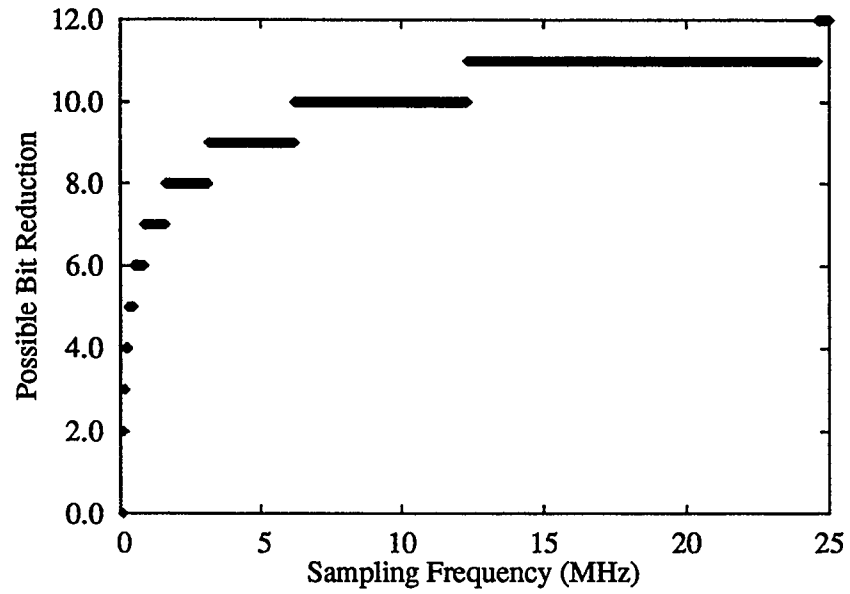


Figure 5.2: Bit reduction by oversampling for telephone (3 kHz) bandwidth data.

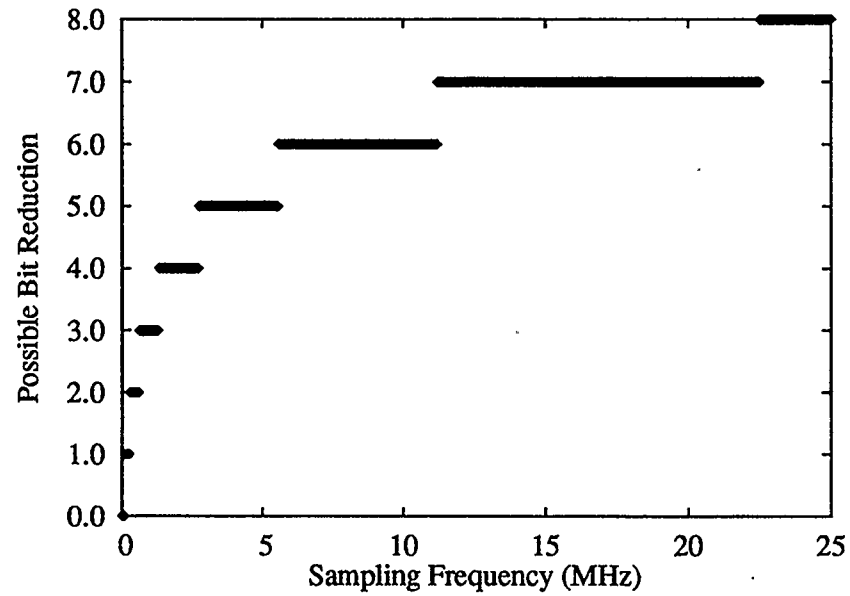


Figure 5.3: Bit reduction by oversampling for audio (22 kHz) bandwidth data.



### 5.3.1 Encoding of Difference Signals

The coding of the input signal into a difference signal is accomplished by a difference encoder (Dencoder) or difference modulator (Figure 5.4). There are many similarities between this modulator and modulators used for delta modulation and ternary delta modulation. In all three cases, the overall structure is similar, but the number of quantization levels differs. In delta modulation, two quantization levels are required, while in ternary delta modulation three quantization levels are needed. The Dencoder has an arbitrary odd number of quantization levels  $L$  which depends on the maximum change in the input and the quantization step interval such that

$$L = \left\lceil 2 \frac{\Delta_{max}}{\delta} \right\rceil + 1 \quad , \quad (5.5)$$

where the ceiling operator  $\lceil \cdot \rceil$  on  $x$  is defined as

$$\lceil x \rceil = \min\{i \in \mathcal{I} : i \geq x\} \quad . \quad (5.6)$$

Since the maximum change in the input,  $\Delta_{max}$ , is dependent on the oversampling ratio,  $R$  and on the maximum amplitude (or dynamic range) of the input  $A$  (Equation 5.3), Equation 5.5 becomes

$$L = \left\lceil 2 \frac{A\pi}{R\delta} \right\rceil + 1 \quad . \quad (5.7)$$

The step quantization interval  $\delta$  reflects the accuracy of the quantization of the input signal. If the maximum dynamic range of the input signal  $A$  is assumed to be constant, then increasing  $\delta$  will decrease the number of quantization levels required, but the larger  $\delta$  becomes, the larger is the error in the quantization of the input. Since the error in the quantization of the input is related to  $\delta$ ,  $\delta$  is effectively fixed for any desired quantization error. Only the oversampling ratio  $R$  remains to adjust the number of levels in the quantizer.

By increasing  $R$  the number of quantization levels  $L$  can be reduced without influencing the accuracy of the quantization of the input. Of course  $R$  can only

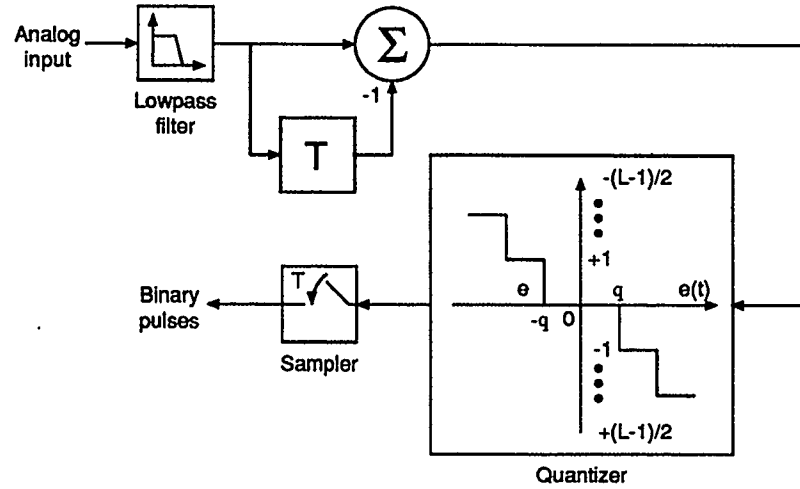


Figure 5.4: Difference encoder (Dencoder) block diagram.

be increased to the limits of the implementing technology, so in the 25 MHz ROM example used earlier,

$$R_{audio} \leq (25 \text{ MHz}) / (2 \times 22 \text{ kHz}) = 568.2 \quad (5.8)$$

for audio bandwidth data, while for telephone bandwidth data

$$R_{telephone} \leq (25 \text{ MHz}) / (2 \times 3 \text{ kHz}) = 4166.7 \quad (5.9)$$

Using the Dencoder, the oversampled input signal is encoded as an  $L$  level signal. These levels are encoded as binary words with a  $b$  bit word length, where

$$b = \left\lceil \log_2(L) \right\rceil \quad (5.10)$$

It is this narrow binary word which is transmitted to the difference decoder (or demodulator) which reconstructs the output signal.

### 5.3.2 Decoding of Difference Signals

The difference signal decoder (Ddecoder) used to reconstruct the output signal from the difference signal created by the Dencoder is identical to the decoder used in both delta modulation and ternary delta modulation, that is, it is an integrator. The input

sample is summed with the previous output sample and the binary representation of the signal is then converted to an analog signal via a digital-to-analog (D/A) converter and a lowpass filter. The D/A converter is effectively a multiplication by  $\delta$  if the binary multilevel signal created by the Ddecoder is considered a signed integer. The lowpass filter smooths the quantized output to produce the final analog signal.

In the discrete time domain, the integrator effectively places a pole at  $z = 1$  canceling the zero at  $z = 1$  produced by the encoder. The cancellation of the zero by the pole implies a transfer function of unity for the overall system.

In the discrete time domain, the operation of the Dencoder can be modeled as

$$\hat{x}(n) = x(n) - x(n-1) \quad , \quad (5.11)$$

where  $\hat{x}$  represents the quantized discrete time difference signal of the quantized input  $x$ . The difference signal  $\hat{x}$  is quantized to  $L$  levels, but in this case  $L$  is assumed to be an infinite number of levels, so the non-linear effect of the quantizer is effectively ignored to allow a linear analysis.

The Ddecoder can be modeled as

$$y(n) = \hat{x}(n) + y(n-1) \quad (5.12)$$

where  $y(n)$  represent the quantized discrete time output signal at discrete time  $n$ .

Using the  $\mathcal{Z}$ -transform ( $\mathcal{Z}[\cdot]$ ) of the discrete time domain equations, the corresponding transfer functions are

$$H_{\text{Dencoder}} = \frac{\mathcal{Z}[\hat{x}]}{\mathcal{Z}[x]} = 1 - z^{-1} \quad (5.13)$$

$$H_{\text{Ddecoder}} = \frac{\mathcal{Z}[y]}{\mathcal{Z}[\hat{x}]} = \frac{1}{1 - z^{-1}} \quad (5.14)$$

$$H_{\text{Overall}} = H_{\text{Dencoder}} H_{\text{Ddecoder}} = (1 - z^{-1}) \frac{1}{1 - z^{-1}} = 1 \quad . \quad (5.15)$$

This analysis is valid for discrete time signals where  $L \rightarrow \infty$  which implies  $\delta \rightarrow 0$ . If the quantization step  $\delta$  is not infinitely small the error at the output of the system

will not be infinitely small. The error is related to  $\delta$  and like the delta modulation system explained in the previous chapter, an arbitrarily small error is possible by a suitable choice of  $\delta$ .

### 5.3.3 Error in the Encoding/Decoding System

When the step quantization interval,  $\delta$ , is greater than 0 the quantizer has an associated error. This error can be modeled as noise. If the quantizer in Figure 5.4 is replaced by an adder with a noise input  $\eta$ , then using superposition and assuming zero initial conditions, the output of the overall system is

$$y = x + \eta \quad (5.16)$$

This implies that the noise at the output of the system is dependent on the noise introduced by the quantizer.

The quantization scheme  $\mathcal{Q}[\cdot]$  is the method by which the input is quantized into the digital levels used internally by the system and it is this quantization scheme that determines the range of  $\eta$ . The difference between any two adjacent levels in the quantization scheme is the step quantization interval  $\delta$ . The error quantization interval  $q$  is the maximum difference between a quantized signal and its continuous counterpart. The error interval  $q$  depends on  $\delta$  and the rounding scheme of  $\mathcal{Q}[\cdot]$ .

If  $\mathcal{Q}[\cdot]$  uses truncation rounding, then  $q = \delta$ . On the other hand, if  $\mathcal{Q}[\cdot]$  uses magnitude rounding,  $q = \frac{1}{2}\delta$ . The noise in the quantizer  $\eta$  is restricted by  $q$  such that  $|\eta| \leq q$ , so the output will be within one error quantization interval of the ideal continuous output.

After the input signal is quantized it is converted to a difference signal (composed of  $L$  levels) by the Decoder. The error in the signal at the output is twice the error of the input signal to the difference encoder. To demonstrate that this is so, assume that an analog input signal is sampled to create a discrete time signal  $x(n)$ . The

discrete time signal is quantized and can be represented as the discrete time signal with discrete time noise,

$$x(n) + \eta(n) \quad . \quad (5.17)$$

The signal is then converted to a difference signal using the Dencoder, giving

$$\hat{x}'(n) = (x(n) + \eta(n)) - (x(n-1) + \eta(n-1)) \quad (5.18)$$

$$= (x(n) - x(n-1)) + (\eta(n) - \eta(n-1)) \quad (5.19)$$

$$= \hat{x}(n) + \hat{\eta}(n) \quad . \quad (5.20)$$

The noisy difference signal is the sum of the difference signal of the input plus the difference signal of the noise. For magnitude rounding the noise  $\eta(n)$  is restricted to

$$-\frac{\delta}{2} \geq \eta(n) \leq \frac{\delta}{2} \quad , \quad (5.21)$$

so the difference signal of the noise is restricted to

$$-\delta \geq \hat{\eta}(n) \leq \delta \quad . \quad (5.22)$$

The above analysis assumes zero initial conditions within the Dencoder. If the Dencoder's previous input (the stored input) does not match the previous value of the input, the Dencoder's output will be erroneous. The error will be corrected on the next sample, but the error will cause a DC offset at the output of the Ddecoder.

The signal at the output of the Dencoder is applied to the input of the Ddecoder, and the output of the Ddecoder is

$$y(n) = \mathcal{Z}^{-1} \left[ \frac{1}{1 - z^{-1}} \mathcal{Z} [\hat{x}'(n)] \right] \quad (5.23)$$

$$= \mathcal{Z}^{-1} \left[ \frac{1}{1 - z^{-1}} \mathcal{Z} [\hat{x}(n) + \hat{\eta}(n)] \right] \quad (5.24)$$

$$= \mathcal{Z}^{-1} \left[ \frac{1}{1 - z^{-1}} \left( \mathcal{Z} [x(n)] (1 - z^{-1}) + \mathcal{Z} [\eta(n)] (1 - z^{-1}) \right) \right] \quad (5.25)$$

$$= \mathcal{Z}^{-1} [\mathcal{Z} [x(n)] + \mathcal{Z} [\eta(n)]] \quad (5.26)$$

$$= x(n) + \eta(n) \quad . \quad (5.27)$$

The output of the Ddecoder is the sum of the noise and the input signal. The noise here is restricted to

$$-\frac{\delta}{2} \geq \eta(n) \leq \frac{\delta}{2} \quad (5.28)$$

for magnitude rounding.

The noise analysis for the Ddecoder assumes that the Ddecoder begins operation in a zero state. If the previous output (which is internally stored) is not correct a DC offset error occurs. The error persists because the Ddecoder is a discrete time integrator and has no method of correcting such an error.

#### 5.3.4 Filtering Difference Signals

By using the Dencoder, a bandlimited input signal can be encoded as a difference signal, and this difference signal need not be converted to analog to be filtered. Consider an arbitrary bandlimited discrete time input  $x(n)$  and a filter transfer function  $H(z)$ .

$$Y(z) = H(z) \cdot X(z) \quad , \quad (5.29)$$

where  $Y(z) = \mathcal{Z}[y(n)]$  and  $X(z) = \mathcal{Z}[x(n)]$ . If the input  $X(z)$  is encoded, and then decoded, it will still be equal to  $X(z)$ ,

$$X(z) = H_{Ddecoder}(z) \cdot H_{Dencoder}(z) \cdot X(z) \quad (5.30)$$

$$= \frac{1}{(1 - z^{-1})} \cdot (1 - z^{-1}) \cdot X(z) \quad (5.31)$$

$$= X(z) \quad . \quad (5.32)$$

Thus,

$$\begin{aligned} Y(z) &= H(z) \cdot X(z) \\ &= H(z) \cdot H_{Ddecoder}(z) \cdot H_{Dencoder}(z) \cdot X(z) \quad . \end{aligned} \quad (5.33)$$

In the frequency domain, multiplication is commutative, so Equation 5.33 can be rearranged to

$$Y(z) = H_{Ddecoder}(z) \cdot H(z) \cdot H_{Dencoder}(z) \cdot X(z) \quad (5.34)$$

Equation 5.34 demonstrates that it is possible to encode the signal, filter the signal while it is encoded and finally construct the output signal using the decoder. The output signal will be the filtered input signal, except for quantization error.

## 5.4 Components

The construction of difference filters requires novel multiplier and adder components. Components developed for amplitude signals are inappropriate for difference signal operations because of the limited wordlength imposed by  $L$  levels. Both addition and multiplication components have been developed for processing difference signals and in this section the structure and error encountered in each of these components are examined.

### 5.4.1 Adding Difference Signals

The addition of two difference signals poses several problems. Earlier efforts solved the summing problem by avoiding true addition [8, 19, 21]. A multiplier from elsewhere in the structure is assimilated into the adder to prevent additions summing to values beyond the representable values. This approach allows a limited class of digital filters to be constructed.

The difference adder or Dadder (Figure 5.5) presented here allows true addition of difference signals. Two inputs and a carry produce an output sum and a carry. The output carry is cycled back (through a unit delay) into the Dadder. The wordlength of the inputs ( $b_1$  and  $b_2$ ) and the wordlength of the sum ( $b_{sum}$ ) tend to be identical

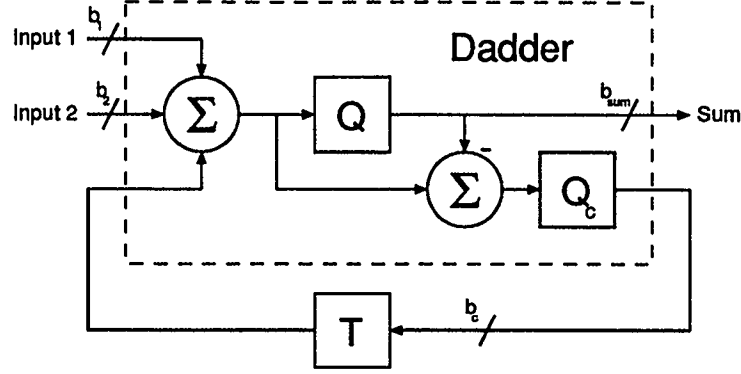


Figure 5.5: Block diagram of the difference adder (Dadder).

while the wordlength of the carry ( $b_c$ ) depends on the quantization scheme and the number of addition operations to be cascaded.

The Dadder is composed of a delay element (labeled T) and a single ROM block (the dashed box in Figure 5.5) programmed to perform

$$\hat{y}(n+1) = Q[C(n) + \hat{x}_1(n) + \hat{x}_2(n)] \quad (5.35)$$

$$C(n+1) = Q_c[C(n) + \hat{x}_1(n) + \hat{x}_2(n) - \hat{y}(n+1)] \quad , \quad (5.36)$$

where

$Q[\cdot]$  is the quantization/saturation scheme of the inputs (and sum)

$Q_c[\cdot]$  is the quantization/saturation scheme of the carry

$\hat{x}_1(n)$  is the input difference signal 1

$\hat{x}_2(n)$  is the input difference signal 2

$C(n)$  is the carry

$\hat{y}(n)$  is the output difference signal representing the sum.

The quantization scheme of the output signal and the carry are represented separately to allow separate quantization schemes.

The carry  $C$  is used to prevent the loss of the error in the computation of the sum. At any time, the output is a quantized approximation to the sum, while the carry is the error in the approximation. The carry is cycled back into the Dadder to



allow a second approximation on the next cycle. By constantly correcting for errors in the sum of the signals, close approximations to the sum are created.

The carry is the error in the output when compared to the sum of the two input signals. If the carry should overflow, or be saturated, the error will be passed on to the output of the Dadder and will be unrecoverable. This error will appear at the output of a Ddecoder as a DC offset. To prevent carry overflow, or saturation, the maximum value of the carry must be calculated.

The maximum amplitude change representable is the maximum change  $\Delta_{max}$  of the input signal  $x(t)$ . Any two inputs  $x_1(t)$  and  $x_2(t)$  must, at most, have amplitude changes summing to  $\Delta_{max}$ ,

$$\Delta_{max} \geq \kappa_1(t) + \kappa_2(t) \quad (5.37)$$

where  $\kappa_1$  and  $\kappa_2$  represent the time varying amplitude changes of inputs  $x_1(t)$  and  $x_2(t)$ , respectively. If the sum of the amplitude changes are greater than  $\Delta_{max}$ , the Dencoder will be unable to represent the signal.

The errors produced during the encoding of  $x_1(t)$  and  $x_2(t)$  into the difference signals  $\hat{x}_1(t)$  and  $\hat{x}_2(t)$  are equally distributed in time. This means that any positive amplitude change  $\kappa$  is represented by a repeating pattern of two integers ( $k$  and  $l$ ) scaled by the quantization step interval  $\delta$  such that

$$k = \left\lceil \frac{\kappa}{\delta} \right\rceil \quad (5.38)$$

$$l = \left\lfloor \frac{\kappa}{\delta} \right\rfloor \quad (5.39)$$

where  $\kappa$  represents the change in amplitude between samples. If the number of  $k$  samples in the sequence is represented by  $N_k$  and the number of  $l$  samples in the sequence is represented by  $N_l$  then for any positive amplitude change  $\kappa$ ,

$$\frac{\kappa}{\delta} = \frac{kN_k + lN_l}{N_k + N_l} \quad (5.40)$$

By separating Equation 5.40 into two equations,

$$\kappa = kN_k + lN_l \quad (5.41)$$

$$\delta = N_k + N_l \quad , \quad (5.42)$$

the number of  $k$  and  $l$  integers in the sequence is found to be

$$N_k = \frac{\delta l - \kappa}{l - k} \quad (5.43)$$

$$N_l = \frac{\kappa - \delta k}{l - k} \quad . \quad (5.44)$$

The number of consecutive  $k$  integers per  $l$  in a difference signal with a constant amplitude change  $\kappa$  is the ratio

$$\frac{N_k}{N_l} = \frac{\delta l - \kappa}{\kappa - \delta k} \quad . \quad (5.45)$$

If (for convenience) we assume

$$\delta = 1 \quad (5.46)$$

then

$$\frac{N_k}{N_l} = \frac{l - \kappa}{\kappa - k} \quad . \quad (5.47)$$

Since

$$k = [\kappa] \quad (5.48)$$

$$l = \lfloor \kappa \rfloor \quad (5.49)$$

the number of consecutive  $ks$  per  $l$  is

$$\frac{N_k}{N_l} = \frac{\lfloor \kappa \rfloor - \kappa}{\kappa - [\kappa]} \quad . \quad (5.50)$$

If the maximum summing input to the Dadder is maintained over more than one sample the carry will accumulate. The maximum sum can only be maintained

when both signals are at the  $k$  level, so the number of samples that the carry will accumulate, called  $N_a$ , is

$$N_a = \min \left\{ \frac{\lfloor \kappa_1 \rfloor - \kappa_1}{\kappa_1 - \lceil \kappa_1 \rceil}, \frac{\lfloor \kappa_2 \rfloor - \kappa_2}{\kappa_2 - \lceil \kappa_2 \rceil} \right\} \quad (5.51)$$

because this is the maximum number of consecutive, simultaneous, maximum summations that may occur for any constant amplitude change input.

The sum in the Dadder will be the maximum when the Dadder is stimulated with two constant inputs, such that

$$\Delta_{max} = \kappa_1(t) + \kappa_2(t) \quad (5.52)$$

By combining Equation 5.51 and Equation 5.52 the maximum number of carry accumulation samples is

$$N_{high} = \min \left\{ \frac{\lfloor \kappa_1 \rfloor - \kappa_1}{\kappa_1 - \lceil \kappa_1 \rceil}, \frac{\lfloor \Delta_{max} - \kappa_1 \rfloor - (\Delta_{max} - \kappa_1)}{(\Delta_{max} - \kappa_1) - \lceil \Delta_{max} - \kappa_1 \rceil} \right\} \quad (5.53)$$

The maximum number of carry accumulation samples is shown in Figure 5.6, which assumes  $\Delta_{max}$  to be 10. The choice of  $\Delta_{max}$  is arbitrary, so the maximum number of carry accumulation samples  $N_{high}$  will always be 1 for any set of input amplitude changes that obey Equation 5.52.

The longest simultaneous consecutive carry accumulation sample points occur when

$$\kappa_{x1} = \kappa_{x2} = \left\{ \frac{\Delta_{max}}{2}, 3\frac{\Delta_{max}}{2}, 5\frac{\Delta_{max}}{2} \dots \right\} \quad (5.54)$$

Under these conditions the two difference input signals are

$$\hat{x}_1 = \{\dots, k, l, k, l, k, l, k, \dots\} \quad (5.55)$$

$$\hat{x}_2 = \{\dots, k, l, k, l, k, l, k, \dots\} \quad (5.56)$$

The carry will increment by 1 on every  $k$  sample since

$$k + k = 2k \quad (5.57)$$

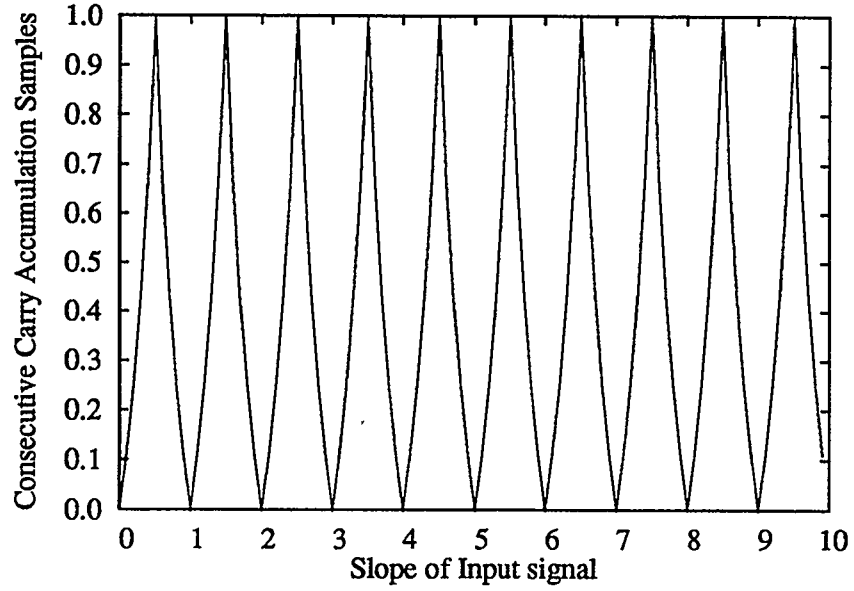


Figure 5.6: Maximum number of consecutive carry accumulation samples.

$$= 2 \lceil \kappa \rceil \quad (5.58)$$

$$= 2 \left( \kappa + \frac{1}{2} \right) \quad (5.59)$$

$$= 2\kappa + 1 \quad (5.60)$$

$$= \Delta_{max} + 1 \quad , \quad (5.61)$$

and the carry will decrement by 1 on every  $l$  sample since

$$l + l = 2l \quad (5.62)$$

$$= 2 \lfloor \kappa \rfloor \quad (5.63)$$

$$= 2 \left( \kappa - \frac{1}{2} \right) \quad (5.64)$$

$$= 2\kappa - 1 \quad (5.65)$$

$$= \Delta_{max} - 1 \quad . \quad (5.66)$$

Therefore, the maximum carry required in the Dadder is 1 level representing an increment of  $\delta$ .

The derivation presented is valid for positive amplitude changes, but maximum

carry for negative amplitude changes can be found by similar means to be a -1 level. In total, three states are required by the Dadder to store the carry:  $\{-1, 0, 1\}$  which represent  $\{-\delta, 0, \delta\}$ .

### Error in the Dadder

The error in the Dadder can be found for inputs quantized to any number of levels,  $L$ . If the rounding scheme of the system is chosen such that the sum requires a maximum of 3 levels for representation, the noise is introduced only by the quantization of the inputs. The quantization noise is modeled as noise sources  $\eta_1$  and  $\eta_2$ . In such a case, assuming the difference sum  $\hat{y}(n)$  is always representable, the output of the Dadder is

$$\hat{y}(n) = \hat{x}_1(n) + \hat{x}_2(n) + C(n)$$

where  $\hat{x}$  is the quantized difference signal  $x(n)$  and  $C$  is the carry which can take values of  $-\delta, 0, \delta$ . The quantization noise is modeled as noise sources in the inputs,

$$\hat{y}(n) = x_1(n) + \eta_1(n) + x_2(n) + \eta_2(n) + C(n)$$

This equation is regrouped to clearly show the summation of the signal and the error.

$$\hat{y}(n) = \underbrace{x_1(n) + x_2(n)}_{y(n)} + \underbrace{\eta_1(n) + \eta_2(n)}_{\eta_y(n)} + C(n) \quad (5.67)$$

where  $C(n)$  can be 0, or  $\pm\delta$ . The greatest possible error in such a configuration is the sum of the maximum input errors and a possible carry error if the initial conditions are in error,

$$\epsilon_{total} = \epsilon_1 + \epsilon_2 + C(0) \quad , \quad (5.68)$$

where  $\epsilon_{total}$  is the error at the output of the Dadder,  $\epsilon_1$  is the error in the first input signal,  $\epsilon_2$  is the error in the second input signal,  $C(0)$  is the initial carry. The maximum values of the noise sources depend on the quantization scheme implemented.

For truncation rounding the maximum errors are

$$\epsilon_1, \epsilon_2 = \delta \quad , \quad (5.69)$$

while for magnitude rounding the maximum errors are

$$\epsilon_1, \epsilon_2 = \frac{\delta}{2} \quad . \quad (5.70)$$

An error in the initial carry will appear at the output as a DC offset. It will be a constant error of either  $\delta$  or  $-\delta$ . If the Dadder is initialized with zero carry ( $C(0) = 0$ ) and the input signals have  $x(0) = 0$  the absolute, total error will be

$$\epsilon_{total} \leq |\epsilon_1 + \epsilon_2| \quad . \quad (5.71)$$

It was mentioned earlier that the Dadder operates properly on input signals provided the error in the input signals is well distributed in time. Unfortunately, the output of the Dadder does not have this quality and this leads to larger errors in systems involving consecutive additions. If the addition of two inputs with amplitude changes of  $\kappa'_1 = \kappa'_2 = \frac{1}{4}$  is considered, the two input sequences generated for  $L = 3$  are

$$\hat{x}_1 = \dots 1000100010001000 \dots \quad (5.72)$$

$$\hat{x}_2 = \dots 1000100010001000 \dots \quad . \quad (5.73)$$

Using a Dadder these sequences will sum to

$$\hat{y} = \hat{x}_1 + \hat{x}_2 = \dots 1100110011001100 \dots \quad . \quad (5.74)$$

The new sequence  $\hat{y}$  represents a amplitude change of  $\kappa' = \frac{1}{2}$ , but the error in  $\hat{y}$  is twice that of either input  $\hat{x}_1$  or  $\hat{x}_2$ . If two such sequences are to be added together the carry cannot be represented by a three states—five states would be required.

This error doubling (and carry increase) points to a serious limitation to the application of the Dadder. Only if two inputs are to be added with no subsequent

adding can the Dadder be used with a single three state carry. In FIR structures the error in the output increases for each Dadder in an addition sequence, but it is possible to build FIR structures by increasing the carry of the later Dadders in the cascade.

In IIR structures the limitation is much more serious. The cycling of the filter state signals means that at no time can the errors be assumed to be equally distributed in time and the Dadder could increase the error to an intolerable level. In such a situation the carry required cannot be easily estimated.

Previous investigations avoid the escalating error by using add divide by  $m$  (where  $m < \frac{1}{2}$ ) [8, 19, 21]. The division reduces the input to a representable level, and bounds the maximum error. The maximum error in such systems is bounded because the error is an infinite geometric sum of a fraction less than  $\frac{1}{2}$ .

The same technique can be used in implementations with the Dadder. Assume that the cascade of Dadders has inputs such that all inputs have a maximum error of  $\epsilon_{\text{input}}$ . If a multiplier with a coefficient  $m < 1$  occurs immediately after every Dadder in the cascade the error will be bounded for any length of cascade to

$$\epsilon_{\text{Dadder}} = (\dots(((\epsilon_{\text{input}} + \epsilon_{\text{input}})m + \epsilon_{\text{input}})m + \epsilon_{\text{input}})m \dots + \epsilon_{\text{input}})m \quad (5.75)$$

$$= \epsilon_{\text{input}} (m + m^2 + m^3 + \dots + m^{n-1} + 2m^n) \quad (5.76)$$

where  $n$  is the length of the cascade. When  $m \leq 0.5$ ,

$$\epsilon_{\text{Dadder}} \leq \epsilon_{\text{input}} \quad , \quad (5.77)$$

and when  $m = 0.5$ ,

$$\epsilon_{\text{Dadder}} = \epsilon_{\text{input}} \quad . \quad (5.78)$$

### Dadder Simulation

The Dadder is simulated and tested with two random input waves. Two random waves are generated such that the maximum change in the sum of the inputs is

always representable, that is, the sum of the amplitude changes is less than the quantization step interval. Random wave inputs were chosen as test inputs because all possible input sequences can be encountered. The longer the simulation is run the more likely the greatest possible error will be encountered. For the simulations shown, only 10 000 samples were used, but this still encounters errors very close to the analytical maximum of twice the input error.

The output of the Dadder and the ideal response are both shown in Figure 5.7. The Dadder system simulated uses three states ( $L = 3$ ) with magnitude rounding in the quantizers. The step quantization interval  $\delta = 0.01$ . The maximum integer representable is 1 so all inputs and outputs are members of  $\{-\delta, 0, +\delta\}$ . The error (Figure 5.8) at the demodulated output of the Dadder is always less than twice the input error. Errors due to the carry are not encountered because both random input waves begin at 0 and the carry is initialized to 0. If the carry is initialized to  $\delta$  or  $-\delta$  the error in the output is the random error encountered before plus the initial error in the carry. The carry produces a DC offset error in the output.

#### 5.4.2 Multiplying Difference Signals

In earlier work, multipliers were created from the cascade of add divide by  $m$  functions [8, 19, 21]. An alternative to this approach is to construct the multipliers directly from state-machines. The state-machine format leads to a simple ROM implementation and in this section such a multiplier is examined.

##### Structure

Using a simple look-up table to produce a multiplier for difference signals would result in very large errors in the output due to the few states available to represent the output at high oversampling rates. If the input and output are both represented by three states  $(+1, 0, -1)$ , then multiplication using a look-up table (other than mul-



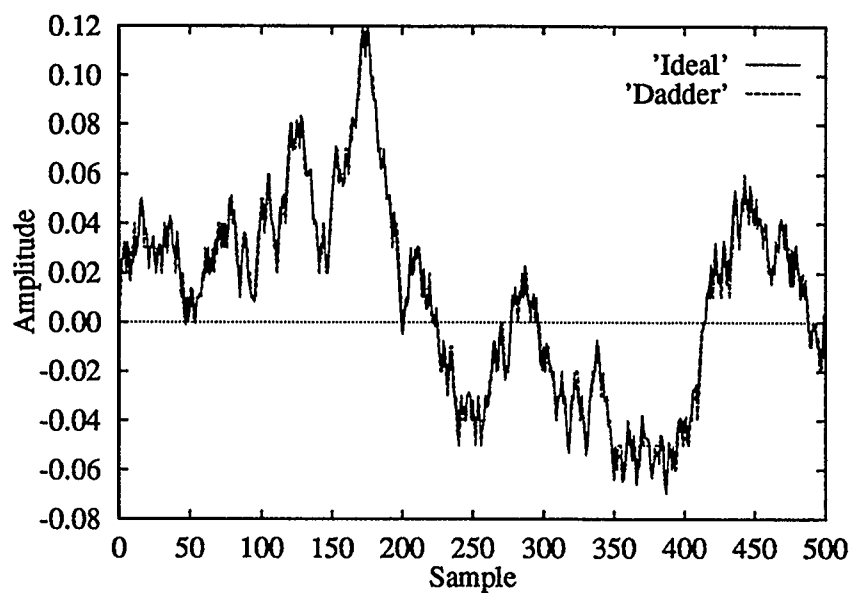


Figure 5.7: Ideal sum and Dadder sum output waveforms for the random input sequence.

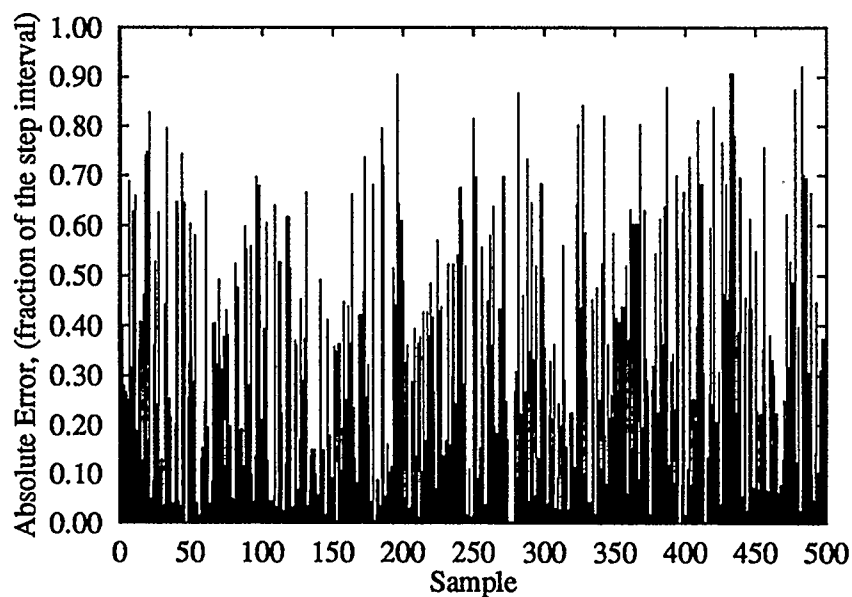


Figure 5.8: Error between the output and the ideal response in the Dadder for random waveform inputs.

multiplication by  $m = +1$ ,  $m = 0$  or  $m = -1$ ) is unsatisfactory, because all multiplier results must be rounded to one of the three available states. Rounding of the output effectively rounds the multiplier coefficient to  $m = +1$ ,  $m = 0$ , or  $m = -1$ . Using a carry to quantize the error to a smaller quantization interval than the signal quantization interval reduces the error internal to the multiplier and a more accurate multiplication results. A difference multiplier (Dmult) is shown in block diagram form in Figure 5.9. The dashed box represents the programmed ROM block.

The Dmult can be modeled as

$$\hat{y}(n) = Q[m\hat{x}(n) + C(n)] \quad (5.79)$$

$$C(n+1) = Q_c[m\hat{x}(n) + C(n) - \hat{y}(n)] \quad , \quad (5.80)$$

where  $Q_c[\cdot]$  is the quantization scheme of the carry,  $m$  is the multiplication constant,  $C$  is the carry of the multiplier,  $\hat{x}(n)$  is the input difference signal and  $\hat{y}(n)$  is the output difference signal.

The quantization scheme of the carry  $Q_c[\cdot]$  is determined by  $Q[\cdot]$  and the multiplier coefficient. The higher the accuracy (i.e., longer the wordlength) of the multiplication coefficient the larger the carry required.

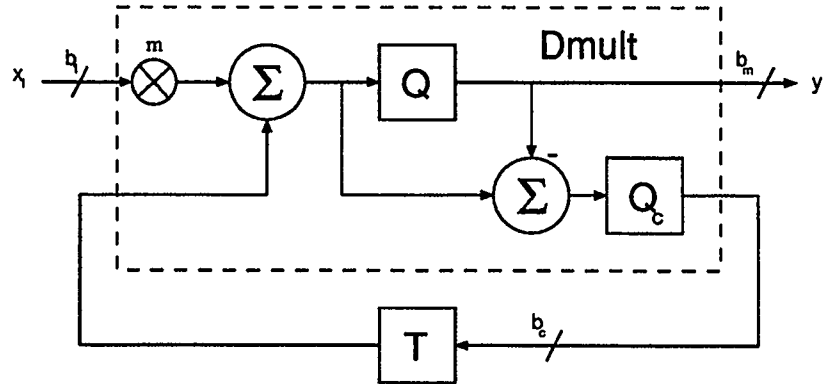


Figure 5.9: Block diagram of the difference multiplier (Dmult).

### The Dmult Coefficient and the Carry Quantization Scheme

This discussion is restricted to Dmults with absolute coefficients less than 1. Absolute coefficients greater than 1 are possible but require output wordlengths larger than the input wordlength and a non-constant wordlength creates difficulties when implementing IIR structures. Coefficients greater than 1 can be constructed using combinations of Dmults and Dadders.

The signal input to the Dmult is assumed to be quantized to the same scheme as the output.  $N$  designates the integer (or encoded) quantized value of the input. Just as the input signal  $x(n)$  has a dynamic range of  $-A$  to  $A$  the coded quantized value  $N$  will have a coded dynamic range between  $-N_{max}$  to  $+N_{max}$ , where  $N_{max}$  is the maximum coded integer. The maximum coded integer is

$$N_{max} = \frac{L-1}{2} \quad , \quad (5.81)$$

where  $L$  is the number of levels in the quantizer. The Dmult coefficient,  $m$ , will reduce the coded output dynamic range to  $+mN_{max}$  to  $-mN_{max}$ . The final output of the Dmult  $\hat{y}$  will be within this new quantized range  $Q[mN_{max}] \geq \hat{y} \geq Q[-mN_{max}]$ .

For a small  $N_{max}$  the error,  $Q[mN] - mN$ , will accumulate in the decoder and cause large errors in the output. By storing the error in a carry and accumulating the error until it may be represented at the output, the error can be reduced. To avoid DC errors, the the carry must not overflow and the carry must have sufficient states to allow all possible errors to be represented.

One method is to allow the carry to represent a decimal number system. The smallest increment in the carry is equal to the smallest representation in the Dmult coefficient. For instance, if the Dmult coefficient is accurate to  $1 \times 10^{-5}$ , then  $1 \times 10^{-5}$  is the smallest increment the carry need represent, called  $C_{step}$ .

If rounding is used in the quantization scheme the carry need not represent a full quantization step (represented by an integer 1), it need only represent the steps

between  $-\frac{1}{2}$  and  $\frac{1}{2}$ . Representations between  $-1$  and  $1$  are required for truncation rounding. In this analysis, only rounding quantization will be considered.

All possible carries  $C$  exist in a finite set

$$\begin{aligned}
 C \in & \left\{ -\frac{1}{2}, -\frac{1}{2} + C_{step}, -\frac{1}{2} + 2C_{step} \dots \right. \\
 & \left. \dots - C_{step}, 0, C_{step} \dots \frac{1}{2} - 2C_{step}, \frac{1}{2} - C_{step}, \frac{1}{2} \right\} \\
 C \in & \bigcup_{n=0}^{\frac{1}{C_{step}}} \left( -\frac{1}{2} + nC_{step} \right) \quad .
 \end{aligned} \tag{5.82}$$

For any Dmult coefficient  $m$  defined in base 10 where  $|m| \leq 1$ ,  $C_{step}$  can be found from

$$C_{step} = 10^{-D} \quad , \tag{5.83}$$

where  $D$  is the number of decimal places in the coefficient. This creates a truncated decimal system where the smallest required interval is represented.

For some coefficients such system leads to wasted states. For example, a coefficient of 0.5 will require

$$C_{step} = 10^{-1} = .1 \tag{5.84}$$

leading to the states

$$-0.5, -0.4, -0.3, \dots -0.1, 0, 0.1, \dots 0.3, 0.4, 0.5 \tag{5.85}$$

or eleven states in total. Since all additions and subtractions of integers multiplied by 0.5 will always lead to multiples of 0.5, the quantizer need only store 0.5, 0,  $-0.5$ . The accuracy of the output is unaffected since no other states are ever encountered by the carry.

The wasted states created by setting  $C_{step}$  using Equation 5.83 can be avoided by determining the largest quantization interval that will represent all additions and subtractions of multiples of  $m$ . This value will be the most efficient choice for  $C_{step}$ ,

because it will require the least number of states. Once  $C_{step}$  is known the number,  $L_{carry}$ , of states required is

$$L_{carry} = \frac{1}{C_{step}} + 1 \quad . \quad (5.86)$$

Consider a Dmult coefficient  $m = 0.015$ . From Equation 5.83,  $C_{step} = 0.001$  and 1001 states are required. If the interval is chosen to be 0.005 no reduction of accuracy occurs and only 201 states are required, which is a considerable savings.

Using either approach the total number of states can be found for any Dmult coefficient, but for  $m = -1$  the carry structure of the Dmult is eliminated. Since all multiplications by  $-1$  are representable integers, no carry is required. The Dmult becomes a look-up table where the output is simply the input negated.

### Error in the Dmult

Three quantizations influence the error at the output of the Dmult. First, the input quantization can be modeled by adding a noise source to the input of the Dmult. Next, the output of the multiplier is quantized to the same quantization scheme as for the input. Once again, a noise source can be added to model this influence. Finally, the carry is quantized producing an error, which can be modeled as a noise source. Since all possible errors are represented by the carry, so the noise due to the carry,  $\eta_3 = 0$ . The quantization point in the carry path can be modeled by a direct feed through path.

When the quantization errors are modeled as noise sources, Figure 5.10, the system becomes linear and superposition can be applied. If  $\eta_2 = 0$  the entire carry branch is always 0. The output is

$$mx + m\eta_1 \quad (5.87)$$

where  $\eta_1$  is the noise in the input quantization scheme, and  $x$  is the input. It is identical to  $\hat{x}$  the quantized difference signal of the input  $x$ .

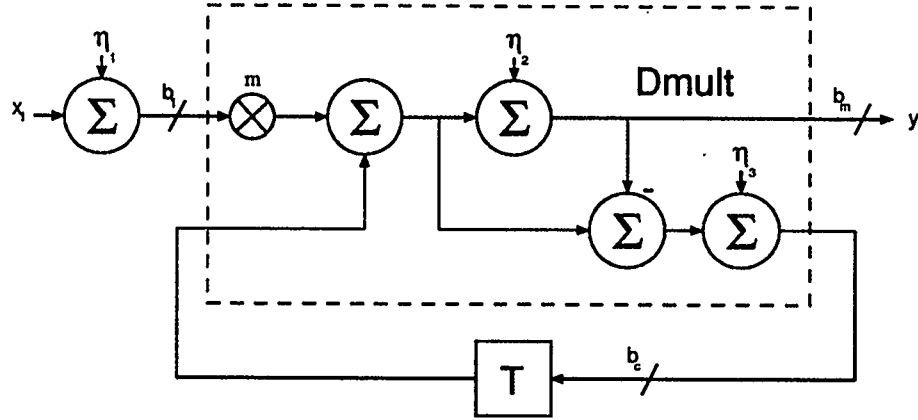


Figure 5.10: The Dmult with quantizers modeled with noise.

The input  $x$  and the input noise source  $\eta_1$  are then set to zero to allow the effect of  $\eta_2$  to be examined. The carry is the negative of  $\eta_2$ , and will act to reduce the previous output error. This means that, at any point in time, all error before the present moment has been canceled. Since all other signal sources are zero, only  $\eta_2(n)$  affects the output and the output of the system is  $\eta_2$ . The total error in the Dmult is

$$\epsilon_{\text{Dmult}} = m\eta_1 + \eta_2 \quad (5.88)$$

The quantization schemes generating  $\eta_1$  and  $\eta_2$  are identical, so the maximum quantization error  $\epsilon$  is

$$\epsilon \geq \eta_1, \eta_2 \quad (5.89)$$

The maximum error in the Dmult from analog input to analog output is

$$\epsilon_{\text{Dmult}} = (1 + m)\epsilon \quad (5.90)$$

where for magnitude rounding quantization,

$$\epsilon = \frac{1}{2}\delta \quad (5.91)$$

and for truncation quantization,

$$\epsilon = \delta \quad (5.92)$$

The error analysis for the Dmult avoids errors caused by improper initialization of the carry. If the initial carry is a state other than zero, a DC error would occur in the output. The total possible error including the possibility of carry error is

$$\epsilon_{\text{Dmult}} = (1 + m) \epsilon + C(0) \quad (5.93)$$

Since

$$-\epsilon \leq C(0) \leq \epsilon \quad (5.94)$$

the maximum possible error for any initial condition is

$$\begin{aligned} \epsilon_{\text{Dmult}} &= (1 + m) \epsilon + \epsilon \\ &= (2 + m) \epsilon \end{aligned} \quad (5.95)$$

The maximum error encountered in a simulation of the Dmult for coefficients between 0 and 1 is shown in Figure 5.11. The simulation assumes a zero initial carry and the results of the simulation reinforce the analysis presented.

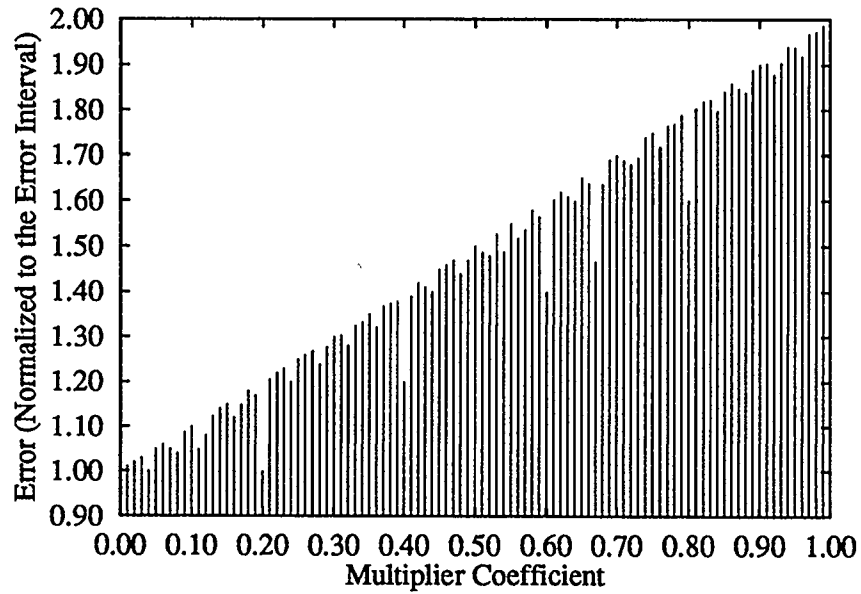


Figure 5.11: Dmult error as a function of dmult coefficient.

### Dmult Simulation

The Dmult is simulated with a random input signal. The random input wave allows any sequence of inputs, provided the sequence is representable in the system. Errors very close to the maximum possible error are likely to be encountered for sufficiently long sequences.

The simulation used magnitude rounding in both the input quantization scheme and the carry quantization scheme. The Dmult coefficient  $m = 0.95$ . The output and the ideal response are shown in Figure 5.12. The error in the signal (shown in Figure 5.13 normalized to delta) is always less than

$$(1 + m) \epsilon = (1 + 0.95) \frac{\delta}{2} = 0.975\delta \quad (5.96)$$

The maximum encountered error ( $0.9732 \delta$ ) is less than the calculated maximum ( $0.975 \delta$ ) because of the nature of the random wave test inputs. If the test were allowed to continue, errors closer to the maximum would eventually be encountered.

### Memory Requirements of the Dmult

The Dmult is constructed as a ROM state machine with  $b_i$  bits for the input,  $b_o$  bits for the output and  $b_c$  bits for the carry. For a Dmult built as a ROM block the memory required, in bits, is

$$\mathcal{M}_{\text{Dmult}} = (b_i + b_c) 2^{(b_o + b_c)} \quad (5.97)$$

The number of bits required by the carry is related to  $C_{\text{step}}$  and, therefore, to the accuracy of the Dmult coefficient. If the Dmult carry is encoded using Equation 5.83 (for simplicity),  $b_c$  can be defined (for rounding quantization) as

$$\begin{aligned} b_c &= \left\lceil \log_2 \left( \frac{1}{C_{\text{step}}} + 1 \right) \right\rceil \\ &= \left\lceil \log_2 \left( \frac{1}{10^{-D}} + 1 \right) \right\rceil, \end{aligned} \quad (5.98)$$



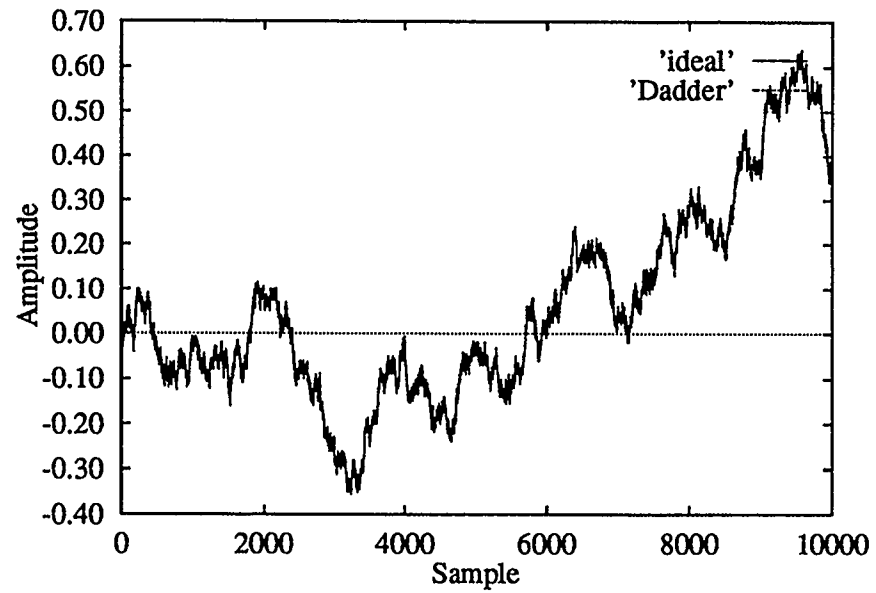


Figure 5.12: Dmult output and ideal response for  $m = 0.95$ .

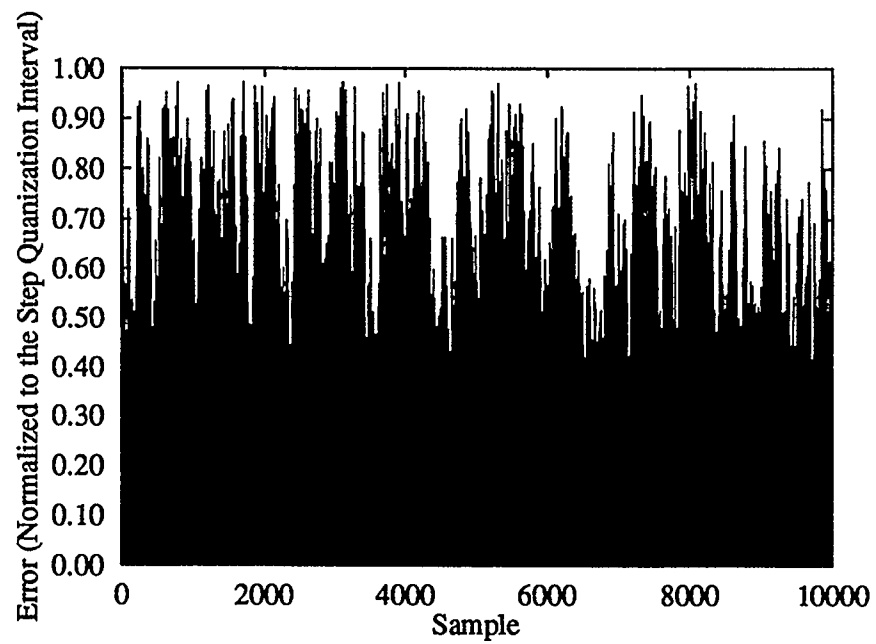


Figure 5.13: Error in the Dmult output for  $m = 0.95$ .

where  $D$  is the number of decimal places in the Dmult coefficient, which is a measure of the accuracy of the coefficient.

Assuming the system is sufficiently oversampled to reduce both  $b_i$  and  $b_o$  to the minimum three states, the memory requirements for a Dmult can be found as a function of Dmult coefficient accuracy  $D$  (Table 5.2). The table shows an exponential rise in memory requirements as the accuracy of the coefficient increases. For a Dmult coefficient accuracy of  $D = 4$  (accurate to  $\pm 0.0001$ ), the memory requirements already exceed a megabit for one multiplier.

The memory requirements of a Dmult can be reduced by cascaded multiplications. If the overall multiplication coefficient is to be  $m$ , two consecutive multiplications by  $\sqrt{m}$  will perform the same operation. The choice of the two multiplication coefficients is not restricted to the square root of the overall coefficient  $m$ . The two coefficients  $m_a$  and  $m_b$  must only obey the condition

$$m = m_a \times m_b \quad (5.99)$$

Coefficient Accuracy $D$	Memory Required in bits
1	$3.8 \times 10^2$
2	$4.6 \times 10^3$
3	$4.9 \times 10^4$
4	$1.0 \times 10^6$
5	$1.0 \times 10^7$
6	$9.2 \times 10^7$
7	$1.7 \times 10^9$
8	$1.6 \times 10^{10}$
9	$1.4 \times 10^{11}$
10	$2.5 \times 10^{12}$
11	$2.1 \times 10^{13}$
12	$1.8 \times 10^{14}$

Table 5.2: Memory requirements for a Dmult for various coefficient wordlengths.

By proper choice of  $m_a$  and  $m_b$ , multiplication by  $m$  becomes possible using less memory. Consider  $m = 0.0001$  as an example. If a Dmult were to be constructed directly the carry would require 10 001 states (rounding quantization) because

$$\begin{aligned} L_{carry} &= \frac{1}{C_{step}} + 1 \\ &= \frac{1}{0.0001} + 1 \\ &= 10\,001 \end{aligned} \quad (5.100)$$

If two Dmults with coefficients  $m_a = m_b = 0.01$  are cascaded, each would only require 101 states since

$$\begin{aligned} L_{carry} &= \frac{1}{C_{step}} + 1 \\ &= \frac{1}{0.01} + 1 \\ &= 101 \end{aligned} \quad (5.101)$$

The two Dmults constructed to implement  $m_a$  and  $m_b$  will require less total memory than a single Dmult implementing  $m$ .

In choosing  $m_a$  and  $m_b$ , care must be exercised to closely approximate  $m$  without requiring more memory than a direct Dmult implementation would require. Filter structures which are relatively insensitive to changes in multiplier coefficients (such as LDI and wave structures) are well suited for this technique since it is simple to approximate  $m$  with  $m_a$  and  $m_b$ , but it is more difficult to choose  $m_a$  and  $m_b$  such that  $m_a \times m_b = m$ .

This technique is useful for filters that have very small multiplier coefficients, since the overall filter coefficient  $m$  must be less than either  $m_a$  or  $m_b$ . Fortunately filters for use with highly oversampled signals will have such coefficients.

By cascading Dmults the memory requirement for the overall multiplication operation is reduced; however the memory reduction is traded off against an increased

error at the output of the multiplying system. The error due to one Dmult (from Equation 5.90) is

$$\epsilon_{\text{Dmult}} = (1 + m)\epsilon \quad ,$$

where  $\epsilon$  is the quantization error of the input. For a cascade of two Dmults the error at the output is

$$\epsilon_{\text{Dmult2}} = (m_2(1 + m_1) + 1)\epsilon \quad , \quad (5.102)$$

where  $m_1$  is the first Dmult coefficient, and  $m_2$  is the second Dmult coefficient.

For an arbitrary length of cascaded Dmults  $M$  multipliers long,

$$\epsilon_{\text{DmultM}} = \left( \left( \sum_{i=1}^M \prod_{n=i}^M m_n \right) + 1 \right) \epsilon \quad . \quad (5.103)$$

Equation 5.103 shows that the order of multiplications has an impact on the overall error. The last Dmult coefficient multiplies the error of all others. By ensuring that the multiplication coefficients are arranged in order of descending magnitude

$$m_1 \geq m_2 \geq m_3 \geq \dots m_{M-1} \geq m_M \quad (5.104)$$

the overall error  $\epsilon_{\text{DmultM}}$  is minimized. The smaller error coefficients later in the multiplier cascade reduce the errors due to large coefficients earlier in the multiplier cascade.

## 5.5 Filter Construction

Using Dadder and Dmult components, it is possible to construct digital filters for difference signals. To gain any advantage from such an implementation it is necessary to greatly oversample the input signal, that is

$$f_s \gg f_{\max} \quad . \quad (5.105)$$

The oversampling ratio of the system,  $R$ , is defined as

$$R = \frac{f_s}{2f_{\max}} \quad . \quad (5.106)$$

The greater the oversampling ratio, the smaller the wordlength of the difference signal and the smaller the amount of memory required for implementation. Unfortunately, as the oversampling ratio increases, the bandwidth of the filter is reduced relative to the total bandwidth of the system and at very high oversampling rates the bandwidth of the filter reduces to a narrow band at the low end of the spectrum. The narrowing of the bandwidth of the filter increases the required accuracy of the internal states and multiplier coefficients. Longer wordlengths would be needed to represent the states and multiplier coefficients.

In the  $z$ -plane, the effect of oversampling and the rise in required accuracy can be easily observed. As  $R$  increases, the poles of the transfer function of the filter are crushed into a small wedge in the unit circle (Figure 5.14). The filter uses quantized coefficients and states, so all points in the  $z$ -plane are quantized to the nearest representable point. Only a finite number of points is available and the points are spread in an even grid on the plane. As the poles move closer together a finer mesh of points is required to distinguish each pole. The finer mesh of points requires more overall points which requires an increase in the number of super-states of the filter.

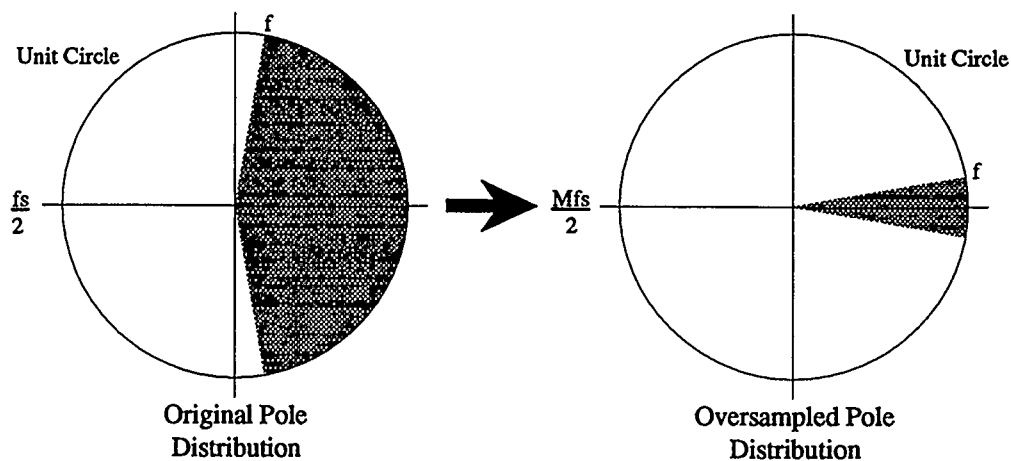


Figure 5.14: The crushing of poles and zeros into a small wedge by oversampling.

To increase the number of super-states, the wordlengths of the internal states and the multiplier coefficients are increased. When the wordlength required to represent the filter increases the memory requirements for a full ROM difference implementation increases. In fact, the increase in wordlength caused by the narrowing of the bandwidth will quickly remove any advantage that a difference filter once possessed. Fortunately, this problem can be avoided by proper filter construction. To avoid constructing narrow band filters the bandlimited nature of the input signal is exploited.

The difference signal produced by the Dencoder (sampling at  $f_s$ ) has the same bandwidth as the input signal, because the Dencoder is a linear system and any linear system can modify input signals in amplitude and phase, but not frequency. Since the input signal  $x(n)$  is bandlimited to  $f_{max}$  it is highly oversampled and can be decimated by  $M$ , where  $M \leq R$ . After decimation the signal can be filtered at

$$f' = \frac{f_s}{M} \quad (5.107)$$

$$= \frac{2 f_{max} R}{M} \quad (5.108)$$

provided that

$$f' \geq 2f_{max} \quad (5.109)$$

If  $M$  is chosen such that the internal sample rate of the filter  $f' = 2f_{max}$  the filter will have a maximum spread of poles on the unit disc because the internal sample rate of the filter will be at the minimum possible value. The maximum spread of poles on the unit disc will allow a small wordlength for the filter states and multiplier coefficients.

After filtering, the decimated signal must be upsampled to  $f_s$  for use by the Ddecoder to produce the final output waveform. The upsampling implies an insertion of  $M - 1$  zeros and lowpass filtering. An alternative to the upsampling and lowpass filtering of the signal is to filter the difference signal  $\hat{x}(n)$  in an interleaved filter [2].

In the interleaved filter the input signal is divided into  $M$  streams each of which is a decimated difference signal offset from the others by a unit delay. The signals are processed by  $M$  separate filters at

$$f' = f_s/M \quad (5.110)$$

and output signal of each filter is interleaved with others to produce the output filtered difference signal.

Consider the behaviour of the filter with an arbitrary sampled input signal,  $x(n)$ . First the input is downsampled by splitting it into  $M$  signals each of which is separated by one sample

$$\begin{aligned} x_0(n) &= x(nM) \\ x_1(n) &= x(nM - 1) \\ x_2(n) &= x(nM - 2) \\ &\vdots \\ x_{M-1}(n) &= x(nM - (M - 1)) \end{aligned} \quad (5.111)$$

The signals are filtered at

$$f' = f_s/M$$

by the filter transfer function  $H(\cdot)$  producing a set of  $M$  output signals,

$$\begin{aligned} y_0(n) &= H(x_0(n)) \\ y_1(n) &= H(x_1(n)) \\ y_2(n) &= H(x_2(n)) \\ &\vdots \\ y_{M-1}(n) &= H(x_{M-1}(n)) \end{aligned} \quad (5.112)$$

The signals are upsampled by inserting  $M - 1$  zeros between each sample,

$$\begin{aligned}
 y'_0(n) &= \begin{cases} y_0(\frac{n}{M}) & \text{if } n \in \{\pm iM\}_{i=0}^{\infty} \\ 0 & \text{otherwise} \end{cases} \\
 y'_1(n) &= \begin{cases} y_1(\frac{n+1}{M}) & \text{if } n \in \{\pm iM - 1\}_{i=0}^{\infty} \\ 0 & \text{otherwise} \end{cases} \\
 y'_2(n) &= \begin{cases} y_1(\frac{n+2}{M}) & \text{if } n \in \{\pm iM - 2\}_{i=0}^{\infty} \\ 0 & \text{otherwise} \end{cases} \\
 &\vdots \\
 y'_{M-1}(n) &= \begin{cases} y_1(\frac{n+(M-1)}{M}) & \text{if } n \in \{\pm iM - (M-1)\}_{i=0}^{\infty} \\ 0 & \text{otherwise} \end{cases}
 \end{aligned} \tag{5.113}$$

and the individual signals are summed together to create the final output

$$y(n) = \sum_{j=0}^{M-1} y'_j(n) \tag{5.114}$$

Since only one signal will be non-zero at any given time  $n$ , the addition in Equation 5.114 can be simplified by interleaving or commutating the signals,

$$\begin{aligned}
 y(n) &= \sum_{j=0}^{M-1} y'_j(n) \\
 &= \begin{cases} y'_0(n) & \text{if } n \in \{\pm iM\}_{i=0}^{\infty} \\ y'_1(n) & \text{if } n \in \{\pm iM - 1\}_{i=0}^{\infty} \\ y'_2(n) & \text{if } n \in \{\pm iM - 2\}_{i=0}^{\infty} \\ \vdots & \\ y'_{M-1}(n + (M-1)) & \text{if } n \in \{\pm iM - (M-1)\}_{i=0}^{\infty} \end{cases}
 \end{aligned} \tag{5.115}$$

Although the approach seems indirect, the interleaved signal  $y(n)$  is the filtered signal  $H(x(n))$  since

$$y(n) = \sum_{j=0}^{M-1} y'_j(n)$$



$$\begin{aligned}
&= \begin{cases} y_0(n) & \text{if } n \in \{\pm iM\}_{i=0}^{\infty} \\ y_1(n+1) & \text{if } n \in \{\pm iM-1\}_{i=0}^{\infty} \\ y_2(n+2) & \text{if } n \in \{\pm iM-2\}_{i=0}^{\infty} \\ \vdots & \\ y_{M-1}(n+(M-1)) & \text{if } n \in \{\pm iM-(M-1)\}_{i=0}^{\infty} \end{cases} \\
&= \begin{cases} H(x_0(n)) & \text{if } n \in \{\pm iM\}_{i=0}^{\infty} \\ H(x_1(n+1)) & \text{if } n \in \{\pm iM-1\}_{i=0}^{\infty} \\ H(x_2(n+2)) & \text{if } n \in \{\pm iM-2\}_{i=0}^{\infty} \\ \vdots & \\ H(x_{M-1}(n+M-1)) & \text{if } n \in \{\pm iM-(M-1)\}_{i=0}^{\infty} \end{cases} \\
&= H(x(n)) \quad . \quad (5.116)
\end{aligned}$$

The interleaving filter allows the filtering of a bandlimited input signal using  $M$  filters operating at  $f' = f_s/M$ . The lower sample rate of the internal filters decreases the resolution required within the filter which is reflected in a decreased wordlength. Such a structure is suitable for filtering difference signals since difference signals are highly oversampled and can have reduced wordlengths.

## 5.6 Difference Filter Simulation

To demonstrate the difference operation of the filter a 5<sup>th</sup> order, LDI structure is simulated. The LDI structure is chosen for low sensitivity to coefficient rounding. Exploiting the its low sensitivity, the multiplier carry wordlength is rounded to 2 decimal places. Two decimal place accuracy ( $D = 2$ ) for the multiplier coefficients requires a seven bit carry. From Equation 5.98,

$$b_c = \left\lceil 1 + \log_2 \left( \frac{1}{10^{-D}} + 1 \right) \right\rceil = \left\lceil 1 + -\log_2 (10^{-2}) \right\rceil = \lceil 6.658 \rceil = 7 \quad . \quad (5.117)$$

The lowpass LDI filter was designed and the parameters chosen for the design are

shown in Table 5.3 along with the necessary multiplier coefficients.

The infinite precision magnitude response was calculated and is labeled *ldi5* in Figure 5.15. The multiplier coefficients are rounded to the nearest two decimal places, and the resulting magnitude response of the modified filter is labeled as *ldi5\_trunc* in Figure 5.15. The graph of the magnitude responses demonstrates that the overall transfer function has remained largely the same after the rounding of the multiplier coefficients. If such deviations are tolerable, which would depend on the application of the filter, the filter can be implemented with reduced multiplier coefficient wordlengths. Rounding the coefficients to two decimal places allows, approximately, an eight bit coefficient wordlength. It should be noted that all further error comparisons will be between the difference filter and the rounded multiplier coefficient LDI structure.

It is assumed that the input signal to the LDI filter is bandlimited to

$$f_{max} = 500 \text{ Hz} \quad (5.118)$$

It is necessary to know the maximum frequency of the input to estimate the oversampling required for an implementation with a specific number of levels. The input to the system has a dynamic range of  $-A$  to  $A$ , and in this simulation  $A$  is chosen to be 1, for convenience. If the input signal is quantized to  $b_o = 12$  bits, the quantization interval (the distance between quantization levels) is

$$q = \frac{2A}{2^{b_o}} = \frac{2 \times 1}{2^{12}} = 0.000\,488\,28\text{V} \quad (5.119)$$

Design Parameters		
$f_s$	5 000	Hz
$f_c$	300	Hz
Order	5	
$A_p$	0.5	dB

Multiplier Coefficients	
m1	0.173 449 201 473 824
m2	0.308 060 272 343 644
m3	0.149 102 355 745 068
m4	0.298 975 089 837 714
m5	0.222 565 487 545 064

Table 5.3: Design parameters and multiplier coefficients for the simulated 5<sup>th</sup> order LDI filter .

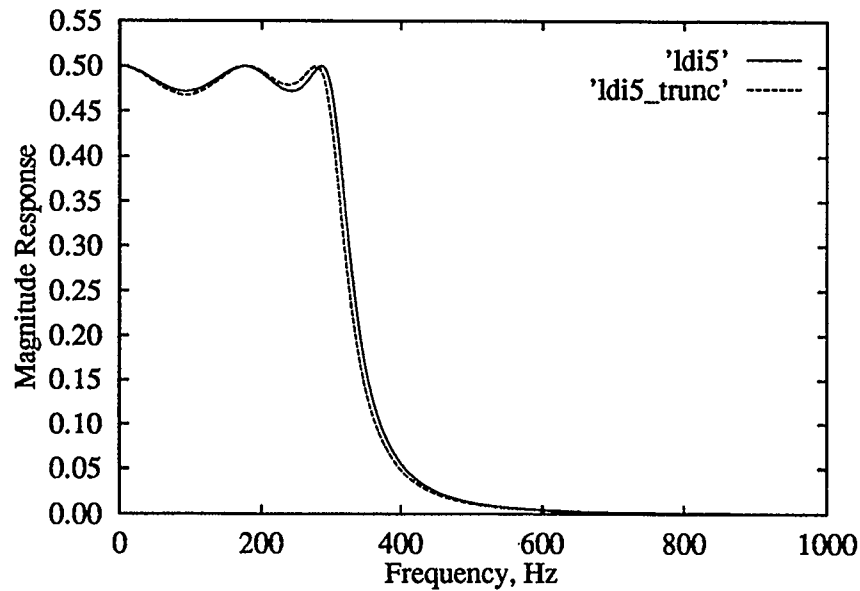


Figure 5.15: Magnitude responses of the ideal and coefficient truncated filters.

It is this quantization interval that is chosen as the step interval  $\delta$  for the difference filter. Using the known dynamic range of the input,  $A$ , and the step interval,  $\delta$ , the necessary oversampling ratio  $R$  is

$$\begin{aligned}
 R &= \frac{A\pi}{\delta} \\
 &= \frac{1 \times \pi}{0.000\,488\,28} \\
 &= 6\,434.0 \quad .
 \end{aligned} \tag{5.120}$$

$R$  is the minimum oversampling rate required to allow the input to be encoded as three states. The oversampling need not be this high if more quantization levels are used to represent the states and the input. In this simulation, the number of levels  $L$  is arbitrarily chosen to be 31 (5 bits). This reduces the oversampling rate to

$$R = \frac{6\,434.0}{(L-1)/2} = \frac{6\,434.0}{(31-1)/2} = 428.93 \quad . \tag{5.121}$$

The LDI filter is constructed with a sample rate  $f' = 5\,000$ , so the oversampling

ratio of the filter (with respect to the input data) is

$$R' = \frac{2f_{max}}{f_s} = \frac{2 \times 500}{5\,000} = 5 \quad (5.122)$$

Thus the difference filter oversampling ratio (and decimation rate)  $M$  is

$$M = \frac{R}{R'} = \frac{428.93}{5} = 85.783 \quad (5.123)$$

which is rounded up to  $M = 86$ .

The required sample rate is  $F_s = 86 \times 5\,000 = 430\,000$  Hz. This is not an unreasonable sampling rate for ROM based technology. In 1992, ROMs typically have access times allowing operation above 25 MHz. If the sample rate of the system is scaled up to 25 MHz, the bandwidth of the input is increased to 30 000 Hz, which will allow filtering of audio range data.

The interleaved filter as presented in Section 5.5 implies the construction of  $M$  filters. Such a construction would be very costly in terms of required hardware, but alternatively the interleaved structure can be constructed by changing the unit delays within the  $M = 1$  filter to delays  $M$  bits long. This effectively produces  $M$  interleaved filters.

In the example LDI filter, the 86 individual filters will operate at 5 000 Hz, but the entire structure and the ROM components will be operating at 430 000 Hz. The interleaved filter implementation using Dmults and Dadders is shown in Figure 5.16, where DA blocks represent the Dadder components, DM blocks represent the Dmult components, '−1' blocks represent the multiply by −1 components and MT blocks represent the  $M$  delay blocks. In this example  $M = 86$ , so the MT blocks are shift registers 86 unit delays long and 5 bits wide.

### 5.6.1 Test Inputs and Results

The difference filter shown in Figure 5.16 was simulated in software. The output difference signal is represented by  $y(n)$ , while  $x(n)$  represents the input difference

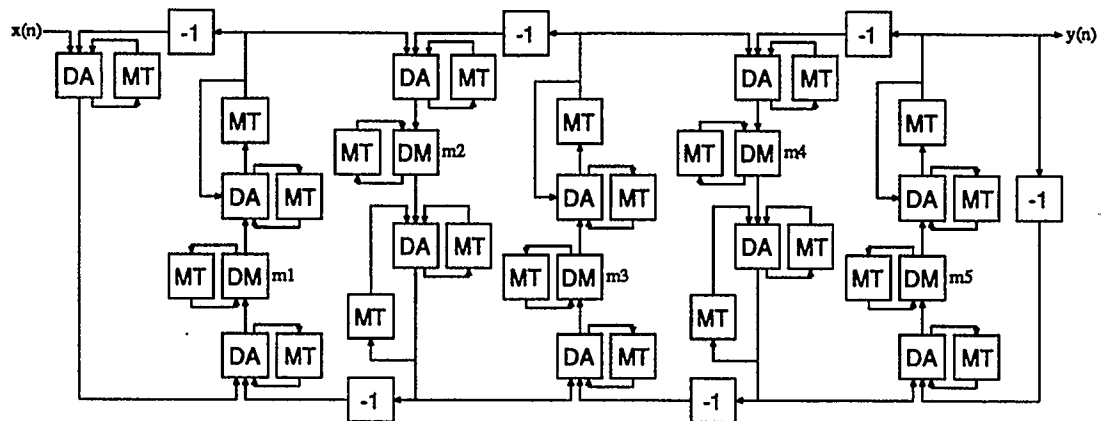


Figure 5.16: A 5<sup>th</sup> order LDI filter constructed from Dmult and Dadder blocks.

signal.

Two test inputs were examined: a random waveform and a Kronecker delta function. The random waveform allows an examination of a large number of irregular transitions on the input. For long input sequences, the errors encountered will approach the maximum possible error. The Kronecker delta input is used to test the linear behaviour of the filter. Using the Kronecker delta input, the magnitude response of the system can be determined and compared against the expected ideal response.

The first test considers the random input waveform 10 000 samples long. The waveform is similar to the random waveforms used to test the individual components. The input waveform  $x$  is allowed to make step transitions of a random size limited by the maximum integer encodable  $L_{max}$  and the step quantization interval  $\delta$  such that at any sample  $n$ ,

$$-\delta \frac{L_{max} - 1}{2} \leq x(n) \leq \delta \frac{L_{max} - 1}{2}. \quad (5.124)$$

In practice, the maximum transitions allowable on the input are limited by the nonlinear nature of the filter. In this simulation, the random input signal is limited

to maximum changes of half of the maximum possible change,

$$-\frac{1}{2}\delta\frac{L_{max}-1}{2} \leq x(n) \leq \frac{1}{2}\delta\frac{L_{max}-1}{2}. \quad (5.125)$$

By limiting the maximum change in the input signal, the possibility of overflow of the internal states is reduced.

The output of the random wave test is shown in Figure 5.17. The error between the output of the infinite precision LDI filter with truncated coefficients and the output of the difference filter is shown in Figure 5.18. The error is normalized to the step quantization interval. The error can be reduced to an arbitrarily small value by reducing the step quantization interval  $\delta$ . This is not a linear relationship, but as  $\delta$  is decreased the error at the output of the filter will decrease. To decrease  $\delta$ , the sampling rate may be increased, or the number of quantization levels may be increased. The increase in sample speed is limited by the implementation technology, while the increase in the number of quantization levels ( $L$ ) is limited by the memory

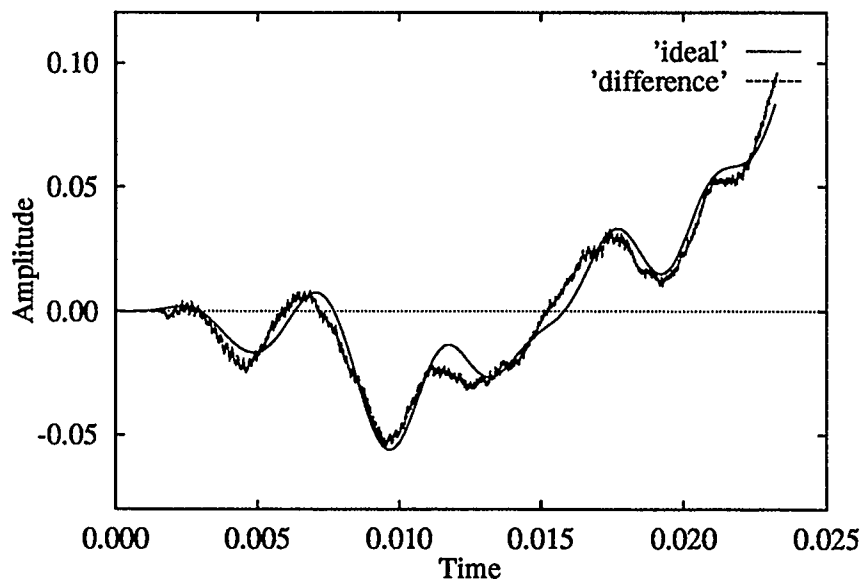


Figure 5.17: Random waveform test output.

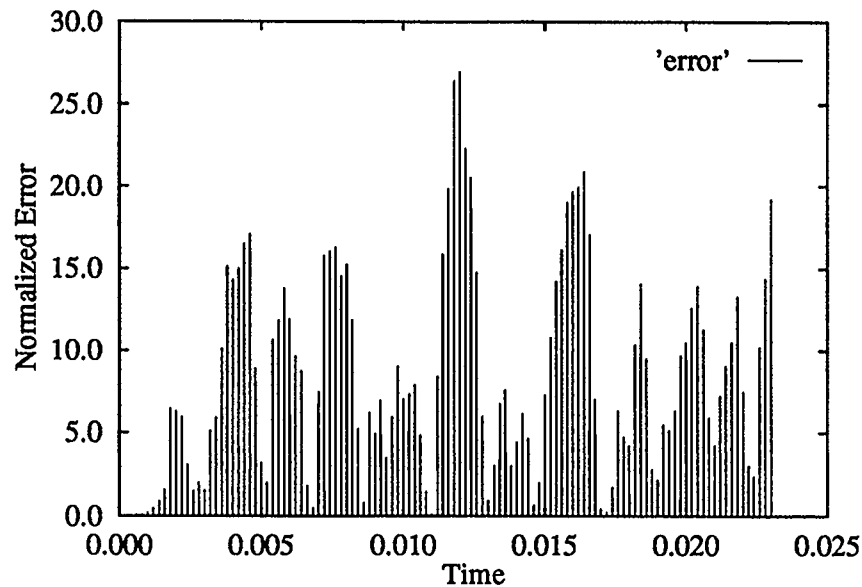


Figure 5.18: Random waveform test error.

requirements for the ROM implementation. As the number of levels  $L$  increases the number of input lines to the ROM state machine increases. The size of the ROM will grow exponentially as the number of levels is increased.

The second test input applied to the system is a Kronecker delta signal to determine the magnitude response of the filter. The Kronecker delta input signal presents problems for the difference filter. The difference modulator is unable to represent the input because the input has an infinite frequency bandwidth. When a Dencoder is presented with a Kronecker delta input, the Dencoder will saturate at its maximum representable value, since all further inputs are 0 the quantizer will properly represent them as 0. This difficulty is not unique to the difference filter. Neither delta modulators nor delta sigma modulators can represent Kronecker delta inputs. In order to allow the calculation of the magnitude response, the restriction on the maximum representable integer is lifted and all inputs are allowed. This way a Kronecker delta can be input into the system, and be large enough to provide a sufficient

number of samples for transformation into the frequency domain via the fast Fourier transform (FFT). The output of the truncated coefficient, infinite precision filter and the difference filter when stimulated with a unit Kronecker delta is shown in Figure 5.19. The difference filter output is a series of impulses separated by 85 zeros. The 85 zeros occur because the delta input occurs as an input to only one of the 86 interleaved filters. The smooth curve of points (labeled *ideal* in the graph) is the impulse response of the truncated coefficient, infinite precision filter at 5 000 Hz sample rate. The peaks of the difference filter and the curve of the infinite precision filter coincide well which indicates that the two filters will have very similar magnitude responses.

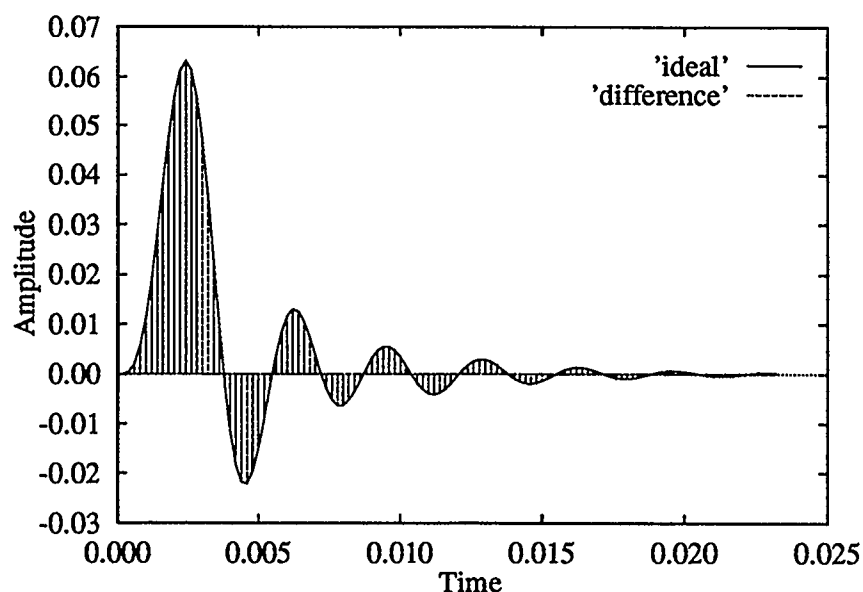


Figure 5.19: Difference filter and ideal filter response when stimulated with a unit Kronecker delta

The magnitude responses of the two filters are shown together in Figure 5.20. The frequency range is restricted to 0 to 2 500 Hz since this is the maximum input signal bandwidth to prevent aliasing. The difference filter will have a repeating magnitude response every 5 000 Hz from 0 to 430 000 Hz. From Figure 5.20, the



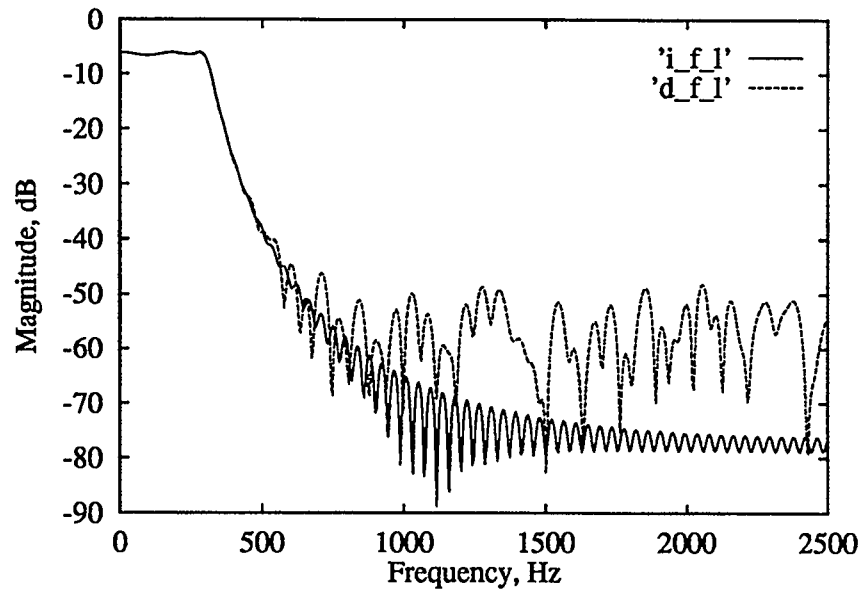


Figure 5.20: Difference filter and ideal filter magnitude response.

similarities of the magnitude response in the passband can be seen; however in the stop band the infinite precision filter has greater attenuation. The difference is due to the quantization error of the filter.

It should be noted that the magnitude response of both filters have a ripple effect in the stop band, which implies zeros. This is an effect of the FFT on the low number of sample points. The filter attenuates the delta input quickly and, thus, the analysis shows a rippling effect which is not present.

### 5.6.2 Limitations

The transfer functions that can be implemented with difference filtering are less restricted than the transfer functions in earlier work [8, 19, 21]. With adders and multipliers any filter structure can be constructed, but as discussed earlier the high error in the output of the Dadder is a serious limitation. For the Dadder to prevent accumulation of high range errors, a multiplier should lie in any data

path loop between two adders, and a multiplier of less than 1 will bound the error. Many filters can be constructed such that loops involving adders do not have a gain higher than 0.5 between adders. In the LDI structure used as an example here, all multipliers have coefficients less than 0.5.

A second limitation is due to the encoding of the input. Although a lowpass filter can be constructed, it will operate as a bandpass filter. Information on the low end of the spectrum is attenuated in the input coding. The Dencoder places a zero on  $z = 1$ , so it is effectively a highpass function. The zero is later canceled by the Ddecoder, but any information in the low end of the spectrum is removed before filtering. This effect implies that filters designed for implementation with this method should be bandpass rather than lowpass.

A third disadvantage is introduced during construction of the filter from ROM components. The interleaved filter requires a large number of small ROM blocks with long parallel shift registers. This implementation has the advantage that the large parallel paths required by the multiplier carries are localized, but wide data paths are still difficult to construct. Finally, construction requires the use of many small ROM blocks so that large numbers of interconnects are required. In discrete component implementations, a large number of interconnects is a potential source of error, while in VLSI implementations the routing will be difficult.

### 5.6.3 Memory Requirements

The implementation shown in Figure 5.16 requires 11 adders, 5 multipliers and 6 multiplies by  $-1$ . The implementation uses 12 bit words for the original wordlength, 5 bit words for the internal word length and 7 bit words for the carry in the multipliers. For an arbitrary oversampling ratio, the total amount of ROM required along

with the storage flip-flops (FFs) for the delay elements can be generalized as

$$\begin{aligned} \mathcal{M}_{total} = & N_M(b + b_c)2^{b+b_c} + N_M b_c R + N_A(b + 1)2^{2b+1} + \\ & N_A 2R + N_{-1}b2^b + bOR \quad , \end{aligned} \quad (5.126)$$

where

$$\begin{aligned} \mathcal{M}_{total} &= \text{the space required in bits} \\ N_M &= \text{the number of multipliers} \\ N_A &= \text{the number of adders} \\ N_{-1} &= \text{the number of multiplications by } -1 \\ O &= \text{the order of the filter} \\ R &= \text{the oversampling ratio of the input} \\ b &= \text{the wordlength of the signal} \\ b_c &= \text{the multiplier carry wordlength} \end{aligned}$$

The above assumes all multipliers have the same carry wordlength  $b_c$ . From Equation 5.126, the simulated filter requires 417 148 bits.

The oversampling ratio  $R$  and the wordlength of the signal  $b$  are approximately related by

$$b = \max \left( b_o - \left\lfloor \log_2(R) \right\rfloor, 1 \right) \quad , \quad (5.127)$$

where  $b_o$  is the original wordlength of the filter at  $R = 1$ .

Let us assume a 12 bit wordlength for  $R = 1$  for the 5<sup>th</sup> order LDI filter. Using both Equation 5.126 and Equation 5.127, the size of the filter can be found for various oversampling ratio (Figure 5.21). For the example filter implemented,

$$R = \frac{f_s}{2f_{max}} = \frac{86 \times 5\,000}{2 \times 500} = 430 \quad (5.128)$$

Using  $R = 430$ , the implementation requires 174 294 bits which contradicts the previously observed size of 417148 bits for the example filter. The discrepancy occurs because the internal word length found using Equation 5.127 ignores the nonlinear effects of the system.

The memory requirements shown in Figure 5.21 include the ROM look-up tables and the required FFs for storage. The minimum amount of memory required for implementation is 93 840 bits using an oversampling ratio  $R = 512$ . This minimum occurs because as the oversampling ratio  $R$  increases the amount of memory required to encode the multipliers and adders decreases, but the memory needed for the delay elements increases.

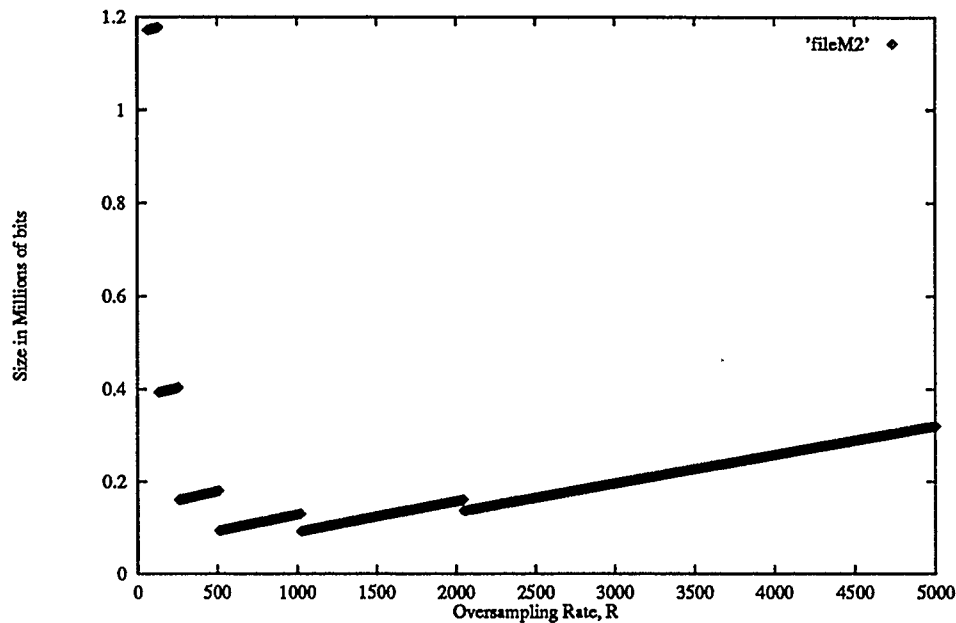


Figure 5.21: Size of ROM required as a function of the oversampling ratio .

#### 5.6.4 Implementations other than ROM

The focus of this research was to find a cost effective method for construction of filters using only ROM components. If the implementation technology is expanded beyond ROM, other interesting possibilities exist.

The difference filter could be implemented on a single chip using localized ROM blocks, where each Dadder and Dmult is constructed as a ROM state machine. The individual state machines could be connected by narrow signal bus lines. Such an

implementation would localize the wide signal busses and require only narrow buses to be routed chip wide. The localization of the wide bus lines aids in chip design, since it is much easier to route designs with small bus lines than those with many large parallel bus lines.

A second step away from ROM implementation is to avoid the use of ROM components in the construction of the state machines all together. The state machine can be constructed from combinational logic. This implementation has the advantage of more efficient construction on a chip level, while retaining the small bus widths for routing. Such a design could be implemented on a FPGA, since combinational logic is available, but routing is a difficult problem for such chips.

## Chapter 6

### Future Research Directions and Conclusions

In this thesis two new approaches to the construction of ROM based digital filters were examined along with a survey of appropriate earlier techniques involving ROM components in DSP. In this chapter, a critique of the two techniques is presented along with the results of the research conducted and directions for future research. In addition, present and future applications of the techniques are examined in light of present trends in mass storage.

#### 6.1 Technique Critique

##### 6.1.1 Elimination of Unreachable States Approach

The first new technique presented is the reduction of a block ROM state machine via the elimination of unreachable states. Unreachable states are not required in a filter realization, since the unreachable states are never encountered during operation. The removal of the unreachable states from the state space representation of the filter allows the block ROM to be reduced to

$$\frac{\pi^{\frac{1}{2}N}}{N\Gamma(\frac{1}{2}N)2^{(N-1)}} \times 100\% \quad (6.1)$$

of the original required state space volume.

This approach allows the full utilization of the memory hardware. Unreachable states are eliminated by eliminating their existence, or by remapping the state space such that only reachable points exist within the filter's map of the state space. Such a complex, nonlinear remapping is possible in ROM based devices since the ROM operates as a lookup table. The complex computations required to produce the

lookup table are performed during construction allowing complex computations to be performed during operation of the filter in one access cycle of the ROM.

The technique produces functioning digital filters which require a fraction of initial memory requirements. Unfortunately, the reduction is disappointingly small when the initial memory requirements of a filter are considered. The 5<sup>th</sup> order, 12 bit example filter is reduced from  $3.4001 \times 10^{23}$  bits for a block ROM implementation to  $5.5929 \times 10^{22}$  bits. The memory requirements for high order filters can be reduced to fractions of 1% using this technique, but the resulting total memory requirements are still very large and do not allow fabrication with present ROM based technologies.

Although the present sizes of ROM blocks do not allow fabrication of ROM based filters with this technique, mass storage devices, of which ROM is but a single example, are becoming available in ever increasing sizes. When the available storage devices reach the size of the reduced memory requirements produced by this technique, state-machine based filters can be constructed. The technique allows the construction of state-machine filters in the near rather than the far future.

One of the prime advantages of block ROM state-machine implementations is the high speed of operation. This technique allows the digital filter to operate at the access speed of the ROM. When other mass storage devices are used as the hardware for the block state-machine the filter can operate at the access speed of that storage device.

In summary, the unreachable state removal technique reduces the memory requirements while maintaining the highest possible speed of operation. Unfortunately, the reduced memory requirements remain far in excess of present technological capabilities.

### 6.1.2 Difference Signal Approach

The second method proposed in this thesis for the construction of ROM based digital filters requires the creation of a new class of signals based on difference signals. Since existing arithmetic operators do not function correctly on difference signals, new arithmetic operators for sum and multiplication operations are defined. The new signal, called a difference encoded signal, has a narrower wordlength than conventional amplitude encoded signals. It is the narrower word length which allows a reduction in the memory requirements of the state-machine arithmetic operators. The memory requirements are reduced because the numbers of input and output lines are reduced.

By applying the difference signal method, for the example filter, the memory requirements can be reduced from  $3.4001 \times 10^{23}$  bits for a block ROM implementation to 417 148 bits. This significant reduction in memory requirements is obtained at the cost of the partitioning of the ROM state-machine. Unlike the block ROM implementation where one large ROM block is required, the difference signal method requires many small ROM blocks.

Systems composed of numerous small blocks are more difficult to construct, since the individual blocks must be connected. Each connection line must be placed such that its path will not interfere with other lines. This process is called routing.

Routing tends to be a difficult task during chip lay-out as well as for circuit board lay-out. The routing difficulties are reduced in a difference signal implementation because difference signals require narrower bus lines than a similar system implemented using amplitude encoded signals. By contrast, a single block ROM state machine implementation does not encounter routing difficulties since only the input and output lines occur in such designs. The internal routing of the ROM is left as a design problem for the supplier of the ROM components.



In addition to routing difficulties of difference signal implementations, the maximum operating speed is reduced. As mentioned earlier, block ROM implementations may operate at speeds up to the access time of the ROM components. Difference signal implementations obtain the reduction in word length by highly oversampling the signal. The maximum rate at which the input signal can be sampled is the maximum access rate of the ROM components, but the maximum bandwidth of the input signal is significantly lower if any advantage is to be derived from difference signal implementations.

For telephone and audio range signals, the difference signal technique can provide significant bit reductions (12 and 8 bit reductions, respectively, for 25 MHz sampling rates) over conventional parallel implementations. As the bandwidth of the signal increases the bit reduction on the wordlength of the encoded signal is reduced exponentially with no bit reductions occurring for oversampling rates less than 2.

In summary, the difference signal technique reduces the memory requirements to present technological capabilities. To achieve this memory reduction, the speed of operation is also reduced to present technological capabilities. In addition, the technique may hold routing advantages over conventional implementations.

## 6.2 Future Research Directions

In this thesis, the primary direction of research was the reduction of memory requirements for full ROM implementation of digital filters. Future research directions could investigate alternative technologies for implementation of the two techniques presented.

Mass storage devices are not limited to ROM. Magnetic storage is often used and more esoteric forms such as holographic memories have been proposed. As research

continues on mass storage, devices three trends are likely to continue: mass storage devices will increase in size, mass storage devices will increase in speed, mass storage devices will decrease in cost.

Mass storage devices are used to store large quantities of information. As the amount of storage space in a device increases, the speed of access must increase as well. It is required that a very large memory device not only store the information, but the information must be accessible in a reasonable length of time. The exact length of time which is reasonable is determined by the application. For example, in computer data and program storage large amounts of memory are required and all locations in the memory must be accessible.

The unreachable state elimination method requires that the mass storage device have several qualities. First, the storage device must be very large. The technique reduces filter implementations to fractions of one percent of the original memory requirement, but the reduced memory requirement is still very large. Second, all memory locations in the mass storage device must be accessible at some minimum speed. Although access times may vary depending on which memory location is accessed, it is the slowest access time that will determine the maximum speed of the system. This speed will be the maximum filter sample rate. Finally, the memory must be programmable at very high speeds. When the memory requirements reach the tera-terabyte range, as for the 5<sup>th</sup> order example filter, the speed of programming the memory is important. The length of time required to program the initial configuration must be sufficiently small to allow construction.

The difference signal approach to digital filter implementation requires different characteristics of the storage media. The very large storage demands are replaced by routing demands. The difference signal technique requires many small computational blocks each of which are easily constructible under present technological constraints.

The computational blocks may be constructed from ROM, RAM, or other memory based media as look-up tables. Alternatively, combinational logic circuits can be used to implement the computational blocks. For any chosen computational engine, the difficulty in the construction of the system lies not in the memory demands of the computational block, but in the demands presented by the connections of the blocks.

Parallel buses allow signals to be transmitted within the system. These buses must be routed either on a circuit board when the system is constructed from large scale integrated components, or routed on board the integrated circuit if a full integrated design is desired. In either case, the wide parallel buses required present considerable routing difficulties.

Difference signal implementations reduce the width of the parallel bus. The routing difficulties are reduced when the width of the parallel bus is reduced. Future research could investigate the hypothesized routing advantage of difference signal systems. The length of time required to perform the routing could be compared between conventional implementations and difference signal implementations. A second evaluation of the routing advantage would compare the minimum chip area use between difference signal systems and conventional implementations.

### **6.3 Conclusions**

This thesis has presented two approaches for the implementation of ROM based digital filters. Although both approaches are inappropriate for present ROM based implementation, each approach holds promise. The unreachable state elimination method holds promise for future digital filter implementations when present mass storage trends are extrapolated. The difference signal technique holds the promise of reduced routing demands for implementation technologies other than ROM.

## Bibliography

- [1] M. Abramowitz and I. Stegun, 'Handbook of Mathematical Functions', 9<sup>th</sup> Printing, Dover Publications Inc., New York, 1970.
- [2] M. Bellanger, 'Digital Processing of Signals', 2nd Edition, John Wiley & Sons, Toronto, 1989.
- [3] L. T. Bruton, 'Low-Sensitivity Digital Ladder Filters', *IEEE Transaction on Circuits and Systems*, Vol. CAS-22, No. 3, pp. 168–176, March 1975.
- [4] M. Freedman and D. Zrilić, 'Nonlinear Arithmetic Operations on the Delta Sigma Pulse Stream', *Signal Processing*, Vol. 21, No. 1, pp. 25–35, September 1990.
- [5] M. Kendall, 'A Course in The Geometry of n Dimensions', 1st Edition, Charles Griffin & Company, London, 1961.
- [6] L. Jackson, 'Digital Filters and Signal Processing', 2nd Edition, Kluwer Academic Publishers, Boston, 1989.
- [7] W Jenkins and B. Leon, 'The use of residue number systems in the design of finite impulse response digital filters', *IEEE Transaction on Circuits and Systems*, Vol. CAS-24, No. 4, pp. 191–201, April 1977.
- [8] N. Kouvaras, 'Operations on delta modulated signals and their application in the realization of digital filters', *The Radio and Electronic Engineer*, Vol. 48, No. 9, pp. 431–438, September 1978.
- [9] N. Kouvaras, 'A special-purpose delta multiplier', *The Radio and Electronic Engineer*, Vol. 50, No. 4, pp. 156–157, April 1980.

- [21] D. Zrilić, K. Zangi, A. Mavretic and M. Freedman, 'Realization of digital filters for delta-modulated signals', *Proc. of 30th Midwest Symposium on Circuits and Systems*, pp. 1343–1346, 1988.