# CONTRIBUTIONS TO TRANSFORM CODING SYSTEM IMPLEMENTATION

by

Wenqing Jiang

---

A Dissertation Presented to the

FACULTY OF THE GRADUATE SCHOOL

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

(Electrical Engineering)

May 2000

# Abstract

With the increasing dominance of the transform coding technique virtually in every image and video coding schemes proposed up to date, efficient transform coding system implementation has become an important research topic. This thesis addresses two system issues that may arise in practice: (i) efficient architecture designs for the discrete wavelet transform; and (ii) efficient transform coding for robust communication over erasure channels.

The first contribution of the thesis is to develop an *overlap-state* technique for efficient multilevel wavelet decompositions when memory and delay constraints have to be strictly observed. In this case, the wavelet transform can be computed in a block-by-block fashion, i.e., the input data is segmented into blocks and each block is processed separately, either sequentially or in parallel. The proposed technique enables efficient data exchange between consecutive data blocks such that the required memory buffer size and/or communication overhead can be significantly reduced compared to existing techniques.

The second contribution is that we provide two efficient loss data recovery techniques for robust communication over erasure channels, such as today's Internet. For the problem of multiple description transform coding, we propose a structured correlating transform design making use of the available channel information. The enforced structure enables a significant reduction of the number of design parameters, which results in significantly reduced design and implementation complexities compared to existing approaches. An alternative technique for loss recovery using *explicit* redundancy is also proposed. This is achieved by splitting the source into different components and quantize them differently; the primary information is finely quantized at a relative high rate and the redundant information at a relative low rate. When the primary information is lost, the redundancy information can be used for recovery. The performance analysis and

simulation results show that the proposed technique is very competitive compared to previously published works.

# Acknowledgement

I would like first to express my gratitude to my advisor and friend Professor Antonio Ortega, whose continuous guidance and support kept lifting me up through all these years at USC, making it both a challenging and fun experience.

I would also like to thank Professors Alvin M. Despain, C.-C. Jay Kuo for serving in my qualifying committee and Professors Gerard G. Medioni and Zhen Zhang for serving both on my qualifying and defense committees.

During my years at USC, I had the pleasure to work with a number of friends including Hua, Krisda, Nazeeh, Phoom, Xiao-Miao, Younggap and my office-mates Raghav, Sangyong, and Woontack. I would like to thank all of them for the enjoyable collaborations we had (including sharing pizzas every week and competing for the couch in the lab).

On this Thanksgiving holiday, my love goes to my parents and my brother for their years of encouragement which kept me going. I thank my wife for sharing with me all these years, and for her love, support and faith in me.

# Contents

**Abstract** ii

**Acknowledgement** iv

**List of Tables** viii

**List of Figures** ix

**1 Introduction** 1
- 1.1 Motivation and Overview . . . 1
  - 1.1.1 DWT System Design . . . 2
  - 1.1.2 Robust Communication . . . 5
- 1.2 Contributions and Chapter Organizations . . . 6

**2 DWT Architecture Design** 9
- 2.1 Introduction . . . 9
- 2.2 Problem Definition . . . 12
  - 2.2.1 Sequential Architecture Design . . . 12
  - 2.2.2 Parallel Architecture Design . . . 13
- 2.3 Preliminaries . . . 15
  - 2.3.1 The Standard Algorithm . . . 16
  - 2.3.2 The Lifting Algorithm . . . 17
  - 2.3.3 Practical DWT System Design . . . 19
- 2.4 The Overlap-State Technique . . . 21
  - 2.4.1 The Finite State Machine Model . . . 21
  - 2.4.2 Overlap-State . . . 25
  - 2.4.3 Performance Analysis . . . 29
  - 2.4.4 Delayed Normalization . . . 31
- 2.5 The Proposed DWT Architectures . . . 36
  - 2.5.1 1D Systems . . . 36
  - 2.5.2 2D Systems . . . 39
  - 2.5.3 Sequential Architectures . . . 41
  - 2.5.4 Parallel Architectures . . . 45

2.6 Experimental Results . . . . . 47
2.6.1 Delayed Normalization . . . . . 47
2.6.2 Strip Parallel . . . . . 48
2.7 Conclusions . . . . . 50

**3 Constrained Transform Design For Multiple Description Coding 52**
3.1 Introduction . . . . . 52
3.2 Problem Definition . . . . . 56
3.3 Proposed Design Approach . . . . . 59
3.3.1 Geometric Interpretations . . . . . 59
3.3.2 Parametric Transform and Factorizations . . . . . 62
3.3.3 Two Stage Transform Design . . . . . 64
3.4 MDTC Design Examples . . . . . 66
3.4.1 Equal Rate Channels . . . . . 66
3.4.2 Sequential Protection Channels . . . . . 68
3.4.3 Simulation Results of Gaussian Sources . . . . . 68
3.5 Karhunen-Loeve Vector Transform . . . . . 70
3.6 Conclusions . . . . . 73

**4 Multiple Description Coding for Erasure Channels 74**
4.1 Introduction . . . . . 74
4.2 Erasure Channel Model and Problem Definition . . . . . 77
4.3 MDC Based On Explicit Redundancy . . . . . 78
4.3.1 Base System . . . . . 78
4.3.2 Context-Adaptive Extension . . . . . 79
4.4 Optimal Redundancy Bit Allocation for Gaussian Sources . . . . . 80
4.4.1 Implicit Channel Modeling . . . . . 81
4.4.2 Explicit Channel Modeling . . . . . 85
4.5 Experimental Results . . . . . 86
4.5.1 An Image MDC Example . . . . . 86
4.5.2 A Speech MDC Example . . . . . 91
4.6 Summary and Future Work . . . . . 95

**A Overlap-Add and Overlap-Save 96**

**B DWT FSM Examples 98**

**C MDTC Transform Design Algorithm 103**
C.0.1 Redundancy bit rate $\rho$ . . . . . 104
C.0.2 Side Distortion $D_s$ . . . . . 104

**D KLVT Vector Space Partition 107**

**Bibliography** . . . . . **109**

# List of Tables


2.1 The standard algorithm. . . . . 16
2.2 Costs comparison for multilevel wavelet decompositions ($J > 3$) . 34
2.3 The proposed sequential DWT algorithm. . . . . 37
2.4 Overlap buffer size $B_s$ in 1D DWT for $J$-level decompositions using a $L$-tap wavelet filterbank. . . . . 37
2.5 The proposed parallel DWT algorithm. . . . . 39
2.6 Comparison of memory requirements where $W$ is the width of the image scanline and $\alpha = (2^J - 1), \beta = (1 - 2^{-J})$ . . . . . 43
2.7 Comparison of memory requirements where $\alpha = (2^J - 1), \beta = (1 - 2^{-J})$ 45
2.8 DWT CPU time of different sequential algorithms (in seconds). . 48
2.9 The proposed parallel DWT algorithm. . . . . 50
2.10 DWT running time of different parallel algorithms (in seconds). . 51

4.1 "Squrriel" reconstruction NMR comparison (dB) . . . . . 93
4.2 "Draw" reconstruction NMR comparison (dB) . . . . . 95

# List of Figures


1.1 A simplified image/video coding system. . . . . 1
1.2 A three-level tree decomposition. . . . . 4

2.1 An example dataflow chart of a three-level wavelet decomposition. Solid lines: completely transformed data; Dashed lines: boundary samples from the neighboring block. Operations 1,3,5: communicate boundary data samples to neighboring blocks; Operations 2,4,6: transform for current level. . . . . 10
2.2 A typical sequential DWT system diagram. . . . . 12
2.3 (a) 2D mesh processor network; (b) The corresponding data partition (b). . . . . 14
2.4 (a)Bus-connected processor network; (b)The corresponding strip data partition. . . . . 15
2.5 Two-channel wavelet filterbank in polyphase representation. . . . 17
2.6 Wavelet transform via lifting. (a) Forward transform. (b) Inverse transform. . . . . 18
2.7 Boundary processing for DWT. When filter (length L) moves to the right boundary of block 1, it needs input data samples from block 2. In a sequential architecture, block 1 and 2 do not reside in memory at the same time. In a parallel architecture, block 1 and 2 are allocated to different processors. . . . . 19
2.8 State transition diagram of DWT as a FSM. . . . . 24
2.9 State transitions across block boundary using $\mathbf{e}^i$ with $a^i = 3$ and $b^i = 2$. (a) Partial computations near boundaries. (b) After updating, boundary samples stay in intermediate states. The "new boundary" separates fully updated output coefficients from partially computed ones. . . . . 26

2.10 Sequential DWT using *overlap-state*. (a) Input in initial state $i$. (b) Block 1 consists samples up to $x^i(2m)$. After state transition, samples near block boundary are only partially updated (anti-causal filtering results not available). (c) Partially updated samples (state information) are overlapped with next block of input samples. They are now completely updated by adding their anti-causal filtering results. (d) Completely transformed (updated) input from state $i$ to $i+1$. . . . . 28
2.11 Parallel DWT using *overlap-state*. (a) Input in initial state $i$. (b) Input is partitioned over two processors. Block 1 and 2 are transformed separately and each have state information appearing near the block boundary. (c) State information is exchanged between two processors and the partially updated samples are fully updated. (d) Completely transformed (updated) input from state $i$ to $i+1$. 29
2.12 An example dataflow chart of a three-level wavelet decomposition using the proposed *Overlap-State* technique. Solid lines: completely transformed data; Dashed lines: partially transformed data. Operation 1: each block transforms its own allocated data independently and state information is buffered; Operation 2: state information is communicated to neighboring blocks; Operation 3: complete transform for the boundary data samples. . . . . 31
2.13 Illustration of delayed normalization. (a)Recursive two-channel lifting. (b)Two channel lifting with delayed normalization. . . . . 32
2.14 A 2D DWT example of delayed normalization. . . . . 33
2.15 Joint design of transform and quantization to reduce computation. (a) Independent transform and quantization. (b) Joint transform and quantization. The normalization operation is merged with the quantization. . . . . 35
2.16 The proposed sequential DWT architecture. . . . . 36
2.17 The proposed parallel DWT architecture. In *Split* stage, each processor computes its allocated data independently up to the required decomposition level. In *Merge* stage, a one-way communication is initiated to communicate the state information to neighboring processors. A postprocessing operation is then started to complete the transform for boundary samples. . . . . 38
2.18 2D DWT/FSM illustration. Shaded areas represent the state information with $R0/C0$ the row/column state information at level 0 and so on. The input block is first row transformed (a) and column transformed (b) at level 0, then downsampled (taking the LL0 subband) and row transformed at level 1 (c), and column transformed at level 1 (d). . . . . 40

2.19 Strip sequential DWT system diagram. The input is segmented into data strips which are transformed sequentially from top to bottom. . . . . 42
2.20 Block sequential DWT system diagram. The input is segmented into blocks which are transformed from left to right and from top to bottom. . . . . 44
2.21 *Merge* operations in 2D mesh processor network. ($a$)transfer row state information from $P_{i,j}$ to $P_{i,j+1}$; ($b$)transfer column state information from $P_{i,j}$ to $P_{i+1,j}$; ($c$)transfer newly generated row state information from $P_{i,j}$ to $P_{i,j+1}$; ($d$)complete transform for boundary samples. Notice the total amount of data in each processor in the final state ($d$) is different from the original uniform allocation due to the *Merge* operations. . . . . 46
2.22 *Merge* operations for strip parallel implementation. (a)transfer row state information from $P_i$ to $P_{i+1}$; and (b)complete transforms for boundary samples in each processor. . . . . 47

3.1 A typical MDTC system . . . . . 54
3.2 Redundancy rate distortion curve of a MDTC system for Gaussian vector with $\sigma_1 = 1, \sigma_2 = 0.5$. . . . . 58
3.3 Distributions of two independent Gaussian variables with variance $\sigma_1 = 1, \sigma_2 = 0.5$. (a) original; (b) after 45° rotation with correlation coefficient 0.6. (c) after scaling($a = 2$); (d) after scaling and 45° rotation with correlation coefficient 0.94. Clearly one can see that data in (d) is more correlated than that in (b). . . . . 61
3.4 Lattice structure of a **RS** transform . . . . . 63
3.5 Transform search space. . . . . 66
3.6 Comparisons among different transforms. . . . . 69

4.1 The proposed MDC system. . . . . 79
4.2 Context-based MDC System . . . . . 80
4.3 Rate-distortion performances comparison for a Gaussian source $\mathcal{N}(0,1)$: (1) Ozarow: optimal bound. (2) MDSQ: optimal level-constrained results. (3) Proposed: Lloyd-Max quantizer results with fixed length code. (4) Optimal: results using the rate-distortion function of the Gaussian source. . . . . 84
4.4 An example configuration of two different polyphase transforms using a two-level wavelet decomposition on a 16x16 input matrix. In all the subbands, all the 1 samples constitute one polyphase component and the remaining samples constitute the other polyphase component. (a) plain polyphase transform. (b) vector polyphase transform. . . . . 87

4.5 Experimental results with Lena gray-level image. (a) Two descriptions. Polyphase (1): plain polyphase transform. Polyphase (2): vector polyphase transform. (b) Performances with independent packet losses. . . . . 89
4.6 Reconstructed Lena images at total rate 0.5*bps* using only one channel information. (a) Original image; (b) Redundancy bit rate 0.05*bps*, PSNR 29.64dB; (c) Redundancy bit rate 0.10*bps*, PSNR 31.91dB; (d) Redundancy bit rate 0.15*bps*, PSNR 32.98dB . . . . 90
4.7 The proposed and the RAT schemes for robust packet speech coding. 92
4.8 Reconstructed NMR distribution plot for the *Draw* sentence under different packet loss probabilities. RAT: o; Proposed: +; JAY: x . 94

A.1 (a) Overlap-save. (b) Overlap-add. . . . . 97

B.1 State transitions of the Daubechies (9,7) filters using factorization (B.1). The state information consists of 4 samples (the overlap buffer size) in shaded boxes. Dashed lines represent operations necessary for the transform of the new input sample pair $\{x^0(9), x^0(10)\}$. . . . . 99
B.2 State transitions of the (2,10) filters using factorization (B.1). The state information consists of 4 samples in shaded boxes. Dashed lines represent operations necessary for the transform of the new input sample pair $\{x^0(10), x^0(11)\}$. . . . . 101
B.3 State transitions of the CDF (4,2) filters using factorization (B.4). The state information consists of 3 samples in shaded boxes. Dashed lines represent operations necessary for the transform of the new input sample pair $\{x^0(8), x^0(9)\}$. . . . . 102

# Chapter 1

# Introduction

## 1.1 Motivation and Overview

To motivate our work in this thesis, let us first consider a simplified communication system shown in Fig. 1.1. The coding and decoding procedures can be described as follows. First, the redundancy in the data is removed by applying a decorrelating transform, such as the Karhunen-Loeve transform (KLT) or its approximations (e.g., the discrete cosine transform (DCT) and the discrete wavelet transform (DWT)). Next, the transform coefficients are quantized and entropy-coded to meet the bit budget constraint. The encoded bitstream is then sent over the channel for transmission. At the receiver, an inverse procedure is applied to reconstruct the original data.

While such a coding framework constitutes the core part of virtually every existing image/video coding standards, e.g., JPEGx, MPEGx (MPEG1, MPEG2

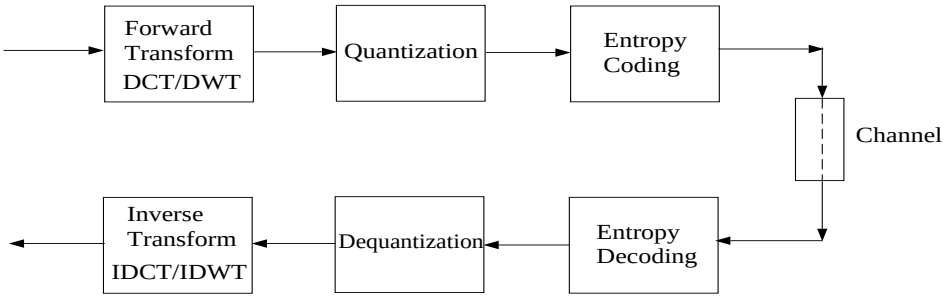



Figure 1.1: A simplified image/video coding system.

and MPEG4, etc..) and H.26x (H.261, H.263, and H.263+, etc..), these standards themselves, however, are designed to target different data compression applications. As a result, practical system designs and implementations have to be subjected to different constraints.

For example, the JPEG standard is mainly used for compression of still images, such as pictures in digital cameras, printers and scanners. One important issue here is to reduce the algorithm memory usage in these consumer electronics products to reduce the price [1, 2, 3]. That is, the transform, quantization and entropy coding blocks in Fig. 1.1 have to be implemented using as less memory as possible. The MPEGx and H.26x standards, on the other hand, are for compression of moving pictures, such as movie and television video signals. In this case, fast algorithms, including fast transform, fast quantization and fast entropy coding, become more important since most applications have very stringent time constraints and some even require real-time encoding and/or decoding, e.g., video-conferencing and live concerts digital broadcasting. As one can see, it is thus very important to design and implement efficient systems under various application constraints.

With the fast development of the Internet, more and more multimedia signals (e.g., image, video and audio signals) are being communicated over the packet network. However, most internet service providers can only deliver best-effort services using current available network technologies, i.e., packets transmitted may get lost before arrival at the destination. This can happen when the network congestion occurs or link failures at intermediate routers. To achieve robust communication over such unreliable channels thus become another important topic.

In this thesis work, two practical issues are addressed: (i) Efficient DWT system design under memory and delay constraints; and (ii) Robust communication over packet erasure channels. We now give a detailed review on each of these two topics.

### 1.1.1 DWT System Design

In recent years, there have been considerable research activities centered around building efficient systems for computing the discrete wavelet transform (DWT)

[4, 5, 6, 7, 8, 9, 10, 11, 1, 12, 13, 14, 15, 16, 17]. This is certainly due in part to the fact that the wavelet transform is a powerful tool for multiscale time-frequency signal decomposition and analysis, which has found applications in many areas, such as signal processing, digital communications, numerical analysis, and computer graphics [18]. Moreover, practical system design is itself a very challenging problem because there may be stringent constraints, such as buffer size, delay, power, chip area and control complexity, imposed by specific DWT applications [7, 11, 19, 20, 21, 8, 13, 22, 1, 2, 14, 17].

In many cases, a sequential architecture is used where the DWT is computed by splitting the input into blocks, with the processor operating on one block at a time [6, 1, 2]. One reason for such a choice is that only a limited amount of memory is available for the transform computation. Example scenarios include image compression/decompression systems using a DSP/ASIC chip in products of consumer electronics products (e.g., digital cameras) or space-borne instruments [1, 2, 23]. In these applications, reducing the memory buffer size helps not only to maintain low costs but also to reduce the chip design area and thus the volume of the final product.

As an alternative a parallel architecture would split the input among several processors to speed up the transform computation [11, 24, 9, 22, 17]. This is typical for applications where a large volume of data has to be processed in a reasonably short time. For instance, the seismic data processing [22] or illumination computations in computer graphics [18] are potential applications. Obviously, fast DWT computation to meet stringent delay constraints is critical to the success of any wavelet-based techniques.

While the problem of system design under constraints, such as memory and delay, is not new and is also encountered in the design for traditional transforms (e.g., FFT or DCT), system design for the wavelet transform poses particular difficulties not present before. Consider an example three-level wavelet decomposition as depicted in Fig. 1.2 where $(h, g)$ are respectively the lowpass and highpass filters. The input is first filtered by $h$ and $g$. The lowpass output is then downsampled and filtered again. For multilevel decompositions, this process of filtering and downsampling operations has to be performed *recursively* on the

input data. Since filtering operations in DWT can not be implemented as a block transform (with the exception of the trivial Haar transform), this *recursiveness* nature of the DWT computation poses special challenges when stringent memory and delay constraints have to be observed for practical DWT system designs.

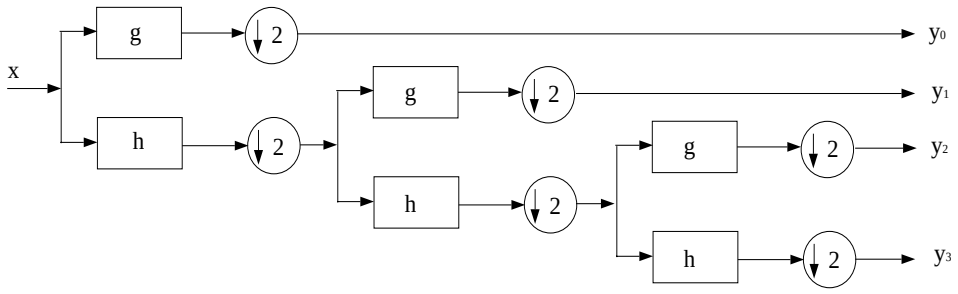



Figure 1.2: A three-level tree decomposition.

Consider, for example, this three level wavelet decomposition is to be performed using two processors with each processor allocated half of the input data. A problem arises when DWT is computed near data boundaries at both processors (refer also to Fig. 2.7). Because DWT is not a block transform, for correct wavelet coefficients computation near data boundaries, each processor would need to access data allocated to the other processor. In this case, either the two processors exchange data before each level of the decomposition or the two processors are given sufficient overlapped data to carry on the whole computation without communication with each other. The first approach demands frequent data exchanges between processors which increases the communication overhead thus adversely affect the system scalability in parallel architectures, particularly these with slow communication links, for example, Network Of Workstations (NOWs) or Local Area Multicomputers (LAMs) [25, 16, 26, 27]. The second approach, although avoiding frequent communication, needs to overlap data at each processor. This overlap, due to the *recursive* nature of the DWT filtering operations, can be very large hence very expensive when the number of levels of decomposition increases [28].

This provides the motivation to investigate efficient DWT system design under memory and delay constraints. While there are many factors which could possibly affect the memory and delay for the DWT computation (see, e.g., the

memory requirement for the complete DWT compression/decompression systems [1, 2] and the interprocessor communication required by data transposition in 2D DWT [17]), in this work, we focus on the memory and the interprocessor communication constraints imposed by the segmentation of the input data, either for sequential or for parallel architecture designs as demonstrated by the above example. Consequently, the two parameters we use to measure the performance would be the amount of data to be transmitted between processors (or to be stored in the processor if a sequential computation is used) and the number of times data has to be communicated between processors.

### 1.1.2 Robust Communication

The second part of this thesis is devoted to the study of robust communication techniques for delay-constrained applications over erasure channels. Examples of applications in which such techniques become crucial include videoconferencing, realtime audio and speech over packet networks. The best-effort service model, as currently being implemented by most internet service providers (ISPs), does not guarantee timely lossless packet delivery. Packets can be dropped or over-delayed in a number of scenarios. For example, packets can be over-delayed in congested network segments or even dropped if packet dropping policies (e.g., random early drop (RED)) are implemented to relieve the congestion. Packets can also be corrupted thus becoming useless at the receiver when sent over hostile channels, e.g., a mobile radio channel suffering from severe multipath fading. As observed by a number of works recently [29, 30, 31], packet losses, if not dealt with appropriately, can cause very annoying quality changes in the received signal.

Numerous research efforts have been aiming at providing quality-of-service (QoS) by redesigning the network infrastructure (e.g., RSVP [32]) thus providing bounds on packet losses or avoiding losses altogether. However, in this work, we study techniques to enable recovery from packet losses and to mitigate the losses in signal quality due to the underlying erasure channel. Our motivation is two-fold: (i) the applications we study (e.g., image, video and audio communications) do not require lossless data recovery, i.e., some degradation can be tolerated as long as the degradation is below a certain threshold; and (ii) these techniques can

complement QoS-based transmission (especially if packet losses still occur even if they are bounded), or at least serve as near term solutions before the wide deployment of QoS networks. Our goal is then to design techniques to enable the signal quality to degrade gracefully in the presence of packet losses.

The problem of robust communication over erasure channels is not new and has been studied thoroughly in the context of channel coding theory [33] and data communication protocols [34]. By using error correcting codes (e.g., the block and convolutional codes), one can increase the information redundancy (thus resulting in bandwidth inefficiency) to achieve more robust communication [35, 36, 37]. By using retransmission mechanisms (e.g., the ARQ schemes) one can also achieve robust communication at the expense of an increase in communication delay [38, 39]. These existing techniques, though quite successful in the past, have to be applied with caution for communications over the packet network because of the differences of the underlying channel models. For example, packet losses occur more often due to network congestion rather than because of bit errors due to channel noise, as assumed in many conventional channels. In this case, immediate retransmission attempts may even aggravate the situation and lead to more packet losses. The effectiveness of retransmission also decreases in multicast applications when end users suffer from uncorrelated losses. In this case, an appropriate choice would be to use forward-error-control (FEC) schemes. However, end users may experience different levels of packet loss due to the network heterogeneity and a scheme that provides single level error protection can be insufficient for some users but redundant for others. These examples show that robust communication over the lossy packet network deserves further study.

## 1.2 Contributions and Chapter Organizations

In the next chapter, we present our work on DWT system architecture design. We start with an overview on previous works on DWT computations, including algorithm studies and architecture designs. Then a *finite state machine* (FSM) model is proposed to characterize the DWT behavior especially around the data boundaries. Based on this DWT/FSM model, an *overlap-state* technique is proposed to

correctly transform the boundary samples in a multilevel wavelet decomposition with reduced interprocessor communication and memory usage. We then present various sequential and parallel DWT system designs using the proposed technique and show how transform buffer size and communication overhead can be reduced.

Our main contribution in this part is to provide a new technique, *overlap-state*, to compute multilevel wavelet decompositions in a way that can significantly reduce the memory and communication overhead compared to existing technologies. We believe that this will greatly help practical DWT system designs where memory and delay constraints have to be strictly observed.

The second part of our thesis is on the study of techniques that can achieve robust communication over packet erasure channels, such as today's Internet. More specifically, we study techniques that can help the signal quality to degrade gracefully in case of packet losses.

Two different approaches are taken in our study. In chapter 3, a constrained correlating transform design for multiple description coding is presented. This correlating transform is used to add redundancy in the encoded data such that lost transform coefficients can be estimated (thus recovered) using the correlation existing in correctly received coefficients. Starting with an review of related works in multiple description transform coding (MDTC), we provide an detailed analysis on why non-orthogonal correlating transforms can perform better than orthogonal correlating transforms for MDTC system design. We then address the difficulties of existing MDTC system designs in which arbitrary non-orthogonal transforms are used. A two-stage transform design approach, structure design and magnitude design, is then presented to drastically reduce both the design and implementation complexities. We also provide design examples using the proposed technique for the design of transforms for equal rate and sequential protection channels.

Chapter 4 is devoted to a different technique for packet loss recovery. Rather than using a correlating transform to *implicitly* add redundancy into the data for loss recovery, redundancy is *explicitly* added by source splitting and selective quantization. The source splitting is performed using the polyphase transform as an example though other forms are also possible. The selective quantization is achieved by quantizing the primary information and redundant information at

different resolutions. The performance analysis of our proposed system is provided in detail for Gaussian sources and comparisons with existing approaches are also provided. Experimental results of image and speech coding and communication over independent packet loss channels are also provided.

The main contribution in this part is that we propose simple (in design and implementation) yet efficient (competitive coding performances compared to previously reported works) techniques for adding redundancy into encoded data for loss recovery. We believe that this is important for practical system implementations when robust communication is required over unreliable channels.

# Chapter 2

# DWT Architecture Design

In this chapter, we study efficient DWT architecture designs under memory and delay constraints [1].

## 2.1 Introduction

Various investigations of efficient wavelet transforms implementation have been reported in recent years. The most popular DWT algorithm is the recursive filtering approach using the corresponding wavelet filterbank, the so-called standard algorithm [44], whose computational complexity is $O(L)$ per output coefficient ($L$ is the filter length). The FFT-based DWT algorithm proposed by Rioul et al. [5] can reduce the complexity from $O(L)$ to $O(\log L)$ for large filter lengths $L$. For short filters, they presented a "fast running FIR filtering" technique which can achieve 30% saving in computations. Using a lattice structure, Vaidyanathan et al. [45, 46] have shown that the complexity can be reduced by a factor of 50% for orthogonal wavelet filterbanks. The ladder structure by Marshall [47] and the lifting algorithm by Daubechies and Sweldens [48] further show that, asymptotically, for large filter lengths $L$, 50% savings in computations can be achieved for any FIR wavelet filterbanks including orthogonal and biorthogonal filterbanks.

In sequential architecture designs, most approaches adopt the standard FFT-based filtering techniques [49], *overlap-add* or *overlap-save*. These include the

[1] Part of this chapter represents work published before, see [40, 41, 42, 43].

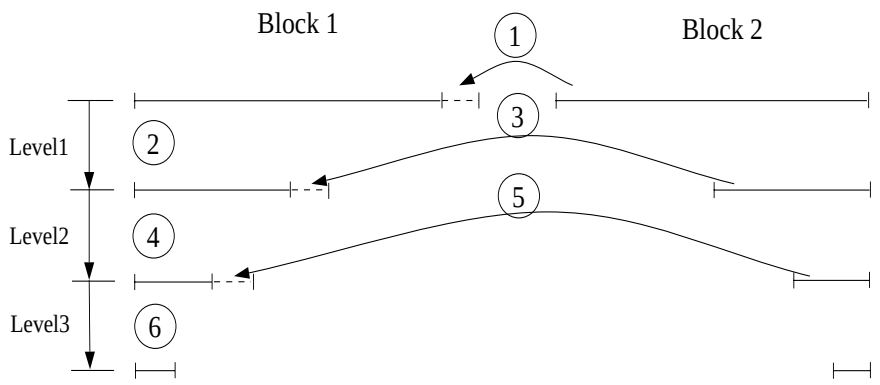



Figure 2.1: An example dataflow chart of a three-level wavelet decomposition. Solid lines: completely transformed data; Dashed lines: boundary samples from the neighboring block. Operations 1,3,5: communicate boundary data samples to neighboring blocks; Operations 2,4,6: transform for current level.

recursive pyramid algorithm (RPA) by Vishwanath [6], the spatially segmented wavelet transform (SSWT) by Kossentini [28], and the reduced line-based compression system by Chrysafis et al. [1]. Since the SSWT overlaps data only once before the start of the transform, the overlap buffer size increases exponentially with the increase of decomposition levels. An alternative is implemented in [6, 1] where data is overlapped at each level of decomposition and the buffer size is reduced. In parallel architecture designs, most approaches proposed require communication of the boundary data at each level of decomposition (see, for example, the works by Fridman et al. [11] and by Nielsen et al. [24]). One such design with three level decompositions is shown in Fig. 2.1. To reduce the overhead caused by frequent inter-processor communication, Yang at el. [25] proposed to use boundary extensions in their DWT system configured from a cluster of SGI workstations. This, however, computes incorrect wavelet coefficients near data boundaries, which causes performance degradation in some applications, for example, low-bit rate image coding [50].

In this chapter, we present a novel technique, *overlap-state*, for the DWT computation, which can help to achieve significant memory and communication savings. The idea is motivated by the standard *overlap-add* technique which performs filtering operations on neighboring data blocks independently *first* and completes the computation *later* by summing the partial boundary results together

[49]. We extend this idea to the case of multilevel wavelet decompositions using the lifting framework formulated by Daubechies and Sweldens [48]. The DWT is first modeled as a *finite state machine (FSM)* using the lifting algorithm and multilevel partial computations (intermediate states) are performed for samples near block boundaries. We show that, for correct transform near data boundaries, these intermediate states can be preserved in their original storage spaces (an extension of the in-place computation feature of the lifting algorithm) and exchanged only once between neighboring data blocks for any arbitrary $J$ level decompositions.

Some recent works have also explored (independently of our work) the use of lifting factorizations for memory savings in DWT implementations [51, 52, 53, 54]. The novelty of our work is that, first, we introduce partial computations for boundary samples at multiple decomposition levels and preserve these partially computed results (intermediate states) in their original locations for later processing, and second, we propose that processors exchange data after multilevel decompositions rather than at each decomposition level. We will show how the *overlap-state* technique can be used to reduce the memory requirement and the interprocessor communication overhead in the sequential and parallel architecture designs.

This chapter is organized as follows. In the next section, we define the problem of memory and delay constrained DWT system design. respectively for sequential and parallel architectures. In Section 2.3 an overview of various DWT algorithms, including the standard DWT algorithm and the lifting algorithm, is provided. Difficulties of computing DWT near data boundaries are detailed from the memory and delay perspective. Section 2.4 then presents the *overlap-state* technique based on the idea of partial computation at multiple levels in the process of DWT computation. It is shown that the proposed technique can help to significantly reduce the memory and communication need in practical system designs. A *delayed normalization* technique is also introduced to speedup multilevel DWT computations. In Section 2.5 examples DWT system designs for 1D and 2D data are provided in detail. Experimental results and anaylsis are given in Section 2.6. Finally, Section 2.7 concludes this chapter.

## 2.2 Problem Definition

In this Section, we define the problem of sequential and parallel architecture designs using the 2D separable DWT as an example. Other types of DWT system can be defined similarly.

### 2.2.1 Sequential Architecture Design

A sequential system for 2D DWT is shown in Fig. 2.2. The transform working buffer (e.g., on-chip memory or cache memory) usually is small in size compared to the data size. Therefore the original data, stored in a secondary storage space (e.g., hard disk, frame buffer), therefore has to be segmented such that each segment can be loaded to the working buffer and the transform is computed one segment at a time. Variations of this generic system include:

1. The block-based system presented in SSWT by Kossentini [28] which computes the wavelet transform one image block at a time.

2. The line-based system presented by Chrysafis et al [1, 55] which computes the wavelet transform "on the fly" and where the basic input units are image lines.

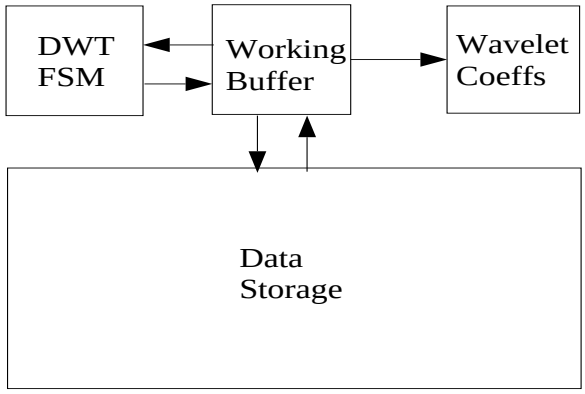



Figure 2.2: A typical sequential DWT system diagram.

Assume blocks of size $B$ (in bytes) are given to the processor, i.e., at most $B$ bytes of input data can be loaded into the working buffer at a time. Define

the *overlap buffer size* $B_s$ in bytes as the memory space taken by the overlapped data which has to be kept in memory for the correct transform of next block of input data. In the case of the line-based systems [1], $B_s$ is the minimum buffer size needed for the transform. After transform of each block, only $(B - B_s)$ bytes can be freed and wavelet coefficients generated can be transfered to the next processing stage, e.g. quantization. We now define the system throughput $\eta$ as

$$\eta = \frac{B - B_s}{B} = 1 - \frac{B_s}{B} \tag{2.1}$$

Intuitive explanations of $\eta$ are as follows. If $\eta = 1$, this indicates that all of the original data samples can be fully transformed which corresponds to the case of pure block transforms, such as DCT or the Haar transform. If, using the whole buffer, no complete decomposition can be performed (i.e., data is not enough for $J$-level of decompositions), then $\eta = 0$. We mention, however, that in this case it is possible that some of the wavelet coefficients in high frequency bands can be generated.

The problem is formulated as: *Given a fixed working buffer size $B$, how to compute the DWT to maximize the system throughput $\eta$?* Obviously, to increase the system throughput, one has to reduce the *overlap buffer size* $B_s$ as much as possible.

### 2.2.2 Parallel Architecture Design

**Mesh Processor Network** We first consider a 2D mesh-connected processor network depicted in Fig. 2.3(a) where each processor is only connected with its immediately neighboring processors (a similar model is also studied in [11]). Communications between processors can be according to the *single port* model, in which processors may only send or receive a message over one communication link at a time, or the *multi-port* model, in which a processor may send or receive multiple messages over multiple links at a time.

Using such a model, the natural partition for 2D data is the block partition strategy shown in Fig. 2.3(b). The processor $P_{m,n}$ is allocated with input samples with indices $(x, y), mN_r \leq x \leq (m+1)N_r, mN_c \leq y \leq (m+1)N_c$. Without loss

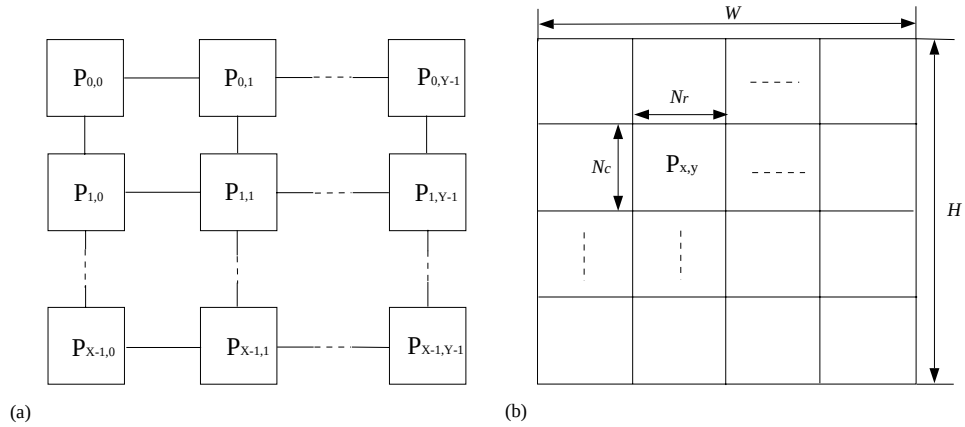



Figure 2.3: (a) 2D mesh processor network; (b) The corresponding data partition (b).

of generality, assume $W = MN_r$ and $H = NN_c$ where $(N_r, N_c)$ are the block row and column length, and $(M, N)$ are the number of processors in row and column direction respectively.

**Bus Processor Network** We also consider another type of processor network in which each processor can communicate to every other processor through a common bus. An example is the LAM (Local Area Multicomputer) systems, see Fig. 2.4(a), where locally connected machines are reconfigured into a parallel system (a similar model can also be found in [22]). One possible data partitioning approach is the strip partition which is depicted in Fig. 2.4(b) where processor $P_n$ is allocated with input samples of indices $(x, y), 0 \leq x \leq W - 1, nN_c \leq y \leq (n+1)N_c$.

The message passing mechanisms in both processor networks are modeled as follows. The communication time $T_c$ for a size-$m$ message is

$$T_c = t_s + mt_w + t_p$$

where $t_s$ is the time it takes to establish a connection. $t_p$ is the propagation time, and $t_w$ is the time to transmit a size-1 message. If one message unit is an integer, then $t_w$ is the time to transmit one integer. Other cases are defined similarly. Notice that for the bus processor network, $t_p$ is taken as the average propagation time and, for the mesh processor network, $t_p = lt_h$ where $l$ is the number of links

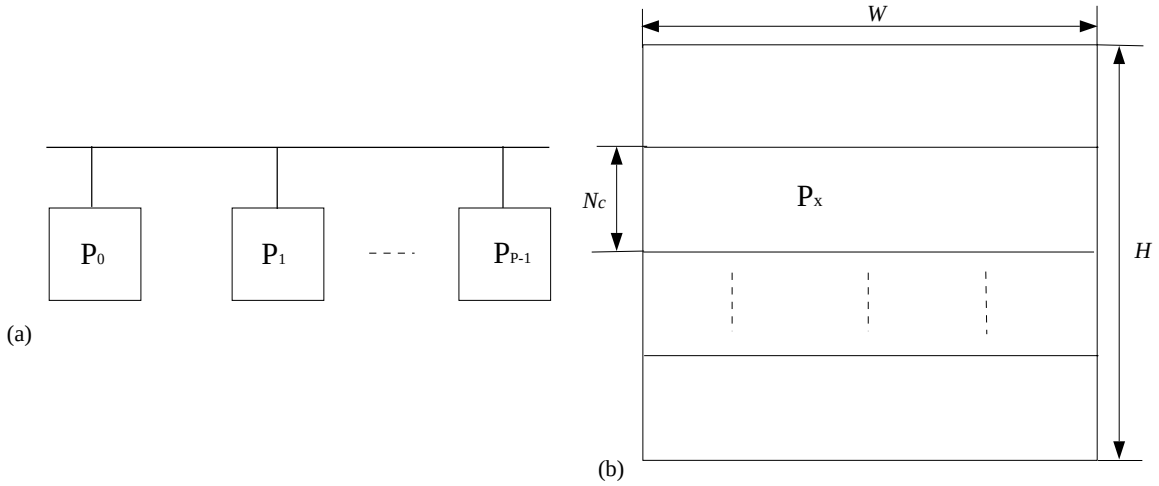



Figure 2.4: (a)Bus-connected processor network; (b)The corresponding strip data partition.

and $t_h$ is the propagation time over one link.

The design problem is formulated as: *Given the communication model as defined above, minimize the communication overhead in a parallel DWT system.* To this end, clearly we can reduce the overhead by reducing the number of communications and/or reducing the amount of data that has to be exchanged.

## 2.3 Preliminaries

To lay the foundation for our proposed architecture designs, we first give a review on DWT algorithms including the standard and the lifting algorithms. The difficulties of applying traditional techniques to meet the system design constraints (memory and delay as outlined in the previous section) are then explained in detail.

We mention that, throughout this chapter, we focus on the tree-structured e [44] multilevel octave-band wavelet decomposition system with critical sampling using a two-channel wavelet filterbank. The extensions of our study to systems of *standard* DWTs [56], multichannel wavelet filterbanks, and wavelet packet decompositions are straightforward.

### 2.3.1 The Standard Algorithm

Theoretically [57], the wavelet transform is a signal decomposition technique which projects an input signal onto a multiscale space constructed from the dilated and translated versions of a prototype wavelet function, i.e., the *mother* wavelet. Computationally most wavelet transforms can be implemented as recursive filtering operations using the corresponding wavelet filterbank as shown in Fig. 1.2. This implementation will be refered to as the *standard* algorithm hereafter. We emphasize that the filtering operations are only performed every other sample for two-channel filterbanks, i.e. a subsample-filtering approach, which is already a fast algorithm [5]. The pseudo-code implementation is given in Table. 2.3.1.

Table 2.1: The standard algorithm.

```
input: x[n], n ∈ [0, N]
       N: input sequence length
       J: decomposition level
       L: filter length
output: y[k], k ∈ [0, N]
begin
    for (j = 0; j < J; j++)
    for (n = 0; n < 2^(J-j); n++)
        {
          x^j[n] = Σ_{m=0}^{L-1} x^{j-1}[m] h[2n - m]
          y^j[n] = Σ_{m=0}^{L-1} x^{j-1}[m] g[2n - m]
        }
end
```

For practical applications with memory and delay constraints, the standard algorithm, however, may not be a good choice for three reasons: (i) it requires a buffer of same size as the input to store the intermediate results (the lowest subband ) for recursive filtering operations; (ii) it has a large latency since all the outputs of one subband are generated before the output of the next subband; and (iii) the computation cost is high. Define *algorithm computation cost* as the number of multiplications and additions per output point. Using wavelet filters with $L$-taps, $L$ multiplications and $(L-1)$ additions are needed for one output

point at each level. The cost $C_S$ of the standard algorithm, for a $J$-level wavelet decomposition, can be computed as [5]

$$C_S^J = (L + L - 1)(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{J-1}}) \tag{2.2}$$

$$= 2(2L - 1)(1 - 2^{-J}) \tag{2.3}$$

### 2.3.2 The Lifting Algorithm

A size-$N$ *polyphase transform* [57] of a signal $x[n]$ is defined as a mapping which generates $N$ subsequences with each being a shifted and downsampled version of $x[n]$. i.e., $x_i[n] = x[nN + i]$. These subsequences are called the *polyphase components* of signal $x[n]$. In the case of $N = 2$, this transform simply divides the input sequence into two polyphase components which consist of samples with odd indices and samples with even indices, respectively. In $z$-transform domain, the polyphase representation of $x[n]$ is $X(z) = \sum_{i=0}^{N-1} z^{-i} X_i(z^N)$.

Define the polyphase matrix $\mathbf{P}(z)$ as

$$\mathbf{P}(z) = \begin{bmatrix} H_0(z) & H_1(z) \\ G_0(z) & G_1(z) \end{bmatrix} \tag{2.4}$$

where $H_i(z)$ is the $i$th polyphase component of the $H(z)$ (similarly defined for $G(z)$). The DWT in polyphase domain can be written as

$$\begin{bmatrix} \mathbf{Y}_0(z) \\ \mathbf{Y}_1(z) \end{bmatrix} = \begin{bmatrix} H_0(z) & H_1(z) \\ G_0(z) & G_1(z) \end{bmatrix} \begin{bmatrix} \mathbf{X}_0(z) \\ \mathbf{X}_1(z) \end{bmatrix} \tag{2.5}$$

A schematic plot is shown in Fig. 2.5 for DWT with a two-channel wavelet filterbank operating in the polyphase domain.

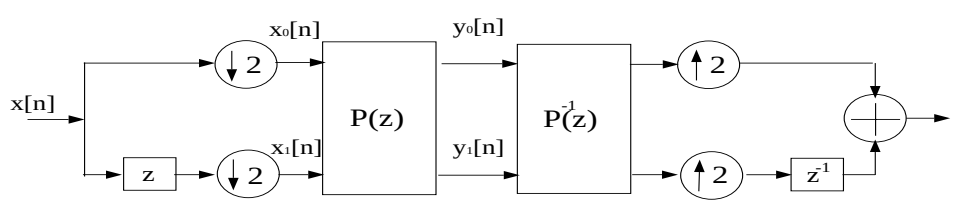



Figure 2.5: Two-channel wavelet filterbank in polyphase representation.

The main advantage of the polyphase domain transform computation is that polyphase matrix $\mathbf{P}(z)$ can be furhter factored and the factorization leads to fast DWT algorithms [45, 46, 47, 48]. Using the Euclidean algorithm, Daubechies and Sweldens [48] have shown that polyphase matrix $\mathbf{P}(z)$ of any PR FIR filterbanks can be factored into a product form of elementary matrices as

$$\mathbf{P}(z) = \begin{bmatrix} 1/K & 0 \\ 0 & K \end{bmatrix} \prod_{i=m}^{1} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \tag{2.6}$$

where $(s_i(z), t_i(z))$ are the *prediction* and *updating* filters, respectively, at stage $i$. It has been shown that such a lifting-factorization based DWT algorithm is, asymptotically for long filters, twice as fast as that of the standard algorithm (Theorem 8 in [48]). In Fig. 2.6, the forward and inverse DWT using lifting factorization are illustrated schematically.

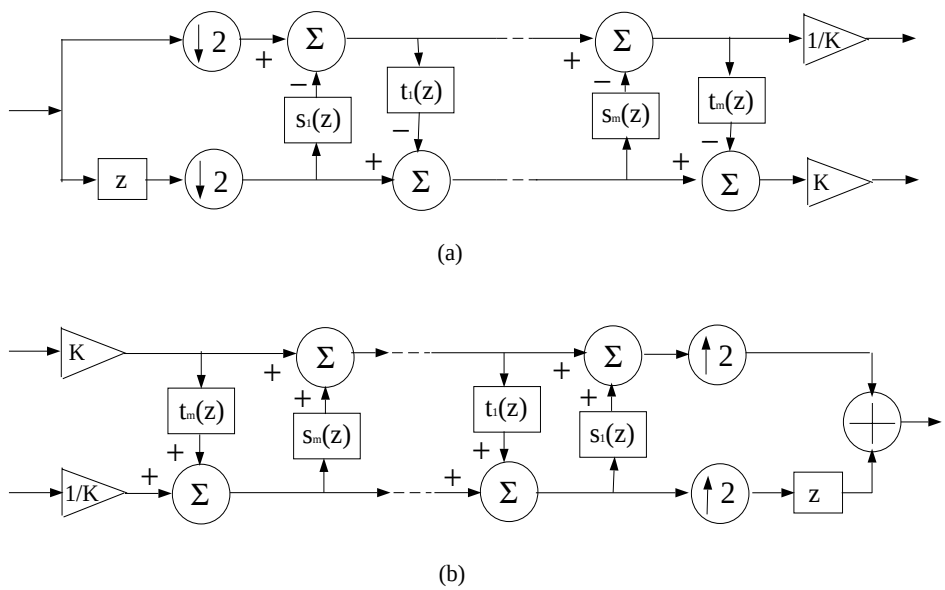



Figure 2.6: Wavelet transform via lifting. (a) Forward transform. (b) Inverse transform.

Notice that the elementary matrices in the lifting factorization are all triangular (upper or lower triangular) with constants in the diagonal entries. Such a choice of elementary matrices enables the implementation of the DWT to be *in-place* (see next section for details), a key factor different from other types of

factorizations (e.g., the lattice and ladder factorizations). While all these factorizations can reduce the DWT computation, the *in-place* feature can also reduce the transform memory. Consequently, the lifting algorithm is chosen as the baseline DWT algorithm for our proposed architecture designs.

### 2.3.3 Practical DWT System Design

For practical DWT system design under memory and delay constraints, choosing only a fast algorithm (e.g. the lifting algorithm) may not be sufficient. First, the complexity of the lifting algorithm is still linear with the size $N$ of the input data, i.e., $O(N)$. If a parallel system is used to further speed up the computation, the first problem to solve is that how to efficiently access the data allocated to other processors for correct boundary transform. Second, though the *in-place* feature of the lifting algorithm eliminates the buffer for intermediate results, it does not address the problem of extra buffer requirement when the input data has to be transformed on a block-by-block basis.

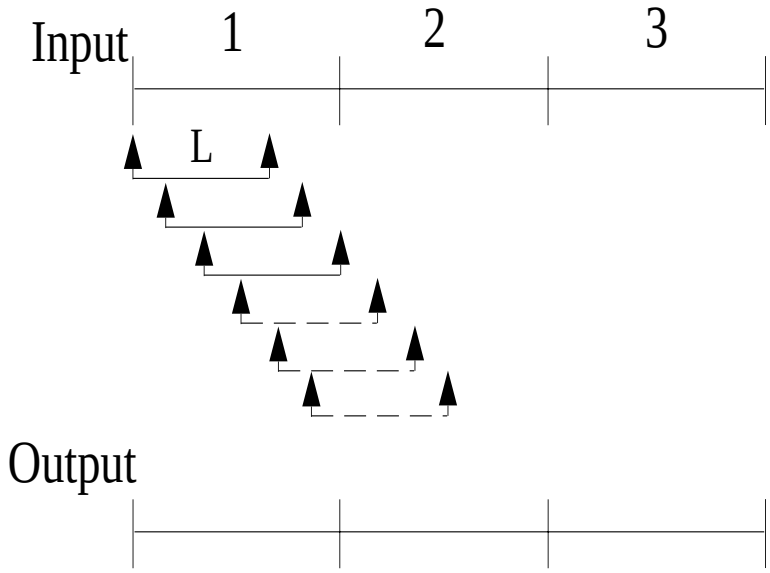



Figure 2.7: Boundary processing for DWT. When filter (length L) moves to the right boundary of block 1, it needs input data samples from block 2. In a sequential architecture, block 1 and 2 do not reside in memory at the same time. In a parallel architecture, block 1 and 2 are allocated to different processors.

In Fig. 2.7, we show the situation of DWT near block boundaries for one level decomposition. Obviously, extra buffer or communication is needed to ensure correct computations near data block boundaries. Such a problem also exists in cases of conventional linear filtering of long data sequences and can be dealt with using either *overlap-add* or *overlap-save* techniques (see Appendix A). However, because the DWT consists of *recursive* filtering operations on multilevel down-sampled data, direct applications of these two existing techniques may increase significantly the cost in terms of memory and/or communication.

Consider a $J$-level wavelet decomposition with block size $N$ and filter length $L$. Both *overlap-add* and *overlap-save* require an extra buffer (for boundary filtering operations) of size $L-2$ for each level of decomposition. If the overlap is done once for all decomposition levels (the SSWT approach by Kossentini [28]), the total overlap buffer size is $(2^J-1)(L-2)$ which increases exponentially with the increase of $J$. This becomes significant if deep decomposition and long wavelet filters are used. An alternative is to overlap at each level. In this case, the overlap buffer size is $J(L-2)$ for $J$-level decompositions. This, however, causes delay in parallel architectures since one processor has to wait the other to send new data after each level of decomposition (an approach described in [11, 24]). A third approach [25, 22] is to use boundary extension (e.g. symmetric extension) to approximate the data in the neighboring blocks. This completely eliminates the overlap buffer and also eliminates the communication for data exchanges between processors. Unfortunately, the DWT coefficients near block boundaries are computed incorrectly.

The above analysis thus shows the inefficiencies, in terms of memory and/or communication overhead, of DWT system designs which adopt the existing overlapping techniques. In the next section, we will introduce a novel *overlap-state* technique for DWT computation across block boundaries which can help to reduce the communication overhead in parallel architectures and the overlap buffer size in sequential architectures.

## 2.4 The Overlap-State Technique

In this Section, we first introduce the FSM model for DWT based on the lifting factorization. Then we present the *overlap-state* technique for DWT computation across consecutive data blocks, which can help to reduce significantly memory and communication overhead in DWT system designs.

### 2.4.1 The Finite State Machine Model

From the lifting point of view [58, 48], the elementary triangular matrices in the factorization (2.6) can be further classified as *prediction/lifting* and *updating/dual lifting* operations respectively. From a computational point of view, however, there is no big difference among these elementary matrices, each of which essentially updates one polyphase component at a time using linear convolutions.

Without loss of generality, we introduce notation $\mathbf{e}^i(z)$ to represent the elementary matrices. That is

$$\mathbf{e}^i(z) \equiv \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \qquad or \qquad \mathbf{e}^i(z) \equiv \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix}$$

Let the input be $\mathbf{X}(z)$ with polyphase components $(\mathbf{X}_0(z), \mathbf{X}_1(z))$ and $(x_0(n), x_1(n))$ in frequency domain and time domain respectively). Now define the intermediate states in the process of transformation, $\{\mathbf{X}^i(z), i = 0, 1, \cdots, 2m+1\}$, as

$$\begin{aligned} \mathbf{X}^i(z) &= \mathbf{e}^{i-1}\mathbf{e}^{i-2}\cdots\mathbf{e}^0\mathbf{X}(z) && (2.7) \\ &= \prod_{j=i-1}^{0} \mathbf{e}^j(z)\mathbf{X}(z) && (2.8) \\ &= \mathbf{e}^{i-1}(z)\mathbf{X}^{i-1}(z) && (2.9) \end{aligned}$$

where $\mathbf{X}^i(z)$ is the resulting signal after the first $i$ elementary matrices have been applied. Consider one lifting stage using a lower triangular elementary matrix

$\mathbf{e}^i(z)$ to update $\mathbf{X}^i(z)$ into $\mathbf{X}^{i+1}(z)$ as follows.

$$\begin{bmatrix} \mathbf{X}_0^{i+1}(z) \\ \mathbf{X}_1^{i+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ t^i(z) & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_0^i(z) \\ \mathbf{X}_1^i(z) \end{bmatrix} \tag{2.10}$$

$$= \begin{bmatrix} \mathbf{X}_0^i(z) \\ \mathbf{X}_1^i(z) + t^i(z)\mathbf{X}_0^i(z) \end{bmatrix} \tag{2.11}$$

As one can see, in this transformation step the polyphase component $\mathbf{X}_0^i(z)$ is unchanged while polyphase component $\mathbf{X}_1^i(z)$ is updated by adding a quantity computed from the other polyphase component. In time domain, this means that all even samples are preserved while all odd samples are updated. For an input vector $\mathbf{X}$ of size $N$ (assuming N even), the state transition can be written as

$$\begin{bmatrix} x^{i+1}(0) \\ x^{i+1}(1) \\ x^{i+1}(2) \\ x^{i+1}(3) \\ \vdots \\ x^{i+1}(2k) \\ x^{i+1}(2k+1) \\ \vdots \\ x^{i+1}(N-2) \\ x^{i+1}(N-1) \end{bmatrix} = \begin{bmatrix} x^i(0) \\ x^i(1)+\sigma(1) \\ x^i(2) \\ x^i(3)+\sigma(3) \\ \vdots \\ x^i(2k) \\ x^i(2k+1)+\sigma(2k+1) \\ \vdots \\ x^i(N-2) \\ x^i(N-1)+\sigma(N-1) \end{bmatrix} = \begin{bmatrix} x^i(0)+\sigma(0) \\ x^i(1)+\sigma(1) \\ x^i(2)+\sigma(2) \\ x^i(3)+\sigma(3) \\ \vdots \\ x^i(2k)+\sigma(2k) \\ x^i(2k+1)+\sigma(2k+1) \\ \vdots \\ x^i(N-2)+\sigma(N-2) \\ x^i(N-1)+\sigma(N-1) \end{bmatrix} \tag{2.12}$$

Denote $t^i(z) = \sum_{n=-a^i}^{b^i} t_n^i z^{-n}$ $(a^i \geq 0, b^i \geq 0)$, then the updating quantity $\sigma(n)$ can be computed as

$$\sigma(n) = \begin{cases} 0 & n = 2k \\ \sum_l t_l^i x_0^i(k-l) & n = 2k+1 \end{cases} \tag{2.13}$$

If $\mathbf{e}(z)$ is upper triangular, then odd samples are unchanged and even samples are updated. In this case, denote $s^i(z) = \sum_{n=-a^i}^{b^i} s_n^i z^{-n}$ $(a^i \geq 0, b^i \geq 0)$, then the

updating quantity $\sigma(n)$ for upper triangular matrix $\mathbf{e}(z)$ is

$$\sigma(n) = \begin{cases} \sum_l s_l^i x_1^i(k-l) & n = 2k \\ 0 & n = 2k+1 \end{cases} \tag{2.14}$$

An important observation is that, only one polyphase component is updated at each state transition and the updating quantity $\delta(n)$ only depends on samples from the other polyphase component. When updating even samples, only odd samples are needed and vice verse. This leads to the following three conclusions for states updating at each stage:

1. Whenever $\mathbf{X}^i$ is updated into $\mathbf{X}^{i+1}$, there is no need to keep the old value of $\mathbf{X}^i$ since no other updating will need it any more. In other words, every time we generate $\mathbf{X}^i$, we only need to store this set of values, i.e., we do not need to know any of the other $\mathbf{X}^j$, for $j < i$, in order to compute the output (the final wavelet coefficients).

2. The updated value of each sample $x^{i+1}(n)$ can be stored in the same memory space allocated for $x^i(n)$ since the old value $x^i(n)$ does not contribute to the updating of its neighbors and any later stage updating. For example, $x^i(1)$ can be over-written by $x^{i+1}(1)$ without affecting the updating of $x^i(3)$. This is the so-called *in-place* property of the lifting algorithm. Obviously, only a buffer of size $N$ is enough for the transform while the standard algorithm needs a buffer of size $2N$ ($N$ for the original input and $N$ for the transform outputs).

3. The updating of each sample $x^i(n)$ can be implemented *independently* from the updating of other samples. That is, there is no ordering of the updating between samples. For example, one can update $x^i(3)$ before or after the updating of $x^i(1)$ and obtain the same result.

For the polyphase matrix factorization, the necessary and sufficient condition for the above properties is that that the elementary matrix $\mathbf{e}^i(z)$ can only be in the form of lower/upper triangular matrices with constants on the diagonal entries as mentioned before. This key property of the lifting factorization guarantees that

the DWT can be computed *in-place*. That is, each raw input data sample $x(n)$ (initial state) is progressively updated into a wavelet coefficient (final state) using samples in its neighborhood. Thus the wavelet transform based on the polyphase factorization can be modeled as a FSM in which each elementary matrix $\mathbf{e}^i$ updates the FSM state $\mathbf{X}^i$ to the next higher level $\mathbf{X}^{i+1}$. The forward wavelet transform $\mathbf{Y}(z)$ can be written as

$$\begin{aligned}
\mathbf{Y}(z) &= \mathbf{P}(z)\mathbf{X}(z) && (2.15)\\
&= \prod_{i=2m}^{0} \mathbf{e}^i(z)\mathbf{X}(z) && (2.16)\\
&= \underbrace{\mathbf{e}^{2m}(z)\cdots\underbrace{\mathbf{e}^1(z)\underbrace{\mathbf{e}^0(z)\mathbf{X}^0(z)}_{\mathbf{X}^1(z)}}_{\vdots\quad \mathbf{X}^2(z)}}_{\mathbf{Y}(z)=\mathbf{X}^{2m+1}(z)} && (2.17)
\end{aligned}$$

and the inverse transform is

$$\begin{aligned}
\hat{\mathbf{X}}(z) &= \mathbf{P}^{-1}(z)\mathbf{Y}(z) && (2.18)\\
&= \prod_{i=0}^{2m} \mathbf{e}^{-i}(z)\mathbf{Y}(z) && (2.19)\\
&= \underbrace{\mathbf{e}^{-0}(z)\cdots\underbrace{\mathbf{e}^{-2m+1}(z)\underbrace{\mathbf{e}^{-2m}(z)\mathbf{Y}^0(z)}_{\mathbf{X}^{2m}(z)}}_{\vdots\quad \mathbf{X}^{2m-1}(z)}}_{\hat{\mathbf{X}}(z)=\mathbf{X}^0(z)} && (2.20)
\end{aligned}$$

where $\mathbf{e}^{-i}(z)$ is the inverse of $\mathbf{e}^i(z)$. The schematic plot of the DWT as a FSM is depicted in Fig. 2.8. A formal definition is given as follows [59].

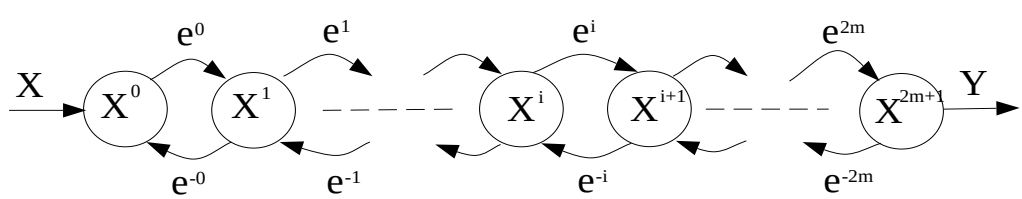



Figure 2.8: State transition diagram of DWT as a FSM.

**Definition** A Discrete Wavelet Transform can be modeled as a Finite State Machine, that is, a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ [59] where

$Q$ a finite set of states, $Q = \{\mathbf{X}^i(z)\}, i = 0, 1, \cdots, m$ with $m$ determined by the given factorization.

$\Sigma$ a finite set of events, $\Sigma = \{\mathbf{e}^i(z)\}, i = 0, 1, \cdots, m$, the lifting operations at each lifting stage.

$q_0$ the initial state, $q_0 = \mathbf{X}^0(z)$ (the raw input data).

$F$ the final state, $F = \mathbf{Y}(z) = \mathbf{X}^{2m}(z)$, (the wavelet transform output).

$\delta$ the transition function mapping $Q\text{x}\Sigma \rightarrow Q$, $\delta = \{(\mathbf{X}^i(z), \mathbf{e}^i(z)) \rightarrow \mathbf{X}^{i+1}(z)\}$.

### 2.4.2 Overlap-State

Assume there are $M$ elementary matrices $\{\mathbf{e}^i, i = 0, 1, \cdots, M-1\}$ in the factorization of the polyphase matrix $\mathbf{P}(z)$, then there are total $M$ states by the FSM definition. The FSM modeling tells us that, to compute the transform, one needs to help each and every sample $x(n)$ to complete its state transitions from state 0 up to state $M-1$ sequentially. This means that one has to compute the updating quantities $\{\sigma^i(n), i = 0, 1, \cdots, M-1\}$ (2.13) and (2.14) at all these stages. Unfortunately this can not be accomplished for samples near block boundaries. This happens when the input has to be transformed on a block-by-block basis due to buffer size limit or for purpose of parallel processing.

Consider one operation across data boundary using an upper triangular elementary matrix. Let the current state be $i$ and the input sequence $x^i(n)$ be segmented at point $2k$, refer to Fig. 2.9. In this state transition, even-indexed samples are updated using odd-indexed samples, The updating quantity $\sigma(2k)$ (2.14) is

$$\delta(2k) = \sum_l s_l^i x_1^i(k-l) \tag{2.21}$$

$$= \sum_l s_l^i x^i(2k-2l+1) \tag{2.22}$$

$$= \underbrace{\sum_{l=-a^i}^{-1} s_l^i x^i(2k-2l+1)}_{C(2k)} + \underbrace{\sum_{l=0}^{b^i} s_l^i x^i(2k-2l+1)}_{A(2k)} \tag{2.23}$$

where $C(2k)$ and $A(2k)$ are respectively the contributions from the causal and anti-causal part of filter $s^i(z)$.

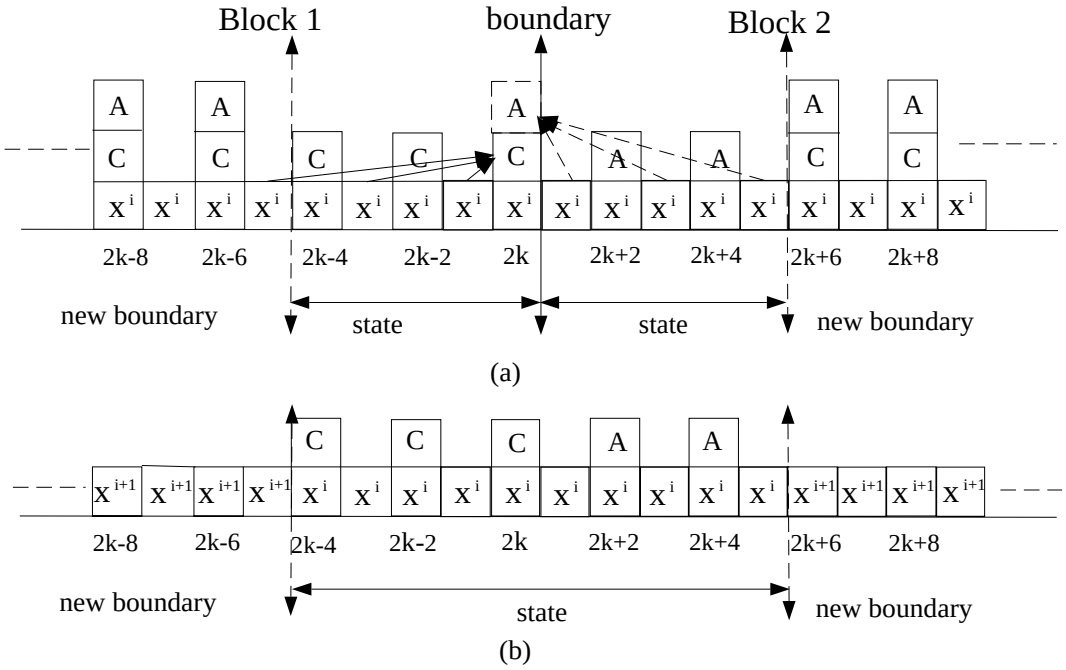



Figure 2.9: State transitions across block boundary using $\mathbf{e}^i$ with $a^i = 3$ and $b^i = 2$. (a) Partial computations near boundaries. (b) After updating, boundary samples stay in intermediate states. The "new boundary" separates fully updated output coefficients from partially computed ones.

Obviously, for samples near the block boundary we cannot compute both $C(2k)$ and $A(2k)$ due to the segmentation. Therefore $\sigma(2k)$ will not be available. As a result, these samples can not be updated into state $i+1$. In Fig. 2.9(a), for example, $x^i(2k)$ in block 1 can not be updated into $x^{i+1}(2k)$ since $\{x^i(2k+1), x^i(2k+3), x^i(2k+5)\}$ are in the right block and thus are not available at the time block 1 is transformed.

Consequently, $\sigma(2k)$, the updating factor for sample $x^i(2k)$ cannot be computed. Therefore, $x^i(2k)$ can not be updated into $x^{i+1}(2k)$. Rather than leaving $x^i(2k)$ in state $i$, we choose to partially update $x^i(2k)$ as $\bar{x}^i(2k) = x^i(2k) + C(2k)$

since $C(2k)$ can be computed from the causal neighborhood ( a function of $\{x^i(2k-1), x^i(2k-3), x^i(2k-5)\}$). The significance of this partial updating is that, one can free all the samples in the casual past for future processing and save memory. In this case, samples $\{x^i(2k-1), x^i(2k-3), x^i(2k-5)\}$ do not need to be buffered for the fully updating of $x^i(2k)$ since their contribution $C(2k)$ has already been added to the partial result $\bar{x}^i(2k)$. On the other hand, if we choose not to partially update $x^i(2k)$, then $\{x^i(2k-1), x^i(2k-3), x^i(2k-5)\}$ have to be buffered. The same partial updating happens also for samples $\{x^i(2k-2), x^i(2k-4)\}$ in the left block and samples $\{x^i(2k+2), x^i(2k+4)\}$ in the right block.

For the complete state transition from $i$ to $i+1$, we need to buffer in each block the following samples:

1. *Partially updated samples* such as $\{\bar{x}^i(2k), \bar{x}^i(2k-2), \bar{x}^i(2k-4)\}$ in the left block and $\{\bar{x}^i(2k+2), \bar{x}^i(2k+4)\}$ in the right block.

2. *Contributing samples* required by partially updated samples (in the other block) to complete the transform, such as $\{x^i(2m-1), x^i(2m-3)\}$ in the left block and $\{x^i(2m+1), x^i(2m+3), x^i(2m+5)\}$ in the right block.

For simplicity, these *partially updated samples* and *Contributing samples* will be called the *state* information hereafter. Obviously, as long as the *state* information is preserved at each stage, the transform can be completed at any later time. That is exactly what a FSM is supposed to do.

We mention that such a later processing is possible because partial updating in the right block (updating of $x^i(2m+2)$ and $x^i(2m+4)$) can be implemented *independently* from the partial updating in the left block (updating of $x^i(2m)$, $x^i(2m-2)$ and $x^i(2m-4)$) as discussed before. The partial updating does not remove any information needed by the other block, since it updates samples that are not inputs at the $i$-th state transition stage. The end state after application of $\mathbf{e}^i$ is shown in Fig. 2.9(b). As one can see, because partially updated samples cannot be used for processing, the size of the segment over which we can compute is reduced, so that the effective boundary is now reached before sample $x^{i+1}(2k-4)$ in block 1 and sample $x^{i+1}(2k+6)$ in block 2. Effectively, the physical boundary splits into two and extends inwards in both blocks. The next state transition via

$\mathbf{e}^{i+1}$ will operate only on samples in state $i+1$. All the samples between the two new boundaries become the state information and the same procedure repeats at each state transition stage.

To complete the transform for samples near the block boundary, the state information in neighboring blocks need to be exchanged. This can be done by overlapping the states between consecutive blocks. Thus we propose the *overlap-state* method for DWT computation across consecutive data blocks. The *overlap-state* procedure is shown in Fig. 2.10. In case of parallel processing, the implementation is shown in Fig. 2.11. Though only one state transition is shown in these two figures, the *overlap-state* design can be easily generalized to multiple state transitions at multiple decomposition levels because all these state transitions share the same three properties as given before (see section 2.4.1).

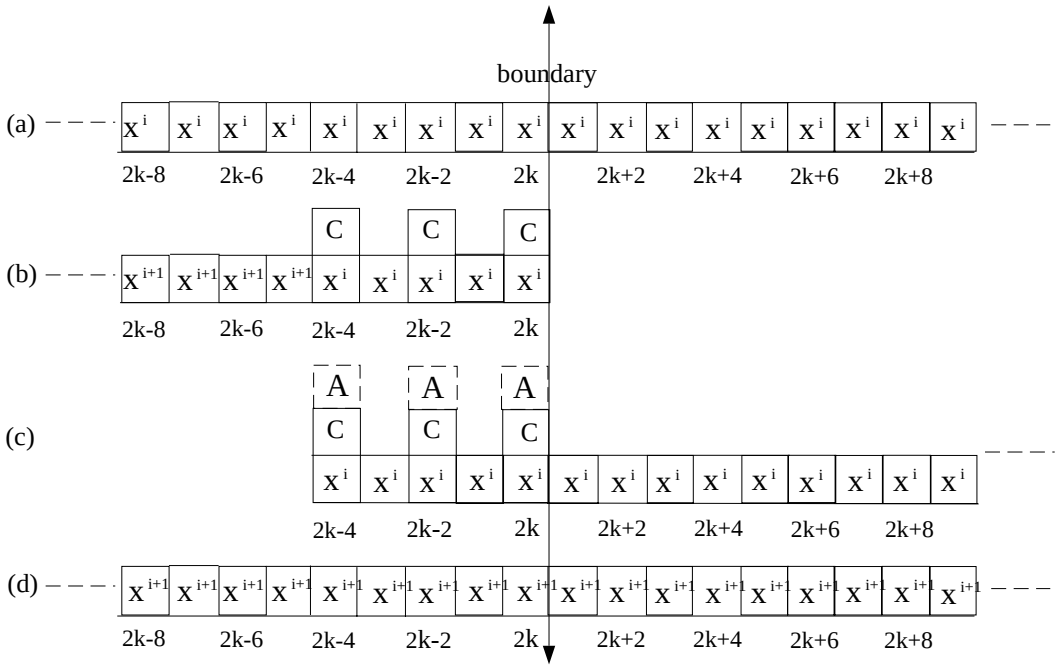



Figure 2.10: Sequential DWT using *overlap-state*. (a) Input in initial state $i$. (b) Block 1 consists samples up to $x^i(2m)$. After state transition, samples near block boundary are only partially updated (anti-causal filtering results not available). (c) Partially updated samples (state information) are overlapped with next block of input samples. They are now completely updated by adding their anti-causal filtering results. (d) Completely transformed (updated) input from state $i$ to $i+1$.

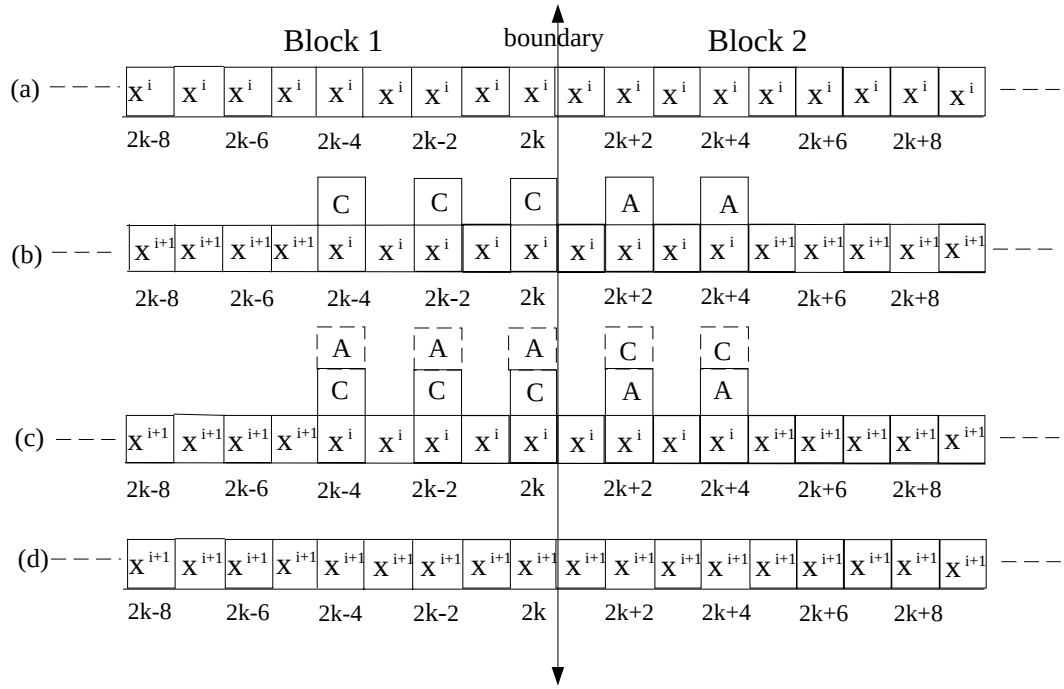



Figure 2.11: Parallel DWT using *overlap-state*. (a) Input in initial state $i$. (b) Input is partitioned over two processors. Block 1 and 2 are transformed separately and each have state information appearing near the block boundary. (c) State information is exchanged between two processors and the partially updated samples are fully updated. (d) Completely transformed (updated) input from state $i$ to $i+1$.

### 2.4.3 Performance Analysis

#### Buffer Size Analysis

Given a lifting factorization of the polyphase matrix $\mathbf{P}(z)$, we now show how much state information one need to store for the DWT computation, i.e., the overlap buffer size. This is a key factor for memory constrained sequential architecture design.

As shown before, at each stage, the *partially updated samples* and *contributing samples* need to be stored. Denote the total number of *partially updated samples* as $B_1^i$ and the total number of *contributing samples* as $B_2^i$. Writing $s^i(z)$ and $t^i(z)$ as

$$s^i(z) = \sum_{n=-a^i}^{b^i} s^i(n)z^{-n} \qquad t^i(z) = \sum_{n=-a^i}^{b^i} s^i(n)z^{-n} \tag{2.24}$$

where $a^i \geq 0, b^i \geq 0$. Then $B_1^i = a^i, B_2^i = b^i$. The number of samples that must be buffered at stage $i$, $B^i$, is $B^i = a^i + b^i$. Assume there are $N$ state transitions in the factorization of $\mathbf{P}(z)$, the buffer size $B_s$ for one level decomposition is

$$B_s = \sum_{i=0}^{N-1} B^i \tag{2.25}$$

$$= \sum_{i=0}^{N-1} (a^i + b^i) \tag{2.26}$$

Since the lifting factorization of a given polyphase matrix is not unique, obviously one would choose the factorization which gives the minimum $B_s$ if the amount of memory is limited. An alternative way to find out the buffer size is to graphically plot the state transitions for a given factorization. See Appendix B for details.

### Communication

The communication delay is the time used for exchanging data between adjacent processors. In existing parallel algorithms [11, 24], before each level of decomposition, $(L-2)$ boundary samples need to be communicated to the adjacent processors ($L$ is the filter length). Using the communication model given in (2.2), the total communication time $D_{old}$, for a $J$ level wavelet decomposition, is

$$D_{old} = J(t_s + (L-2)t_w + t_p) \tag{2.27}$$

In the proposed parallel algorithm, using the *overlap-state* technique, the data exchange can be delayed after the independent transform of each block and only one communication is necessary. An example of three level decompositions is shown in Fig. 2.12 and compare this to Fig. 2.1. Furthermore, the size of the state information at each stage $B_s$ is upper bounded by $(L-2)$. So the communication time in the proposed algorithm is upper bounded by

$$D_{new} \leq t_s + J(L-2)t_w + t_p \tag{2.28}$$

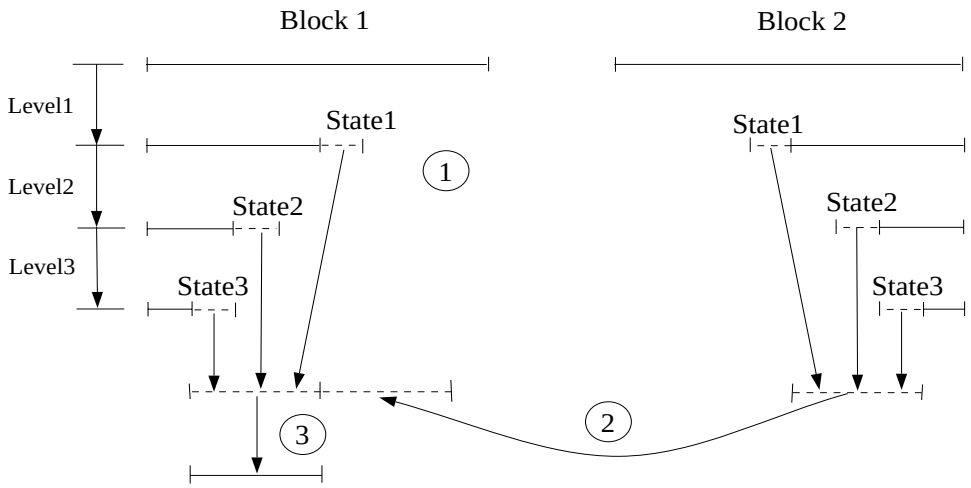



Figure 2.12: An example dataflow chart of a three-level wavelet decomposition using the proposed *Overlap-State* technique. Solid lines: completely transformed data; Dashed lines: partially transformed data. Operation 1: each block transforms its own allocated data independently and state information is buffered; Operation 2: state information is communicated to neighboring blocks; Operation 3: complete transform for the boundary data samples.

As one can see, the communication overhead is reduced in the proposed parallel algorithm because: $(i)$ the number of communication times is reduced; and $(ii)$ the amount of data exchanged is reduced. Essentially, the *overlap-state* technique enables us to exchange more data in one communication setup rather than exchanging a small amount of data in multiple communication setups. It is, however, important to emphasize that how much this communication overhead reduction contributes to the reduction of the total computation time will strongly depend on the parallel system communication link design. Clearly the slower the inter-processor communication, the larger the gain and vice versa.

### 2.4.4 Delayed Normalization

Although the lifting based DWT algorithm has been shown to be twice as fast as that of the standard Standard algorithm by Daubechies and Sweldens, this is only true in general asymptotically for long filters [48]. In this section we introduce a simple technique, *Delayed Normalization*, which can help to reduce the computation of multilevel wavelet decompositions.

As one may have noticed, the last matrix factor in the polyphase factorization form (2.6), is a normalization factor which scales the lowband and highband coefficients respectively. This normalization factor will appear at each level of decomposition for a multilevel wavelet decomposition. Since the wavelet transform is a linear operation and multiplication is commutative for linear operations, this normalization (multiplication) operation can actually be delayed until the last level decomposition. By doing so, computations can be saved.

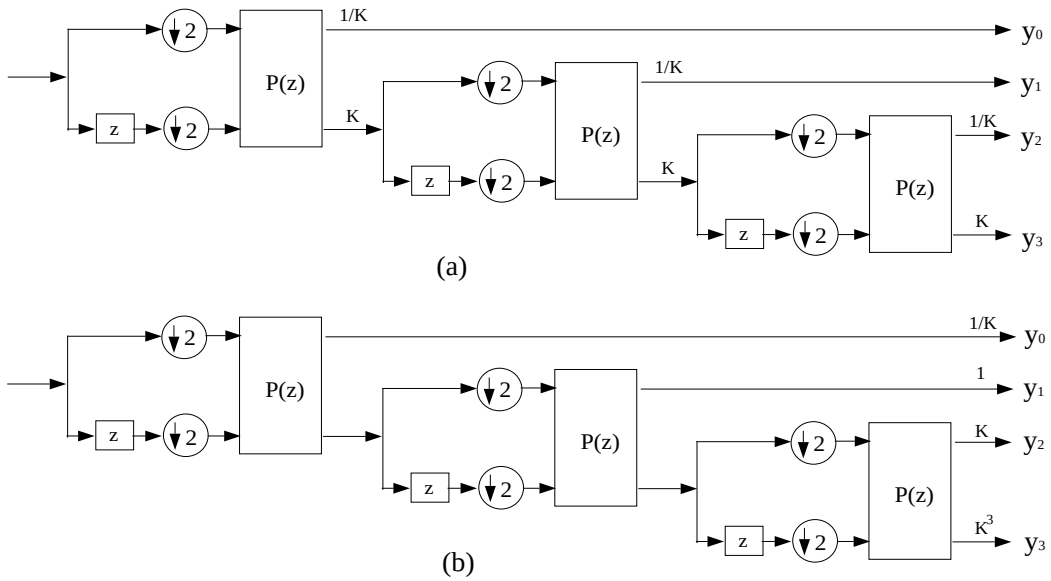



Figure 2.13: Illustration of delayed normalization. (a)Recursive two-channel lifting. (b)Two channel lifting with delayed normalization.

One example of a three-level octave-band wavelet decomposition is shown in Fig. 2.13. Interestingly, normalization operations for all the $y_1$ coefficients can be all eliminated provided that the same wavelet filterbanks are applied at each stage. If different wavelet filterbank is used at different levels of decomposition, then in general only one normalization (multiplication) operation is necessary for each wavelet transform coefficients. Obviously such a delayed normalization technique can also be used for multidimensional wavelet decompositions and wavelet packet decompositions. In Fig. 2.14, a one-level 2D wavelet decomposition is shown with recursive normalization in (a) and delayed normalization in (b).

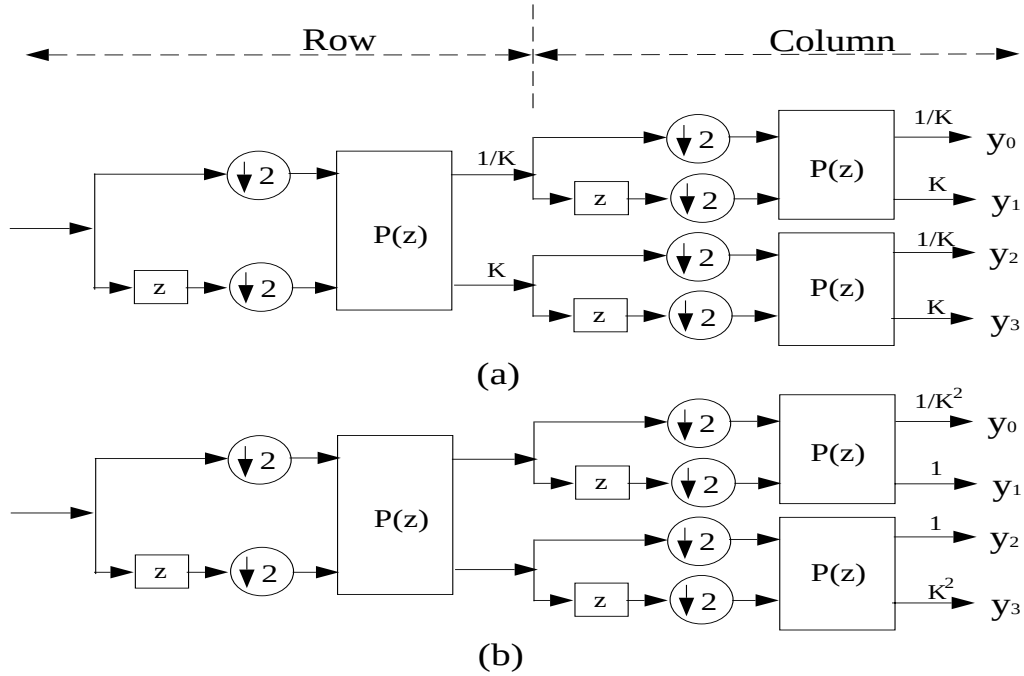



Figure 2.14: A 2D DWT example of delayed normalization.

We now give the performance analysis for 1D octave-band wavelet decomposition. Let the input data sequence length be $N$ and the decomposition level be $J$. The computational costs of the standard algorithm, the lifting scheme, and the lifting scheme with delayed normalization are denoted respectively as $C_M^J$, $C_L^J$, and $C_{L'}^J$. The cost unit we use is the average number of multiplications and additions per output point. Then

$$C_M^J = C_M^1(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{J-1}}) \quad (2.29)$$

$$= 2C_M^1(1 - 2^{-J}) \quad (2.30)$$

where $C_M^1$ is the number of multiplications and additions per output point for one level decomposition using the standard algorithm. Accordingly, the lifing cost is

$$C_L^J = 2C_L^1(1 - 2^{-J}) \quad (2.31)$$

For the lifting scheme with delayed normalization, the whole wavelet transform can be decomposed into two parts. One is the normal lifting operation part which

Table 2.2: Costs comparison for multilevel wavelet decompositions ($J > 3$)

| Wavelet | Standard | Lifting | | Lifting with Delayed Normalization | |
|---|---|---|---|---|---|
| | | cost | speedup | cost | speedup |
| Haar | 1.5 | 1.5 | 0% | 1.5 | 0% |
| D4 | 7 | 4.5 | 56% | 3.5 | 100% |
| D6 | 11 | 7 | 57% | 6 | 83% |
| (9-7) | 11.5 | 7 | 64% | 6 | 92% |
| (4,2) B-spline | 8.5 | 5 | 70% | 4 | 113% |

lasts for $J$ levels without normalization. For this part the one-level average cost is $C^1_{L'} = C^1_L - 1$ since one normalization/multiplication is saved for each coefficient. The second part is the final normalization part for all the coefficients. This part incurs cost 1 (one multiplication) per output point. So the total average cost is

$$C^1_{L'} = 2(C^1_L - 1)(1 - 2^{-J}) + 1 \tag{2.32}$$

$$= 2C^1_L(1 - 2^{-J}) + 2^{-J-1} - 1 \tag{2.33}$$

$$= C^J_L + 2^{-(J-1)} - 1 \tag{2.34}$$

If $N$ is large enough such that $J$ can be large enough, then in the limit $C^1_{L'}$ is on an average one operation fewer than that of a pure lifting scheme. In Table.2.4.4 we show how this will affect the algorithm relative speedup using the same filters given by Daubechies and Sweldens [48].

The above performance analysis applies for transforms with different wavelet filters at each stage. We have made the assumption that $J$ is large enough such that $2^{-(J-1)}$ is negligible. If the same filterbank is used at all decomposition stages, the assumption can be further relaxed.

Recall that the normalizations for $y_1$ coefficients can all be eliminated ( see Fig. 2.12). The savings is 0.25 since one-quarter of the total input data samples do not have to be scaled. Thus average cost of the normalization part should be 0.75 rather than 1 per output point. Taking this into consideration, as long as $J$ is large enough such that $2^{-(J-1)} \leq 0.25$, the above cost estimation $C_{L'}$ is

accurate. That is equivalent to having $J \geq 3$ which is a reasonable assumption for most practical wavelet applications.

Further reduction of the normalization operation is possible if we jointly design the DWT system and the immediate data processing system. For example, in a wavelet data compression system, wavelet coefficients will be quantized immediately after transform. Such a system is shown in Fig. 2.15(a). The $(Q_i, i = 0, 1, 2, 3)$ are quantizers designed for wavelet coefficients in different subbands. Obviously, the normalization operation can be done jointly with this quantization operation thus can be completely eliminated from the transform point of view. This is shown in Fig. 2.15(b). For other applications, such as noise reduction using thresholding, this computation reduction is also possible. Compared to independent transform and quantization, computation can be saved if designed jointly.

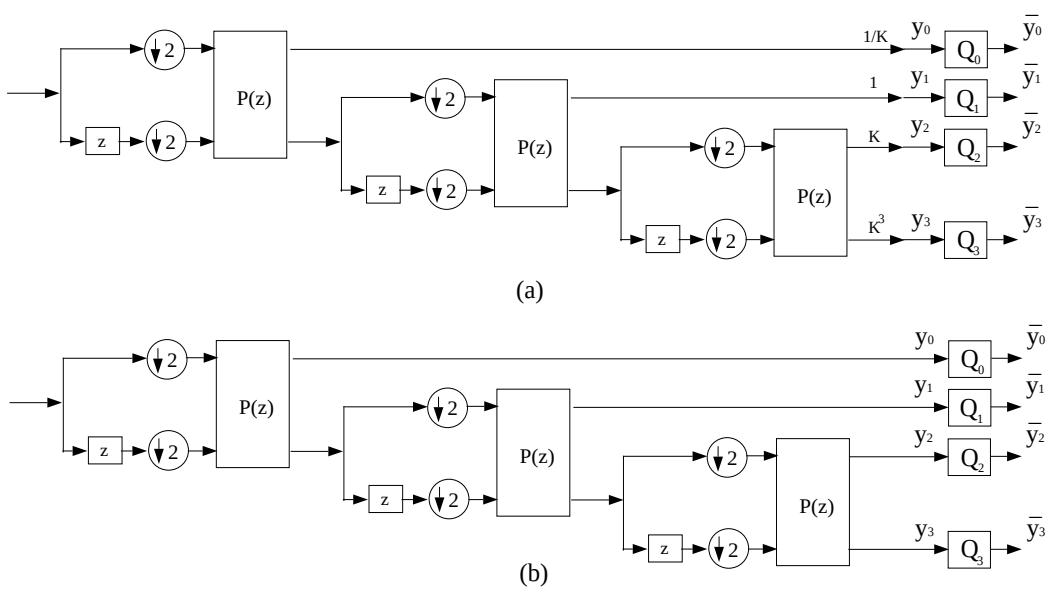



Figure 2.15: Joint design of transform and quantization to reduce computation. (a) Independent transform and quantization. (b) Joint transform and quantization. The normalization operation is merged with the quantization.

## 2.5 The Proposed DWT Architectures

In this section, we present first generic sequential and parallel architecture designs for 1D DWT using the *Overlap-State* technique. Variations are then detailed for 2D separable DWT systems.

### 2.5.1 1D Systems

**Sequential** In Fig. 2.16 the proposed sequential DWT system is shown and teh C-code sequential algorithm is given in Table 2.5.1. The input data sequence is first segmented into non-overlapping blocks of length $N$ and fed into the FSM one block at a time. The *state* information, however, is saved so that after one block has been computed the next one can use it. After transformation, the wavelet coefficients are concatenated together to give the final result.

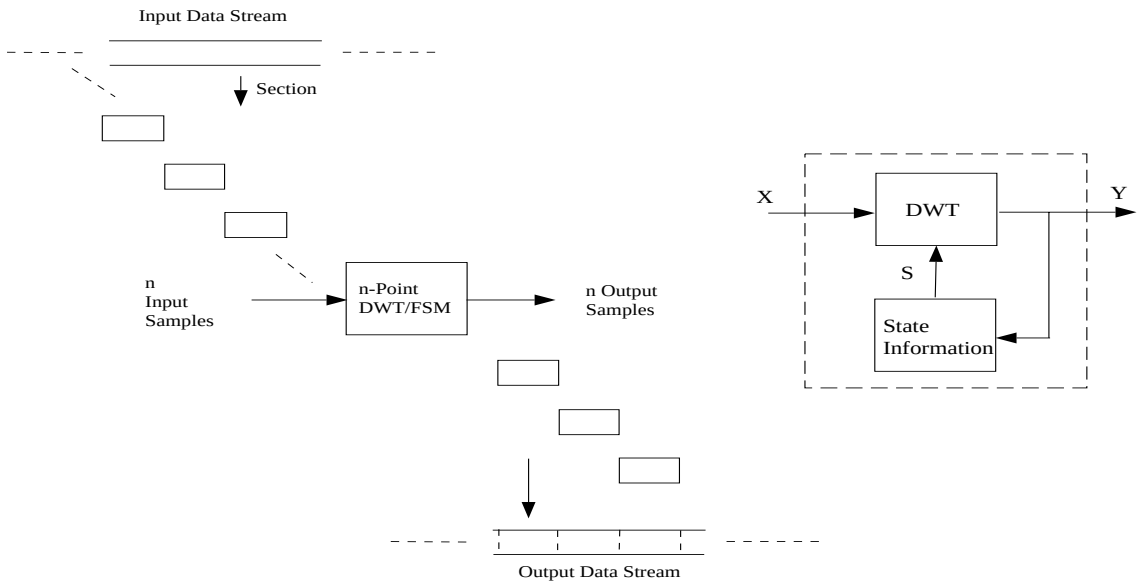



Figure 2.16: The proposed sequential DWT architecture.

As one can see, the general system structure for DWT computation is essentially the same as that in the standard *overlap-add* approach. The DWT/FSM acts as a state machine with memory and the state information (partially computed boundary samples from the previous block) at multiple decomposition level are overlapped. This helps to reduce the memory requirement for the transform

computation. This overlap leads to output delay in practice, i.e., the $n$ output samples shown in Fig. 2.16 are delayed relative to the $n$ input samples.

Table 2.3: The proposed sequential DWT algorithm.

```
begin
   initialize state S;
      for (k = 0; k < N; k++)
         {
            J-level wavelet transform for block k;
            update state S;
         }
end
```

In Table. 2.5.1 the required overlap buffer size, $B_s$, of different sequential DWT algorithms are given. For an $N$-point input data block, if the lifting algorithm is implemented, the total buffer size is $N + B_s$. The system throughput $\eta$ is

$$\eta = \frac{N}{N + B_s} \tag{2.35}$$

Obviously, the proposed sequential DWT algorithm, using the *overlap-state* technique, requires a smaller overlap buffer size $B_s$ and thus improves the system throughput. However, if $N >> O(JL)$ then the relative improvement becomes small. On the other hand, if $N = 0$ when all completely transformed coefficients are immediately transfered (e.g., the line-based system in [1, 55]), the savings in memory can be significant (details are given in the next section).

Table 2.4: Overlap buffer size $B_s$ in 1D DWT for $J$-level decompositions using a $L$-tap wavelet filterbank.

| | *SSWT[28]* | *RPA[6]* | *Proposed* |
|---|---|---|---|
| L-tap | $(2^J - 1)(L - 2)$ | $J(L - 2)$ | $\leq J(L - 2)$ |
| (9,7) | $7(2^J - 1)$ | $7J$ | $4J$ |
| (2,10) | $8(2^J - 1)$ | $8J$ | $4J$ |
| CDF(4,2) | $5(2^J - 1)$ | $5J$ | $3J$ |

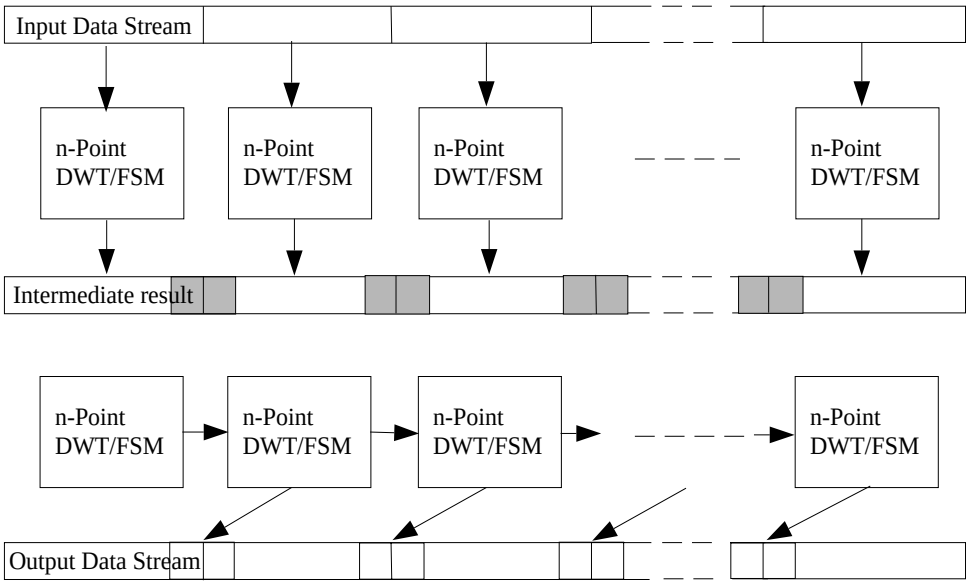



Figure 2.17: The proposed parallel DWT architecture. In *Split* stage, each processor computes its allocated data independently up to the required decomposition level. In *Merge* stage, a one-way communication is initiated to communicate the state information to neighboring processors. A postprocessing operation is then started to complete the transform for boundary samples.

**Parallel** In Fig. 2.17 the proposed parallel DWT architecture is shown and a C-code algorithm is given in Table 2.5.1. The input data is uniformly segmented non-overlapping blocks and allocated onto $p$ available processors. Each processor computes its own allocated data up to the required wavelet decomposition level. This stage is called *Split*. The output from this stage consists of (i) completely transformed coefficients and (ii) the state information (partially updated boundary samples). In the second stage, *Merge*, a one-way communication is initiated and the state information is transfered to the neighboring processors. The state information from the neighbor processor is then combined together with its own corresponding state information to complete the whole DWT transform.

As shown before, the proposed parallel architecture only requires one communication between neighboring processors for $J$-level decompositions. The amount of data exchanged is also less than that in direct overlapping approaches [11, 24]. Therefore, the communication delay is reduced.

Table 2.5: The proposed parallel DWT algorithm.

```
begin{ transform in processor p}
   for( j = 0; j < J; j++)
      {
         transform at current level j.
         store state information.
      }
   send state information to processor p + 1;
   receive state information from processor p − 1;
   for( j = 0; j < J; j++)
      {
         transform boundary data samples at current level j.
      }
end
```

### 2.5.2 2D Systems

In Fig. 2.18 an example 2D DWT with two level decompositions is shown. The data is row transformed first and then column transformed. Naturally, data samples along block boundaries can not be fully transformed due to lack of neighboring samples. These constitute the row and column state information at each level. Let us introduce some notations first. Let $N_r, N_c$ be the width and the height of the data block, respectively. For decomposition level $j = 0, 1, \cdots, J-1$, define

- $\{W_{r0}^j, W_{r1}^j\}$: numbers of partially transformed samples near left and right boundaries respectively in a row. $\{W_{c0}^j, W_{c1}^j\}$: defined similarly for a column.
- $\{N_r^j, N_c^j\}$: length of a row and a column respectively before the start of the decomposition at each level.
- $\{M_r^j, M_c^j\}$: number of completely transformed samples in a row and a column respectively.
- $B_s^j$: total number of partially updated samples, i.e., the size of the buffer to hold the state information for further processing.

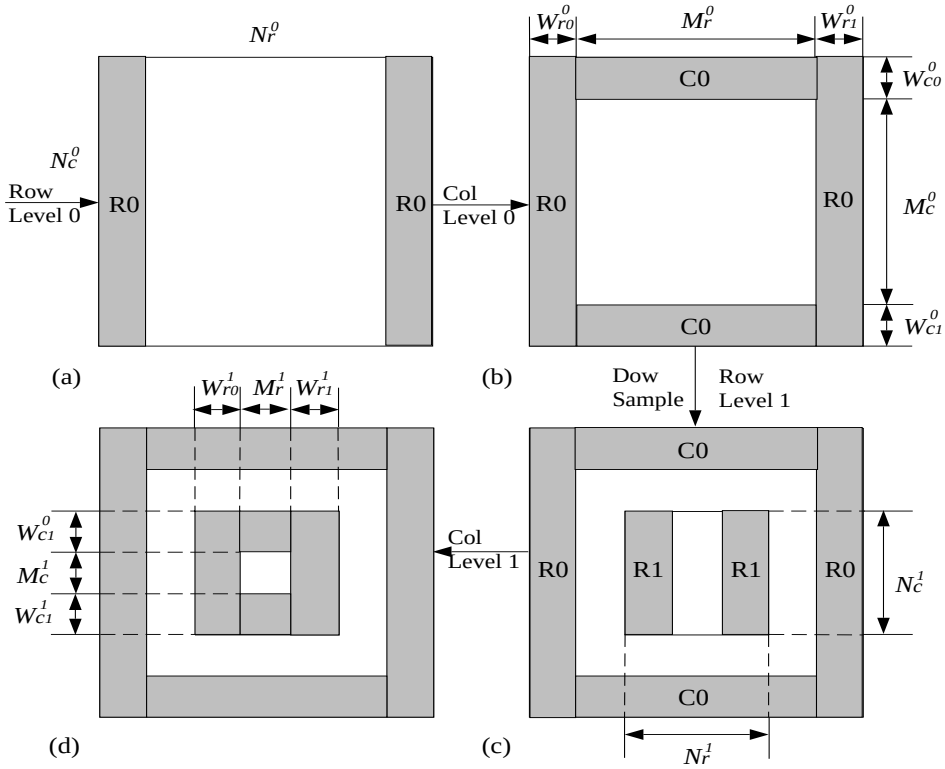



Figure 2.18: 2D DWT/FSM illustration. Shaded areas represent the state information with $R0/C0$ the row/column state information at level 0 and so on. The input block is first row transformed (a) and column transformed (b) at level 0, then downsampled (taking the LL0 subband) and row transformed at level 1 (c), and column transformed at level 1 (d).

The following identities between these defined quantities at each level $j$ can be derived, refer to Fig. 2.18.

$$M_r^j = N_r^j - W_{r0}^j - W_{r1}^j \tag{2.36}$$

$$M_c^j = N_c^j - W_{c0}^j - W_{c1}^j \tag{2.37}$$

$$N_r^j = \begin{cases} \lfloor M_r^{j-1}/2 \rfloor & j \geq 1 \\ N_r & \text{j=0} \end{cases} \tag{2.38}$$

$$N_c^j = \begin{cases} \lfloor M_c^{j-1}/2 \rfloor & j \geq 1 \\ N_c & \text{j=0} \end{cases} \tag{2.39}$$

$$B_s^j = N_r^j N_c^j - M_r^j M_c^j \tag{2.40}$$

Upon completion of all $J$-level decompositions, we have

$$B_s = \sum_{j=0}^{J-1} B_s^j \tag{2.41}$$

$$B_e = B - B_s \tag{2.42}$$

$$= N_r N_c - \sum_{j=0}^{J-1} B_s^j \tag{2.43}$$

where $B_s$ is the total buffer size necessary to store the state information at all decomposition levels and $B_e$ is the effective block size, i.e., number of wavelet coefficients that can be transfered to the next stage for processing, thus freeing up memory.

### 2.5.3 Sequential Architectures

**Strip Sequential** In this case, the buffer is organized to hold one strip of data at a time. Equivalently, this is when $B = WN_c$ or $B = N_r H$ where $W$x$H$ is the original data size. This scenario is depicted in Fig. 2.19 for $B = WN_c$. Because the input data is segmented only in column direction, state information (partially transformed samples) will only appear along the column direction. Certainly, some type of boundary extension techniques, such as the symmetric extensions, have to be used for the transform near the left and right row boundaries. For the transform of the very first strip, extension is also needed for the upper and lower boundaries of each column. Each strip takes over the state information left by the previous strip to transform its own data. Upon completion, it also generates the state information for the next strip. Then the strip slides down and the DWT is calculated strip-by-strip with state information overlapped between strips.

At the bottom of Fig. 2.19 a blow-up version of the state information is shown. For a $J$-level decomposition, the state buffer size $B_s$ can be calculated as

$$B_s = \sum_{j=0}^{J-1} B_s^j \tag{2.44}$$

$$= \sum_{j=0}^{J-1} W W_{c1}^j 2^{-j} \tag{2.45}$$

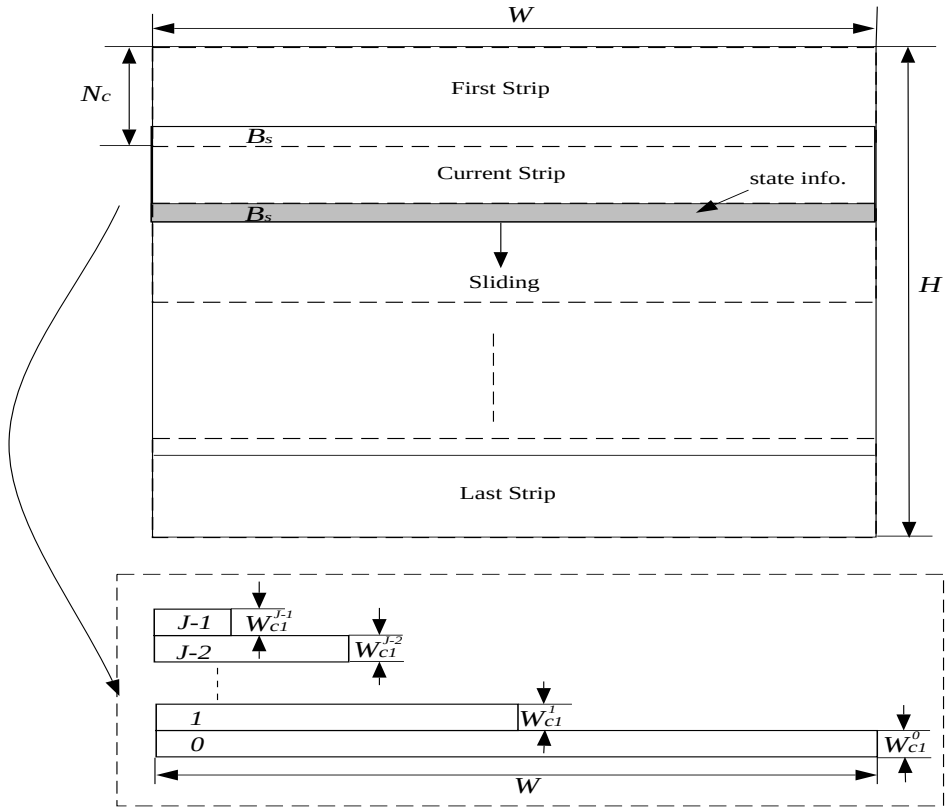



Figure 2.19: Strip sequential DWT system diagram. The input is segmented into data strips which are transformed sequentially from top to bottom.

Obviously, $B_s$ is proportional to the row length $W$ for the case depicted in Fig. 2.19. To reduce the state buffer size, the segmentation should choose the dimension with large data size. That is, if $W > H$ then segment along row direction and segment along column direction if otherwise.

In Table 2.5.3 comparisons of our proposed algorithm with existings ones for the minimum memory requirements. As one can see, the proposed system can produce significant memory savings. Consider a color image size of 4096x4096 where each color component sample is stored as a 4 bytes floating point number for DWT computation. In this case, one image scanline requires 48KB. Using the Daubechies (9,7) wavelet filterbank ($L = 9$), for a 3-level decomposition, the total memory would be 588KB if using the RPA algorithm (the approach given in [1, 55]). Using the *overlap-state* technique, the buffer size can be reduced to $296KB$.

Table 2.6: Comparison of memory requirements where $W$ is the width of the image scanline and $\alpha = (2^J - 1), \beta = (1 - 2^{-J})$

| | *SSWT[28]* | *RPA[6]* | *Proposed* |
|---|---|---|---|
| L-tap | $W\alpha(L-2)$ | $2W\beta(L-2)$ | $\leq 2W\beta(L-2)$ |
| (9,7) | $7W\alpha$ | $14W\beta$ | $8W\beta$ |
| (2,10) | $8W\alpha$ | $16W\beta$ | $8W\beta$ |
| CDF(4,2) | $5W\alpha$ | $10W\beta$ | $6W\beta$ |

**Block Sequential** In this case, the buffer is divided into two parts, one for holding the state information and one for new input samples in the sliding transform window. Equivalently, this is when $B = B_s + N_r N_c$. This scenario is depicted in Fig. 2.20.

As one can see, the data is segmented into blocks of size $N_r$x$N_c$ and transformed one block at a time. Since boundary extensions can be applied for the left and up boundaries of the very first block $A$, state information $\{A_r, A_c\}$ will appear only on the right and down side of the block upon completion of the transform. The $\{A_r, A_c\}$ correspond respectively the partially transformed row and column samples. When the window slides right to the position of block $B$, only the row state information $A_r$ can be overlapped. This shows that $A_r$ can be fully transformed by overlapping while $A_c$ has to be buffered for later processing. Same as block $A$, the column state information generated by $B$ also has to be buffered. This process continues until the completion of transforms of all the blocks in the first block row. By that time, the column state information has accumulated to the size $B_s$ exactly same as that of the sequential strip DWT, refer to (2.44).

It turns out that the state buffer size $B_s$ will not increase beyond this point. This can be verified by checking the first block $C$ in the second block row. For clarity of illustration, the second row is drawn separately from the first block row in Fig. 2.20. Actually, the two block rows are overlapped with the part of the state information. That is, block $C$ takes over the state information $A_c$ to complete the transform. When the transform stops, state information also appears on the right and down boundaries of block $C$. However, since $A_c$ has now been fully transformed and hence can be transfered out, $C_c$ can be written into the locations

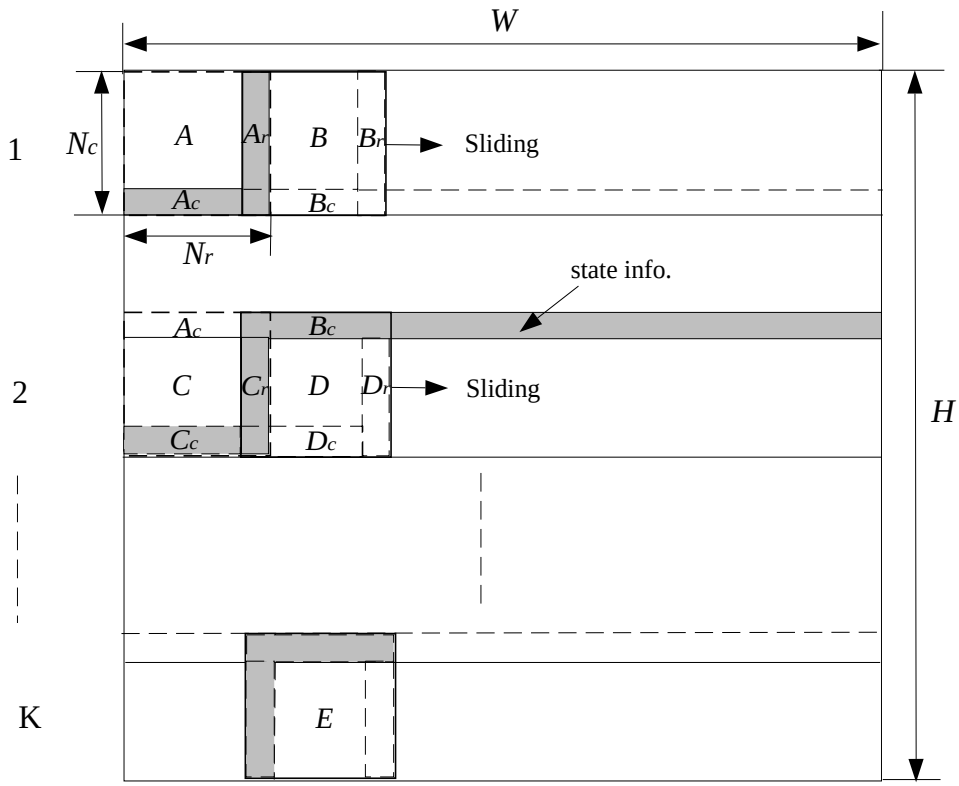



Figure 2.20: Block sequential DWT system diagram. The input is segmented into blocks which are transformed from left to right and from top to bottom.

of $A_c$ without increase of the total state buffer size $B_s$.

The most general case of sequential block DWT is depicted for block $D$. The block $D$ overlaps with previously generated state information in both the row and column directions, $\{C_r, B_c\}$. When it finishes its transform, it leaves $\{D_c, D_r\}$ for later processing. The transform of block $E$ in the last block row is the same as that of $D$ except that boundary extension can be used in the column direction.

To study the system throughput, consider the problem how large the buffer has to be in order to transform a block data of size $N$x$N$ at a time. This is typical in a transformed-based image coding applications where images are coded on a block-by-block basis. Assume the buffer is of size $N_B$x$N_B$. In Table 2.5.3, $N_B$ is given for $J$-level wavelet decompositions using different wavelet filterbanks and overlapping techniques. Taking the Daubechies (9,7) filterbank as an example. Assume the decomposition level is $J = 3$. If the block size is of 32x32, then

$N = 32$. Using SSWT, then $N_B = 32 + 49 = 81$ which means a buffer size of $81x81$ is needed to compute DWT of a data block $32x32$. The throughput $\eta$ in this case is approximately 16%. Using RPA, then $N_B = 45$ and the throughput increases to 50%. If using the *overlap-state* technique, then $N_B = 39$ and the throughput increases to 64%.

Table 2.7: Comparison of memory requirements where $\alpha = (2^J - 1), \beta = (1 - 2^{-J})$

| | *SSWT[28]* | *RPA[6]* | *Proposed* |
|---|---|---|---|
| L-tap | $N + \alpha(L-2)$ | $N + 2\beta(L-2)$ | $\leq N + 2\beta(L-2)$ |
| (9,7) | $N + 7\alpha$ | $N + 14\beta$ | $N + 8\beta$ |
| (2,10) | $N + 8\alpha$ | $N + 16\beta$ | $N + 8\beta$ |
| CDF(4,2) | $N + 5\alpha$ | $N + 10\beta$ | $N + 6\beta$ |

### 2.5.4 Parallel Architectures

**Block Parallel**

As shown before, in the first phase *Split* each processor is allocated with its portion of data and starts the transform all the way to the required decomposition level $J$. Upon completion, the data configuration at each processor is shown in Fig. 2.21(a). The center part of each block is completely transformed while the boundaries are left with the partially transformed samples, i.e., the state information. The next stage *Merge* is to communicate the state information and complete the transform for boundary samples. If the *single-port* model is used, then three communications is necessary to complete the transform, one for row state, one for column and one for the intersection of row and column state. However, if the *multi-port* model is used, the row and column state information exchange can be implemented simultaneously thus reducing one communication. This *Merge* process is shown in Fig. 2.21 from (a) to (d) for the *single-port* model. If the *multi-port* model is used, (a) and (b) can be combined to simultaneously transmit/receive the row and column state information to/from neighboring processors. This is in contrast to the observation given in [10] that "The 2D DWT algorithms seem not able to effectively utilize more than a single communication port at a time",

our analysis show that using multi-port model, the communication overhead can actually be reduced compared to that of a single-port model.

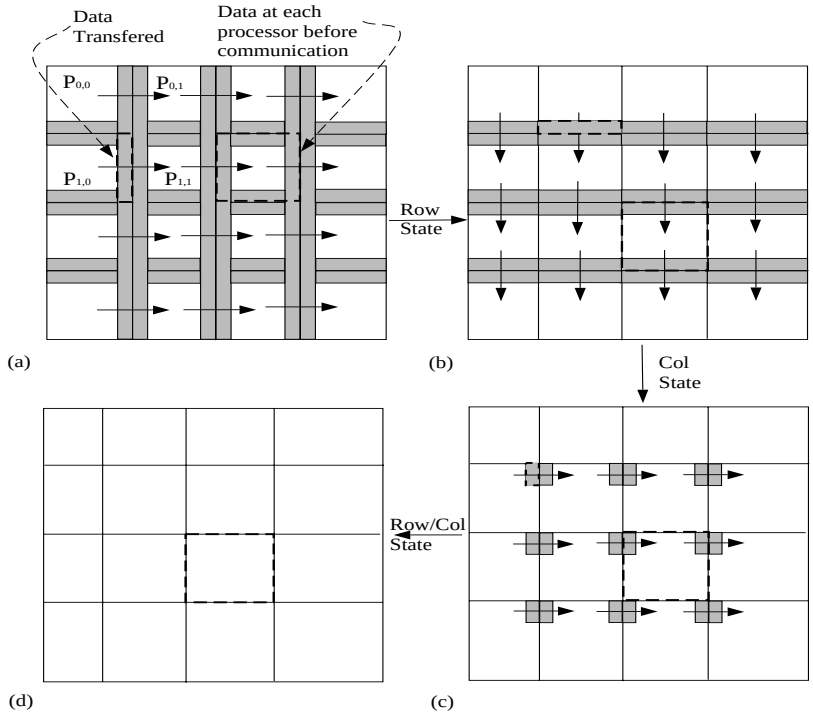



Figure 2.21: *Merge* operations in 2D mesh processor network. $(a)$transfer row state information from $P_{i,j}$ to $P_{i,j+1}$; $(b)$transfer column state information from $P_{i,j}$ to $P_{i+1,j}$; $(c)$transfer newly generated row state information from $P_{i,j}$ to $P_{i,j+1}$; $(d)$complete transform for boundary samples. Notice the total amount of data in each processor in the final state $(d)$ is different from the original uniform allocation due to the *Merge* operations.

### Strip Parallel

In the first stage *Split*, each processor is allocated with its own strip and transforms up to the required level of decomposition $J$. Since no segmentation is done in the row direction, state information obviously will only appear along up and down boundaries in each block. This is shown in Fig. 2.22. Next *Merge*, only one communication is necessary to transfer/receive the column state information from neighboring processors.

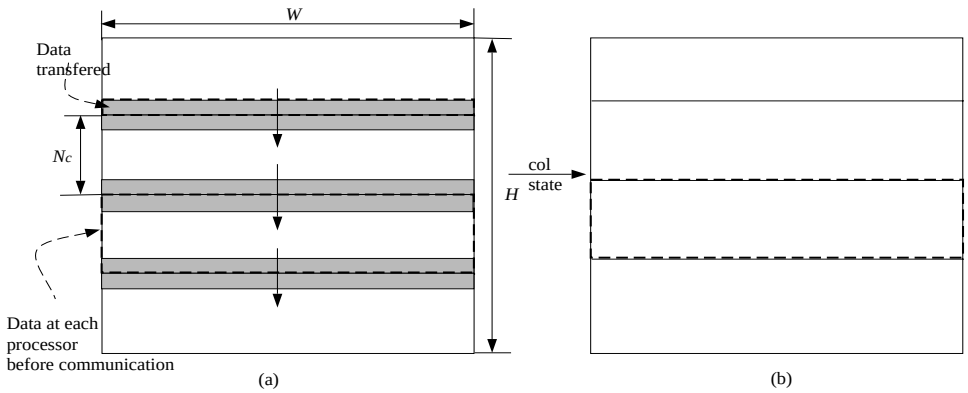



Figure 2.22: *Merge* operations for strip parallel implementation. (a)transfer row state information from $P_i$ to $P_{i+1}$; and (b)complete transforms for boundary samples in each processor.

## 2.6 Experimental Results

In this section, experimental results are provided to show the computation reduction using the *Delayed Normalization* technique in sequential lifting algorithms. Results are also given for the parallel DWT system using the *Overlap-State* technique. The wavelet filterbank used is the Daubechies (9,7) filters. The input image is of size 512x512.

### 2.6.1 Delayed Normalization

In this experiment, three DWT algorithms using the $(9, 7)$ filters are implemented.

1. The recursive standard algorithm (see Table 2.3.1). The computation cost is 11.5 mults/adds per output point.

2. Lifting DWT algorithm. The computation cost is 7 mults/adds per output point.

3. Lifting DWT algorithm which delays the normalization until the last level of decompositions. The computation cost is approximately 6 mults/adds per output point.

In the experiment, 2D separable wavelet transforms are implemented. The algorithms are tested on a ULTRA-1 SUN workstation with clock speed $133MHz$. The algorithm running CPU time is measured using the $clock()$ function call in C. The average CPU time over 50 running instances of each algorithm are listed in Table 2.6.1. To compare the performances, the standard algorithm is used as the basis and the relative speedup is calculated as $T_{standard}/T_{new} - 1$.

Two observations can be seen from the experiment results. One is that the lifting scheme coupled with delayed scaling can have about 30% improvement over the standard algorithm for over three-level decompositions while lifting alone only gives about 20% improvement. Second, neither lifting algorithms achieve the performance gain as predicted in Table 2.4.4. The second observation actually tells us that the number of multiplications/additions in a algorithm is not the only factor contributing to the total DWT running time. The algorithm speed may also be affected by how efficiently the C-code is written and the memory usage too. Obviously, this is a very important factor to consider when building a real DWT system beyond that of reduction of numbers of multiplications and additions.

Table 2.8: DWT CPU time of different sequential algorithms (in seconds).

| Level | Standard | Lifting | | Lifting with Delayed Normalization | |
|---|---|---|---|---|---|
| | | time | speedup | time | speedup |
| 1 | 0.2398 | 0.2088 | 15% | 0.1980 | 21% |
| 2 | 0.2887 | 0.2466 | 17% | 0.2294 | 26% |
| 3 | 0.2988 | 0.2527 | 18% | 0.2303 | 30% |
| 4 | 0.3035 | 0.2598 | 17% | 0.2368 | 28% |
| 5 | 0.3098 | 0.2601 | 19% | 0.2375 | 30% |

### 2.6.2 Strip Parallel

In this experiment, three different parallel DWT algorithms are implemented and tested against a sequential DWT algorithm.

1. Sequential lifting algorithm.

2. Each processor computes the DWT using the standard algorithm. Data exchanges between processors follow the direct overlapping technique, i.e., processors exchange data at each level of decompositions [11, 24].

3. Each processor computes the DWT using the fast lifing algorithm. Data exchanges between processors follow the direct overlapping technique, i.e., processors exchange data at each level of decompositions [11, 24].

4. Each processor computes the DWT using the fast lifting algorithm. Data exchanges between processors follow the proposed *overlap-state* technique.

The first issue in parallel system designs is how to allocate data to different processors. In this experiment, the strip partition strategy [11] is adopted for its simplicity and its appropriateness for the parallel system used in the experiment. The 512x512 image is segmented into two strips with size 256x512, each of which is loaded into one machine for transform. The parallel platform is LAM 6.1 from Ohio Supercomputer Center, which runs over Ethernet connected SUN ULTRA-1 workstations. Two workstations are used to simulate a parallel system with two processors. The algorithm running time is measured using the $MPI\_Wtime()$ function call from MPI libraries. The C-code algorithm is shown in Table 2.6.2. The relative speedup is calculated against the sequential lifting algorithm as $T_{seq}/T_{para} - 1$. The algorithms are tested in 50 running instances and the average DWT running times for different decomposition levels are given in Table 2.6.2.

It can be seen from the results that our proposed parallel algorithm can significantly reduce the DWT computation time even compared with the fastest available parallel algorithm, parallel lifting algorithm. Notice that the improvement is not linear with the increase of the decomposition level. The reason is that, though the communication overhead increases with the decomposition level, the total numerical computation also increases. Another interesting observation is that, even at one level decomposition the proposed algorithm still outperforms the parallel lifting algorithm. This is because though two algorithms both require one data exchange between processors, the amount of data exchanged is different. For the (9,7) filters, the proposed algorithm only needs to exchange approximately

Table 2.9: The proposed parallel DWT algorithm.

```
MPI_Barrier( MPI_COMM_WORLD);
start = MPI_Wtime();
begin{ transform in processor p}
   for( j = 0; j < J; j + +)
      {
         transform at current level j.
         store state information.
      }
   send state information to processor p + 1;
   receive state information from processor p − 1;
   for( j = 0; j < J; j + +)
      {
         transform boundary data samples at current level j.
      }
end
MPI_Barrier(MPI_COMM_WORLD);
finish=MPI_Wtime();
cputime=finish-start;
```

half amount of that necessary in the parallel lifting algorithm.

## 2.7 Conclusions

In this chapter, an *overlap-state* technique is proposed for multilevel wavelet decompositions. The basic idea is to model DWT as a *finite state machine* using the factorization of the polyphase matrix. In this model, each raw input data sample (initial state) is progressively updated into a wavelet coefficient (final state) with the help of samples in its neighborhood. The benefit of such a DWT/FSM model is that the transform (or state transition) of each sample can be stopped at any intermediate stage as long as the state information at the break point is preserved. Since the state information rather than the raw data samples needs to be stored or communicated, we have shown that this helps to reduce the buffer size in a sequential architecture and the communication overhead in a parallel architecture.

Table 2.10: DWT running time of different parallel algorithms (in seconds).

| Level | Sequential | Parallel Standard | | Parallel Lifting | | Parallel Proposed | |
|---|---|---|---|---|---|---|---|
| | | time | speedup | time | speedup | time | speedup |
| 1 | 0.3638 | 0.3115 | 17% | 0.2745 | 33% | 0.2045 | 78% |
| 2 | 0.3649 | 0.3275 | 11% | 0.2899 | 26% | 0.2338 | 56% |
| 3 | 0.3952 | 0.3490 | 13% | 0.2938 | 34% | 0.2369 | 67% |
| 4 | 0.4028 | 0.3513 | 15% | 0.3041 | 34% | 0.2383 | 69% |
| 5 | 0.4041 | 0.3675 | 9% | 0.3165 | 28% | 0.2417 | 67% |

Detailed analysis on buffer size calculation for a given factorization and communication overhead reduction are also provided. To further reduce the computations, a *delayed normalization* technique for multilevel wavelet decompositions is also presented.

Using the *overlap-state* technique, new sequential and parallel DWT architectures are designed. Several system variations for 2D separable DWT are provided and analyzed in detail, which include DWT systems of strip sequential, block sequential, random sequential, block parallel and strip parallel. The performance analyses and the experimental results have shown that the proposed sequential architecture requires less memory and runs faster than existing sequential algorithms. The proposed parallel architecture reduces the interprocessor communication overhead by reducing the number of communication times and the amount of data exchanged. As a result, the DWT running time of the proposed parallel architecture is faster than the best parallel algorithm available, the parallel lifting algorithm.

One important advantage of traditional overlapping techniques, *overlap-add* and *overlap-save*, is that they are well suited for the fast implementations using FFT. Naturally, further research is needed to search for fast DWT algorithms compatible with the proposed *overlap-state* technique. It would increase the chances of a the wide application of the wavelet transform if this can be achieved.

# Chapter 3

# Constrained Transform Design For Multiple Description Coding

Last chapter we studied efficient DWT architectures in a transform coding system. The problem there is simple in the sense that the transform is given and the only thing needs to be done is how to compute the transform efficiently. In this chapter, we raise the problem to a higher level that how to design and compute an *unknown* transform efficiently. Specifically, we study what is *the* transform and how to compute it efficiently if the input data not only needs to be compressed and but also needs to be delivered robustly to the receiver [1].

## 3.1 Introduction

Finding a *good* transform has long been a key issue for various transform coding system designs. Traditionally, a *good* transform is defined to be the one which can maximally decorrelate the input data, i.e., removing the redundancy and thus achieving maximum energy/data compaction. However, to achieve overall performance optimization for data compression and communication over an unreliable channel (e.g., the mobile wireless channel or the best-effort network), maximum decorrelation may not always be the best choice.

[1] Part of this chapter represents work published before, see [60, 61].

Consider the case when the communication channel is not perfect, i.e., the encoded bitstream may arrive at the receiver with error (refer to Fig. 1.1). Whenever this happens, the received bitstream will not be decoded correctly. As a result, the receiver will not be able to recover all the transform coefficients. Neither can it estimates lost coefficients using received ones since the transform has removed the correlation between output coefficients. If lost coefficients are non-principle components (those with small variances), then the end-to-end reconstructed distortion is small. However, suppose principle components (those with large variances) are lost, then the distortion will be high. For communications over unreliable channels, such a quality variation in the received signal can be vary annoying.

Conventionally, channel coding is applied in such cases for error protection and recovery. In this chapter, however, we study an alternative way for error recovery by redesigning the transform to introduce correlation between the transmitted coefficients, a technique of multiple description transform coding (MDTC) proposed recently by Wang et al. [62], Orchard et al. [63] and Goyal et al. [64, 65, 66]. A complete MDTC system is shown in Fig. 3.1 [67]. The input data is first decorrelated using $\mathbf{T}_1$ as that in a conventional transform coding system (see Fig. 1.1). However, quantization coefficients are correlated using another transform $\mathbf{T}_2$. After that, the recorrelated data is encoded into two different bitstreams, called descriptions in MDTC terminology, which are sent through different channels for transmission. If channel failures result in data loss, receiver can now estimate lost coefficients from received ones since there exists correlation between them. As such, the end-to-end reconstruction distortion can be reduced compared to the case when no correlation exists.

Clearly, for such a multiple description transform coding system, its coding efficiency will be lower than that of the system shown in Fig. 1.1 due to the correlation introduced by $\mathbf{T}_2$, i.e., extra bits are needed to encode the correlated output from $\mathbf{T}_2$. The design goal of a MDTC system then focuses on the problem of searching for an optimal correlating transform $\mathbf{T}_2$ which can achieve error recovery at minimum possible redundancy.

For a pair of Gaussian random variables with two output channels, the optimal transform has been provided analytically by Goyal et al. [64] with one

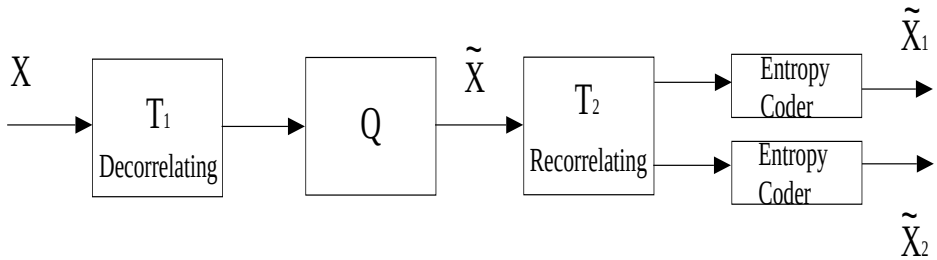



Figure 3.1: A typical MDTC system

special case also reported by Orchard et al. [63]. They have also shown that non-orthogonal correlating transforms perform better compared to orthogonal correlating transforms in terms of redundancy rate-distortion gain, though, in this case the transform itself has to be invertible (mapping integers to integers). For MDTC systems with $M$ inputs and $M$ outputs, the optimal transform design and its performance analysis is still an open problem, though near-optimal solutions for 3 and 4 channels have been given by Goyal et al. [64]. Orchard et. al [63] suggested a redundancy allocation strategy among pairs of input variables but optimal pairing is not yet readily available. A numerical optimization algorithm was proposed by Goyal et al. [64] to design transforms for arbitrary number of channels. However, exhaustive search through the whole space of all non-orthogonal transforms is not only computationally intensive but also leads to implementation difficulties when using an arbitrarily structured non-orthogonal transform.

In this chapter, we first address the problem of correlating transform design, i.e., designing $\mathbf{T}_2$ (refer to Fig. 3.1) to optimize the operational redundancy rate-distortion performance. The approach we propose is a two-stage transform design technique: separating the design into (i) structure design and (ii) magnitude design. The observation is that error protection properties of a MDTC system can be fully characterized by the output correlation matrix, i.e., the correlation matrix of coefficients generated by $\mathbf{T}_2$. Given the output correlation matrix, one can immediately see which descriptions are correlated (structure) and to what extent they are correlated (magnitude). While the magnitude information of the correlation matrix can not, in general, be quantified for specific redundancy

rate distortion constraints, the structural information can sometimes be directly inferred if specific channel conditions or protection requirements are provided.

For example, one common technique for error protection used in the robust audio tool (RAT) over lossy packet networks is to have each packet protect its previous packet or vice verse [29]. In this case, if a MDTC system is to be designed, one possibility is to have a band-diagonal correlation matrix with each coefficient correlated only with neighboring coefficients. Packing each coefficient into one packet and send these packets sequentially over the network, a similar error protection scheme as that of RAT is completed. In this case, the key to the transform design is to first find all transforms (admissible transforms) which can generate a band-diagonal correlation matrix and then search through the admissible transform set for the optimal solution. As we will show, the admissible transform is simply the eigenmatrix of the output correlation matrix. If the structural information is available, such as the band-diagonal structure in this case, the transform can often be pre-designed taking advantage of existing results in the area of decorrelating transform design. Thus the structural information of the output correlation matrix can be used to find all admissible transforms (eigenmatrices of the output correlation matrix). To meet the final redundancy rate-distortion constraints, the optimization algorithm similar to that in [64] can then be applied to complete the magnitude design.

The major advantage of this two-stage design approach is that it can enforce a structure on the transform to be designed. For optimal redundancy rate-distortion performance, the transform design previously has to search through all non-orthogonal transforms, i.e., starting with an arbitrary non-orthogonal transform, as that been shown by Goyal et al. [64]. If, however, the transform structural information can be derived from the available channel information, the search space can be drastically reduced. As a result, the complexity of the optimization algorithm can be drastically reduced. The enforced structure in the design phase also leads to *structured* optimal transforms, which can often be implemented via existing fast algorithms as will be shown later.

As the second part of this chapter, we address the problem of designing a single transform which can both decorrelate and recorrelate the input at the same time.

As one can see from Fig. 3.1, the MDTC system configuration is itself *redundant*, i.e., the input is first decorrelated by $\mathbf{T}_1$, and then recorrelated by $\mathbf{T}_2$. To reduce the system implementation complexity, we propose to replace both $\mathbf{T}_1$ and $\mathbf{T}_2$ with a single transform, the Karnunen-Loeve vector transform (KLVT). The proposed KLVT can take as input a group of vectors and generate decorrelated vectors while preserving correlation between vector components in each vector. Each of these vector components can be grouped to form one description and sent through different channels. The error recovery mechanism in case of channel failures is the same as that in a MDTC system (refer to Fig. 3.1), however, the system can now be configured as simply as that in Fig. 1.1. The idea of KLVT is very similar to the vector transform (VT) proposed by Li at al. [68, 69, 70], which is used to preprocess the data for vector quantization. To maximize the coding efficiency, both KLVT and VT are required to maximally remove the inter-vector correlation. However, an optimal VT is also required to maximally preserve the intra-vector correlation for compression while KLVT can be designed to vary the intra-vector correlation based on channel statistics for loss data recovery.

The remainder of this chapter is organized as follows. In the next section, a brief review on MDTC is provided and the problem of optimal correlating transform design is defined. In Section 3.3, we propose a two-stage transform design approach based on parametric scaling-rotation transforms for the design of MDTC systems with $M$ inputs and $M$ outputs. Section 3.4 provides two correlating transform design examples for equal rate channels and sequential protection channels, respectively. Simulation results of Gaussian vectors are also presented. In Section 3.5, the proposed Karhunen-Loeve vector transform (KLVT) is defined together with analysis of possible applications for the MDTC system design. Finally, we conclude our work in Section 3.6.

## 3.2 Problem Definition

Assume for a source $\mathbf{X}$, $M$ descriptions $\{C_i, i = 1, 2, \cdots, M\}$ are generated. Notations are introduced following that in [63].

**Central Distortion** $D_c$ The average reconstruction error when all $M$ descriptions are used.

**Side Distortion** $D_s$ The average reconstruction error when only a subset of $M$ descriptions are used.

**Redundancy** $\rho$ The difference between the actual coding rate $R$ and $R^* = R_X(D_c)$, the source rate-distortion function evaluated at $D_c$.

**Redundancy Rate-Distortion Function** $\rho(D_s; D_c)$ The amount of extra bits of redundancy necessary to achieve a desired side distortion $D_s$ at a given central distortion $D_c$.

For an input source vector $\mathbf{X}$, the encoding procedure of a MDTC system shown in Fig. 3.1, as described in [63, 64], is

1. $\mathbf{X}$ is decorrelated by $\mathbf{T}_1$.
2. The transform coefficients generated by $\mathbf{T}_1$ are quantized with a uniform scalar quantizer.
3. The quantized vector $\tilde{\mathbf{X}}$ is transformed with an invertible, discrete transform $\mathbf{T}_2$, which introduces correlation among vector components of $\tilde{\mathbf{X}}$.
4. The components of the transform output vector, $\{\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2\}$, are independently entropy coded.
5. The encoded bitstreams are separated from each other and sent over different channels.

Assume that the correlation information is delivered to the decoder correctly. The decoding procedure is

1. Entropy decode all received bistreams.
2. (a) If all descriptions are received, then the inverse transform $\mathbf{T}_2^{-1}$ is applied. The inverse transformed data is then dequantized.

(b) If only a subset of descriptions are received, the received data is dequantized first. The lost descriptions are then estimated from available descriptions using the correlation information received from the encoder. The reconstructed vector (including received and reconstructed descriptions) is inverse transformed by $\mathbf{T}_2^{-1}$.

3. The output from previous stage is then inverse transformed by $\mathbf{T}_1^{-1}$ to get the reconstruction $\hat{\mathbf{X}}$. The final end-to-end reconstruction distortion $D = E||X - \hat{X}||^2$ is the central distortion $D_s$ for case 2(a) and side distortion $D_c$ for case 2(b).

The multiple description coding problem can be formulated as

**Objective** *Design transform* $\mathbf{T}_2$ *to minimize the side distortion* $D_s$ *as well as the central distortion* $D_c$ *subject to redundancy constraint* $\rho$.

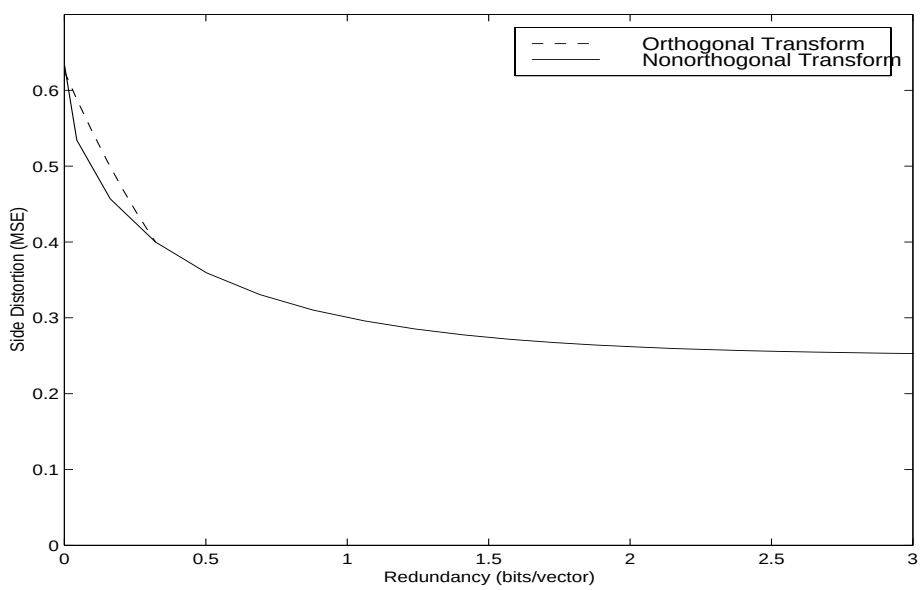



Figure 3.2: Redundancy rate distortion curve of a MDTC system for Gaussian vector with $\sigma_1 = 1, \sigma_2 = 0.5$.

For MDTC of pairs of independent Gaussian random variables, an important result is that non-orthogonal transforms are better than orthogonal transforms in

terms of the redundancy rate-distortion gain [63, 64]. For a given central distortion, the non-orthogonal transform not only can achieve lower average side distortion using the same amount of extra bits, it can also extend the redundancy rate-distortion function to the region where the orthogonal transform cannot reach, see Fig. 3.2. However, non-orthogonal transforms, if used for recorrelation, pose three challenges to the MDTC system design:

1. *Lossless Implementation*: The transform has to be lossless for efficient quantization. For finite precision implementation, this means that the transform has to be an integer mapping, i.e., mapping integers to integers.

2. *Design Complexity*: The numerical optimization algorithm by Goyal et al. [64] becomes computationally intensive with the increase of the dimensionality of the transform necessary. For a $M$-channel MDTC system, the transform has to be of size $M$x$M$ and each of its entry is a design parameter (complexity of $O(M^2)$.

3. *Implementation Complexity*: The optimal transform can have arbitrary structure in practice, which makes fast implementation difficult.

In the next section, we start with an intuitive geometric explanation on why non-orthogonal transforms can perform better than orthogonal transforms. We then propose a structured non-orthogonal transform framework for optimal correlating transform design.

## 3.3 Proposed Design Approach

### 3.3.1 Geometric Interpretations

For two equal rate channels, the optimal pairing transform $\mathbf{T}_a$ given by Goyal and Kovacevic [64] is of the form

$$\mathbf{T}_a = \begin{bmatrix} a & 1/(2a) \\ -a & 1/(2a) \end{bmatrix} \tag{3.1}$$

With a little math, the following can be derived.

$$\mathbf{T}_a = \begin{bmatrix} a & 1/(2a) \\ -a & 1/(2a) \end{bmatrix} \tag{3.2}$$

$$= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{2}a & 0 \\ 0 & 1/(\sqrt{2}a) \end{bmatrix} \tag{3.3}$$

As one can see, the optimal pairing transform is nothing but a concatenated scaling-rotation transform. Obviously, the rotation transform, orthogonal by itself, can introduce correlation for uncorrelated inputs. The use of the scaling transform further enhances its ability to do so.

Write the optimal transform in parametric form $\mathbf{T}_{a,\theta}$ as

$$\mathbf{T}_{a,\theta} = \underbrace{\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}}_{rotation} \underbrace{\begin{bmatrix} a & 0 \\ 0 & 1/a \end{bmatrix}}_{scaling} \tag{3.4}$$

$$= \begin{bmatrix} a\cos\theta & \sin\theta/a \\ -a\sin\theta & \cos\theta/a \end{bmatrix} \tag{3.5}$$

Denote the original orthogonal basis vectors as $\{\mathbf{u_1}, \mathbf{u_2}\}$ and the new basis vectors under the transform $\mathbf{T}_{a,\theta}$ as $\{\mathbf{v_1}, \mathbf{v_2}\}$. One can write

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = \mathbf{T}_{a,\theta} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} a\cos\theta\,\mathbf{u}_1 + (1/a)\sin\theta\,\mathbf{u}_2 \\ -a\sin\theta\,\mathbf{u}_1 + (1/a)\cos\theta\,\mathbf{u}_2 \end{bmatrix} \tag{3.6}$$

The inner product between $\mathbf{v}_1$ and $\mathbf{v}_2$ is

$$< \mathbf{v}_1, \mathbf{v}_2 > = (1/2)\sin 2\theta(-a^2 + 1/a^2) \tag{3.7}$$

It is easy to see that the inner product of the new basis vectors is equal to zero (new basis is still orthogonal) whenever the scaling factor $a = 1$. In this case, $\mathbf{T}_{a,\theta}$ is equivalent to an orthogonal transform. Since an orthogonal transformation amounts to a plane or hyperplane rotation, it can not introduce any correlation for the case when all eigenvalues of the input vector correlation matrix

are equal. Geometrically, such a random vector does not have any directional preference among all eigenvectors, and the joint distribution will be circular symmetric or hyper-spherical in higher dimensions. However, if the scaling factor $a$ is chosen to be either smaller or larger than 1, the inner product will become nonzero (assuming $\theta$ is nonzero) and the new basis will become correlated. In this case, even if the input vector components all have same variances, they can still become correlated after such a scaling-rotation transform. This scaling-rotation structure partially explains that non-orthogonal transforms have greater flexibility to introduce correlation than orthogonal transforms.

In Fig. 3.3, the data distribution plots are shown for a pair of independent Gaussian random variables under different transforms. It can be seen clearly that the scaling-rotation transform introduce stronger correlation than its orthogonal counterpart.

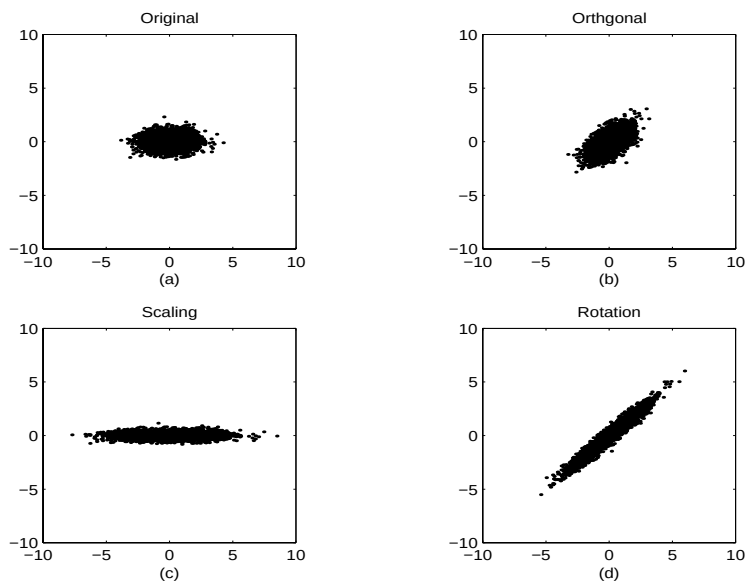



Figure 3.3: Distributions of two independent Gaussian variables with variance $\sigma_1 = 1, \sigma_2 = 0.5$. (a) original; (b) after 45° rotation with correlation coefficient 0.6. (c) after scaling($a = 2$); (d) after scaling and 45° rotation with correlation coefficient 0.94. Clearly one can see that data in (d) is more correlated than that in (b).

### 3.3.2 Parametric Transform and Factorizations

The decomposition of the optimal paring transform into a scaling-rotation framework is not accidental but reflects a general structure of non-orthogonal transforms. Indeed, any matrix can be factored into a product of an orthogonal matrix and an upper triangular matrix, the so-called QR decomposition in matrix theory [71]. Using our notations we write this as $\mathbf{T} = \mathbf{RU}$ where $\mathbf{R}$ is an orthogonal transform and $\mathbf{U}$ is an upper triangular matrix. The upper triangular matrix $\mathbf{U}$ can be further decomposed as a product of a scaling matrix $\mathbf{S}$ and another upper triangular matrix $\mathbf{L}$ (with diagonal entries all 1). In other words, any transform can be written as a concatenation of three transforms, an upper triangular transform $\mathbf{L}$, a scaling transform $\mathbf{S}$ and a rotation transform $\mathbf{R}$ (orthogonal transform).

$$\mathbf{T} = \mathbf{RSL} \tag{3.8}$$

Note that $\mathbf{T}$ can also be a non-square transform which is the case when frame expansions is used for adding redundancy [66].

To reduce the design complexity, in this work, we only study non-orthogonal square transforms with scaling-rotation factorization, i.e., $\mathbf{T}_{rs} = \mathbf{RS}$. To use an non-orthogonal transform for MDTC, the first constraint is *lossless implementation* for efficient quantization [64]. To this end, we further require that the determinant of $\mathbf{T}_{rs}$ to be 1, i.e., $det(\mathbf{T_{rs}}) = det(\mathbf{RS}) = det(\mathbf{S}) = 1$.

We now show that any such transform $\mathbf{T}_{rs}$ can be factored into lifting steps and thus can be implemented losslessly. We first present factorization results for 2x2 rotation and scaling transforms (adapted from [48]) as follows.

$$\begin{bmatrix} a & 0 \\ 0 & 1/a \end{bmatrix} = \begin{bmatrix} 1 & a-a^2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1/a & 1 \end{bmatrix} \begin{bmatrix} 1 & a-1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{3.9}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \frac{(\cos\theta-1)}{\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{(\cos\theta-1)}{\sin\theta} \\ 0 & 1 \end{bmatrix} \tag{3.10}$$

These results show that basic 2x2 rotation and scaling transforms can be implemented losslessly.

For $M$x$M$ rotation transforms, it is well known that any orthogonal $M \times M$ matrix $Q_M$ can be factored into the product of $M(M-1)/2$ orthogonal matrices, each of which is a $M \times M$ Givens rotation that only rotates two components at a time. [71] The factorization of a Givens rotation matrix is equivalent to that of the basic 2x2 rotation. Therefore, any $M$x$M$ orthogonal transform can be implemented losslessly as an integer mapping.

Factorization of a scaling transform is also straightforward using the factorization result of the basic 2x2 scaling transform. One can show that any $M$x$M$ scaling transform can be written as a product of $M-1$ subscaling transforms, each of which only scales two components at a time. As an example, we show the decomposition of a 4x4 scaling transform as follows.

$$\begin{aligned}
\mathbf{S}(\mathbf{A}) &= \begin{bmatrix} a_0 & 0 & 0 & 0 \\ 0 & a_1 & 0 & 0 \\ 0 & 0 & a_2 & 0 \\ 0 & 0 & 0 & a_3 \end{bmatrix} \\
&= \begin{bmatrix} a_0 & 0 & 0 & 0 \\ 0 & 1/a_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a_0a_1 & 0 & 0 \\ 0 & 0 & 1/(a_0a_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a_0a_1a_2 & 0 \\ 0 & 0 & 0 & 1/(a_0a_1a_2) \end{bmatrix}
\end{aligned}$$

Note the constraint of $det\mathbf{S} = 1$ is applied, i.e.,$\prod_{i=0}^{M} a_i = 1$. Therefore $a_3 = 1/(a_0a_1a_2)$.

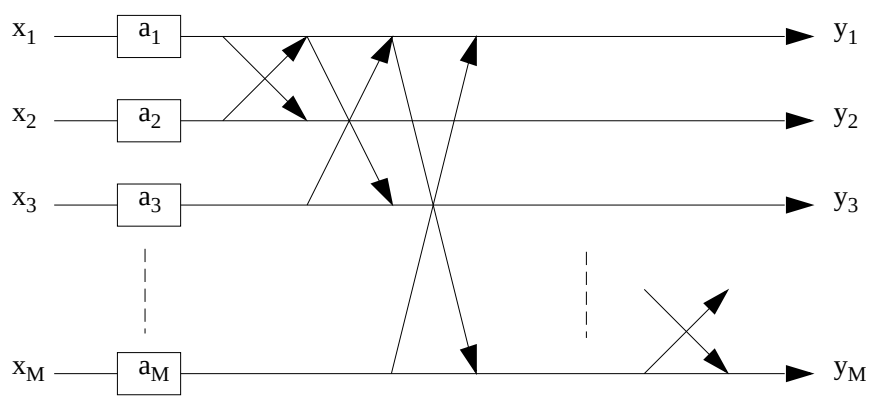



Figure 3.4: Lattice structure of a **RS** transform

Thus we have shown that any $\mathbf{T}_{rs}$ can be factored into lifting steps and therefore can be implemented losslessly. The general implementation structure of $\mathbf{T}_{rs}$ is shown in Fig. 3.4, which has a lattice structure similar to that of a biorthogonal filterbank. The perfect reconstruction property is guaranteed by such a transform framework. To derive the inverse transform, one only needs to change all the rotation angles to the opposite sign and multiply by the inverses of the scaling factors.

Based on our analysis in previous section, we propose to use the parametric form of a $\mathbf{T}_{rs}$ for the design of MDTC. The parametric form of $\mathbf{T}_{rs}$ can be written as

$$\mathbf{T}_{rs}(\mathbf{A}, \mathbf{\Theta}) \quad = \quad \mathbf{R}(\mathbf{\Theta})\mathbf{S}(\mathbf{A}) \tag{3.11}$$

where $\mathbf{A}$ are scaling factors and $\mathbf{\Theta}$ are Givens rotation angles. As analyzed before, the scaling transform changes the relative energy distribution of the input vector components while the rotation introduces correlation among vector components. The net effect of such an operation is that the orthogonal basis is transformed into a non-orthogonal one. As a result, more correlation can be introduced among vector components via decomposition to a non-orthogonal basis rather than an orthogonal basis.

### 3.3.3 Two Stage Transform Design

Although $\mathbf{T}_{rs}$ enjoys structured lattice implementation, the number of design parameters of an $M \times M$ transform still increases quadratically ($O(M^2)$) for a $M$-Dimensional input vector. In this section we propose a two-stage design technique making use of the available channel information to further constrain $\mathbf{T}_{rs}$ and reduce both the design and implementation difficulties.

Recall that the transform to be designed is a recorrelating transform (equivalent to $\mathbf{T}_2$ as shown in Fig. 3.1). Denote the uncorrelated input to $\mathbf{T}_{rs}$ as $\mathbf{X}$ and the correlated output as $\mathbf{Y}$. Let $\mathbf{R}_X = diag\{\sigma_{ii}^2\}, i = 1, 2, \cdots, M$ be the input correlation matrix and $\mathbf{R}_Y = \{r_{ij}\}, i, j = 1, 2, \cdots, M$ the output correlation

matrix of the transform output $Y$. Then we have

$$\mathbf{R}_Y = \mathbf{T}_{rs}(\mathbf{A}, \mathbf{\Theta})\mathbf{R}_X\mathbf{T}_{rs}^t(\mathbf{A}, \mathbf{\Theta}) \tag{3.12}$$

$$= \mathbf{R}(\mathbf{\Theta})\mathbf{R}_S\mathbf{R}^t(\mathbf{\Theta}) \tag{3.13}$$

Since $\mathbf{R_S} = \mathbf{S}(\mathbf{A})\mathbf{R_X}\mathbf{S}(\mathbf{A})^{\mathbf{t}} = \{a_{ii}^2\sigma_{ii}^2, i = 1, 2, \cdots, M\}$ is still a diagonal matrix, the rotation transform $\mathbf{R}(\mathbf{\Theta})$ has to be the eigenmatrix of the output correlation matrix $\mathbf{R}_Y$. If the output correlation matrix can be predesigned, then the problem of correlating transform design can be formulated as the inverse of the of matrix diagonalization problem. An orthogonal solution to the correlating transform is simply the inverse of KL transform. In general, we cannot pre-design the output correlation matrix subject to redundancy and distortion constraints. However, we can select its structural information for specific channel conditions or protection requirements. For example, equal rate channels require the output correlation matrix to have equal diagonal entries for Gaussian inputs. Thus admissible transforms have to be able to generate correlation matrices with equal diagonal entries.

Based on such observations, we propose a two stage transform design approach, i.e., structure design and magnitude design. The structure design finds admissible transforms (eigenmatrices of the output correlation matrix) for specific channels using the $\mathbf{T}_{rs}$ factorization framework. In Fig. 3.5 we show that the transform search space can be reduced gradually using available channel information. Start from the transform space which consists of all transforms, either orthogonal or non-orthogonal, with determinant one. We reduce the search space by enforcing that all the transforms must have a scaling-rotation factorization. This will reduce the complexities of both the design and implementation as described before. From this scaling-rotation transform space, application specific constraints can be imposed to further reduce the space for admissible transforms (details in next section).

After the structure of the transform is found, the magnitude design then searches for the optimal transform from admissible transforms using the algorithm described in [64] where derivation details of the average side distortion

$D_s(\mathbf{A}, \mathbf{\Theta})$ and the redundancy bit rate $\rho(\mathbf{A}, \mathbf{\Theta})$ are given. A different derivation of this optimization algorithm is given in Appendix C. We perform a redundancy constrained transform design using a Lagrangian multiplier $\lambda$. The cost function is $J = D_s(\mathbf{A}, \mathbf{\Theta}) + \lambda\rho(\mathbf{A}, \mathbf{\Theta})$. By varying $\lambda$, one can scan all the operational redundancy rate distortion points $(D_s, \rho)$.

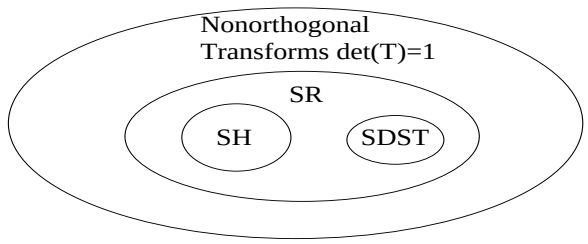



Figure 3.5: Transform search space.

## 3.4 MDTC Design Examples

We give design examples for two important channels, equal rate channels and sequential protection channels, both of which can be characterized by the output correlation matrix $R_Y$. It turns out that for these two special channels, not only can we use fixed rotation transforms, but also these fixed transforms also have fast algorithms. Using a fixed rotation transform, we can reduce the number of design parameters from $M-1(\mathbf{A}) + M(M-1)/2(\mathbf{\Theta}) = (M^2+M-2)/2$ to only $M-1$. This makes the optimization converge faster and reduces the amount of information to be conveyed to the decoder. On the other hand, fast algorithms reduce both the encoding and decoding complexities.

### 3.4.1 Equal Rate Channels

The equal rate channels case requires that output descriptions have the same rates, which helps the buffer management (e.g. packetization/depacketization in a packet network) both at the encoder and the decoder. As stated before, *equal rate* should be interpreted in a statistical sense. For example, two Gaussian sources with same variances will be viewed as equal rate sources if quantized

with the same quantizer. That is to say, for Gaussian random variables, "equal rate" requires that the correlation matrix $\mathbf{R}_Y$ have equal diagonal entries. Here we provide a $\mathbf{T}_{rs}$ transform which generates equal rate descriptions for arbitrary number of channels with $M = 2^n$.

The equal rate transform we propose is a scaling Hadamard transform.

$$\mathbf{T}(\mathbf{A}) \quad = \quad \mathbf{HS}(\mathbf{A}) \tag{3.14}$$

where $H$ is the Hadamard transform. We need to show that, under this transform, for input $\mathbf{X}$, $\mathbf{R_Y}$ has equal diagonal entries (equal variances), i.e., $r_{ii} = r, i = 1, 2, \cdots, M$ and $r$ is a constant.

Since the Hadamard transform $\mathbf{H}$ is real symmetric, the output correlation is $\mathbf{R_Y} = \mathbf{HR_SH}$. Denote $\mathbf{H} = \{h_{ij}\}, i, j = 1, 2, \cdots, M$. We have $h_{ji}^2 = 1, \forall(i, j)$. The output $\mathbf{Y}$ component variance $r_{ii}$ can be written as

$$r_{ii} = \sum_{j=1}^{M} a_{jj}^2 \sigma_{jj}^2 h_{ji}^2 = \sum_{j=1}^{M} a_{jj}^2 \sigma_{jj}^2$$

Thus we have shown a scaling-Hadamard (SH) transform generates equal rate descriptions.

However, correlation coefficients between output vector components, i.e., $r_{ij}, i \neq j$, have to be jointly designed with the scaling transform to meet the rate-distortion requirements via the optimization algorithm given in Appendix C. Nonetheless, the scaling-Hadamard structure gurantees equal rates output and it also reduces the number of design paremeters from $M^2$ (an abitrary non-orthogonal $M$x$M$ transform) to $M - 1$ ($M - 1$ scaling factors required in the scaling-Hadmadard transform). We mention that this scaling-Hadamard transform reduces to the optimal equal rate transform given by Goyal and Kovacevic [64] when $M = 2$, which demonstrates that, at least for two descriptions coding the $\mathbf{RS}$ transform does not compromise the optimality of the MDTC system. We also note that the cascaded structure given by Goyal et al. [64] is equivalent to a Scaling Hadamard transform for $M = 4$.

### 3.4.2 Sequential Protection Channels

Another example channel is the sequential protection channel in which descriptions are sent out sequentially and each description will only protect the losses of its immediate predecessor and its immediate successor. In a lossy packet network, an example scenario is that when each packet carries information for the recovery of its previous and next packets, e.g. a similar case when Robust Audio Tool technique is applied for audio transmission [29]. For a MDTC system, this indicates that the output correlation matrix $\mathbf{R_Y}$ should be a tridiagonal matrix in which descriptions are sequentially correlated. The $\mathbf{R_Y}$ assumes the form

$$\mathbf{R_Y} = \begin{bmatrix} 1 & \rho & 0 & 0 & \cdots & 0 \\ \rho & 1 & \rho & 0 & \cdots & 0 \\ 0 & \rho & 1 & \rho & \cdots & 0 \\ & \vdots & & \vdots & \vdots & \\ 0 & \cdots & 0 & \rho & 1 & \rho \\ 0 & \cdots & 0 & 0 & \rho & 1 \end{bmatrix} \tag{3.15}$$

From matrix theory, we know that the eigenmatrix for this type of symmetric tridiagonal Toeplitz matrices is the Discrete Sine Transform (DST) [72]. So the transform we propose for sequential protection channels is the Scaling-DST transform

$$\mathbf{T}(\mathbf{A}) = DST\ \mathbf{S}(\mathbf{A}) \tag{3.16}$$

### 3.4.3 Simulation Results of Gaussian Sources

In this experiment, we compare results of different configurations of the correlating transform for a 4D Gaussian vector source with standard deviations $\{1, 0.5, 0.3, 0.1\}$ [63]. We compare the side distortion when there is only one description lost with equal channel failure probabilities. The different transforms are *(i) Rotation; (ii) Scaling-Rotation; (iii) Scaling-Hadamard; and (iv) Scaling-DST.* The configuration Scaling-Hadamard is equivalent to the cascaded structure

given by Goyal and Kovacevic (Fig. 3 in [64]) for a 4-D input vector. The optimization is done via Powell's direction set technique [73]. The initial scaling factors are all set to be 1 and the initial rotation angles are all set to be $\pi/4$ for the Scaling-Rotation configuration.

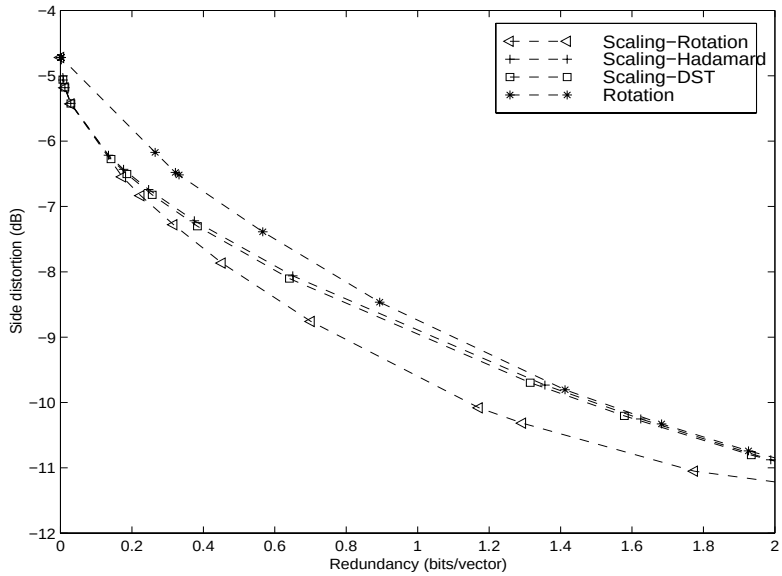



Figure 3.6: Comparisons among different transforms.

The comparison of all four configurations is shown in Fig. 3.6. Clearly, all the non-orthogonal transforms achieve better performances compared to the orthogonal transform (case (i)). This indicates that non-orthogonal transforms can also perform better than orthogonal transforms for MDTC systems of more than two channels. We also observe performance degradations when we impose constraints on the rotation transform, scaling-Hadamard or scaling-DST transform, specially at higher redundancy bit rates. However, structured transforms simplify the design and implementation complexities. In this case, both Hadamard transform and DST can be implemented using their existing fast algorithms.

A drawback of using a non-orthogonal transform is that the complete MDTC system (refer Fig. 3.1) requires two transforms, $T_1$ for decorrelation (before quantization) and $T_2$ for recorrelation (after quantization), both at the encoder and decoder. Obviously, this increases the system implementation complexity. If an orthogonal transform is used, then one can merge $T_2$ with $T_1$ into one transform

to reduce the computation. In the next section, we present one such transform and show possible applications for MDTC system design.

## 3.5 Karhunen-Loeve Vector Transform

Let $\mathbf{X}$ be a vector random sequence of size $N$x1 generated from a stationary and ergodic source with zero mean. Assume each vector in the sequence can be further decomposed into subvectors of size $M$x1, i.e., $\mathbf{X} = \{\mathbf{x_1}, \cdots, \mathbf{x_L}\}$ where $ML = N$. Denote the correlation matrix of $\mathbf{X}$ as $\mathbf{R}$ and the correlation matrix between two subvectors, $\mathbf{x_i}$ and $\mathbf{x_j}$, as $\mathbf{R}_{i,j}$. Then we have the following identities.

$$\mathbf{R}_{i,j} = \begin{bmatrix} E(x_i^1 x_j^1) & \cdots & E(x_i^1 x_j^{M-1}) \\ \cdots & \cdots & \cdots \\ E(x_i^{M-1} x_j^1) & \cdots & E(x_i^{M-1} x_j^{M-1}) \end{bmatrix} \tag{3.17}$$

and

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{1,1} & \cdots & \mathbf{R}_{1,L-1} \\ \cdots & \cdots & \cdots \\ \mathbf{R}_{L-1,1} & \cdots & \mathbf{R}_{L-1,L-1} \end{bmatrix} \tag{3.18}$$

*Definition:* Any unitary transform $\mathbf{T}$ which can block diagonalize the autocorrelation matrix $\mathbf{R}$ is defined to be a KL vector transform (KLVT) for vector random sequence $\mathbf{X} = \{\mathbf{x_i}, 1 \leq i \leq L\}$.

$$\mathbf{TRT}^H = \begin{bmatrix} \mathbf{R}_1 & & & & \\ & \mathbf{R}_2 & & & \\ & & . & & \\ & & & . & \\ & & & & \mathbf{R}_L \end{bmatrix} \tag{3.19}$$

where $\mathbf{R}_i$ is the autocorrelation matrix of transformed subvector $\mathbf{y}_i$ and $\mathbf{Y} = \mathbf{TX} = \{\mathbf{y}_i, i = 1, 2, \cdots, L\}$. The existence of such a transform for any arbitrary inputs is obvious since one can easily see that the Karhunen-Loeve transform is a

special case of KLVT. Actually, KLVT defines a transform set and its properties follow that of KLT.

**KLVT Properties**

*Property 1: KLVT set*

KLVT defines a unitary transform set $\mathbf{S} = \{\mathbf{T} = \mathbf{BK}\}$ which includes all transforms of the form $\mathbf{T} = \mathbf{BK}$ where $\mathbf{K}$ is the KLT for $\mathbf{X}$ and $\mathbf{B}$ can be any arbitrary block unitary matrix as long as the sizes of subblocks in $\mathbf{B}$ agree with the sizes of subblocks in $\mathbf{R}$. So the cardinality of set $S$ is infinity.

If

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & & & & \\ & \mathbf{B}_2 & & & \\ & & . & & \\ & & & . & \\ & & & & \mathbf{B}_L \end{bmatrix}$$

where each $\mathbf{B}_i$ is unitary, and

$$\mathbf{KRK}^H = \begin{bmatrix} \mathbf{\Lambda}_1 & & & & \\ & \mathbf{\Lambda}_2 & & & \\ & & . & & \\ & & & . & \\ & & & & \mathbf{\Lambda}_L \end{bmatrix}$$

then

$$\begin{aligned} \mathbf{TRT}^H &= \mathbf{BKRK}^H\mathbf{B}^H \\ &= \begin{bmatrix} \mathbf{B}_1\mathbf{\Lambda}_1\mathbf{B}_1^H & & & & \\ & \mathbf{B}_2\mathbf{\Lambda}_2\mathbf{B}_2^H & & & \\ & & . & & \\ & & & . & \\ & & & & \mathbf{B}_L\mathbf{\Lambda}_L\mathbf{B}_L^H \end{bmatrix} \end{aligned}$$

is also a block diagonal matrix.

*Property 2: Decorrelation*

Assume $\mathbf{T}$ is any transform from the KLVT set $\mathbf{S}$ and $\mathbf{Y} = \mathbf{TX}$ with $\mathbf{Y} = \{\mathbf{y}_i, 1 \leq i \leq L\}$ and $\mathbf{X} = \{\mathbf{x}_i, 1 \leq i \leq L\}$, then

$$E[\mathbf{y}_i \mathbf{y}_j^H] = \begin{cases} \underline{\mathbf{0}} & i \neq j \\ \mathbf{R}_i & i = j \end{cases} \tag{3.20}$$

where $\mathbf{R}_i$ is diagonal if $\mathbf{T}$ is the KLT of $X$ and arbitrary symmetric otherwise. This property is directly derived from the definition.

*Property 3: Energy Compaction*

Among all the unitary transforms, KLVT transforms pack maximum average energy in $l \leq L$ subvectors of $\mathbf{Y}$. Here the energy of a vector $\mathbf{y}$ is defined to be its mean squared length $E\{|\mathbf{y}|^2\}$. If vector size is 1, then its energy equals to its variance (assuming zero mean).

$$E\{|\mathbf{y}|^2\} = E\{\mathbf{y}^\mathbf{t}\mathbf{y}\} = Tr(E\{\mathbf{y}\mathbf{y}^\mathbf{t}\}) = \sum_{i=1}^{M} \lambda_i \tag{3.21}$$

where $M$ is the size of subvector $\mathbf{y}$ and $\{\lambda_i, i = 1, 2, \cdots, M\}$ are eigenvalues of the autocorrelation matrix $E\{\mathbf{y}\mathbf{y}^\mathbf{t}\}$ ($Tr(\mathbf{T})$ denotes the trace of matrix $\mathbf{T}$.

This property can be easily proved by KLT's maximum energy compaction property. Since each KLVT transform is related to the KLT by a block unitary transform which preserves the eigenvalues [71], the transformed vector will preserve all the energies of the corresponding scalar components in the KLT case. Hence if the scalar components are ordered in KLT case, the accumulative vector energy will also be ordered in the same order after the block unitary transform. To probe further into the KLVT, we provide one more view from vector space partition.

As one can see, KLVT actually defines a transform set $\mathbf{T}$ which includes all the transforms that can block diagonalize the autocorrelation matrix $\mathbf{R}$ hence will decorrelate the corresponding vector signal if applied to the original data. However, this non-uniqueness property of KLVT should not be misinterpreted as the non-uniqueness of the decorrelation vector space partition for a given signal (see

Appendix D). Actually, the decorrelation vector space partition for a given vector signal is unique only up to a permutation of the eigensubspaces corresponding to each of these eigenvalues. For a given autocorrelation matrix, the eigenvalues are uniquely defined. The difference among the transforms is that the residue intra-vector correlation after the transform. At the one end is the KLT transform which removes completely inter-vector correlation as well as intra-vector correlation. There exists, however, an infinite number of transforms which can maximally remove the inter-vector correlation while keeping the intra-vector correlation at some level. This intra-correlation obviously can be used for error protection as that discussed before in a MDTC system. Thus we have shown the possibility to combine decorrelating and recorrelating in one transform for the MDTC system design. Compared to the system configuration shown in Fig. 3.1, the use of KLVT can reduce the design and implementation complexity.

## 3.6 Conclusions

In this chapter, we studied the problem of constrained transform design for robust communication via multiple description transform coding. A two stage transform design approach is proposed for MDTC system design. Such an approach enables us to find structured transform solutions using available channel information. This helps to reduce both the system design and implementation complexities. We also provided example transform designs for equal rate channels and for sequential protection channels. To further reduce the MDTC system complexity, a Kahunen-Loeve vector transform is proposed and possibilities for application in MDTC system designs are illustrated. One future work is to study the redundancy rate-distortion performance of KLVT in a MDTC system.

# Chapter 4

# Multiple Description Coding for Erasure Channels

Last chapter we introduced a correlating transform based coding system which can recover the lost data using the correlation existing in the correctly received data. In this chapter, we take a different approach for loss data recovery, i.e., adding redundancy *explicitly* rather than *implicitly* (as that using the correlating transform in the last chapter) in the encoded data for loss recovery. The motivation is simple that we would like to have a simple system design and implementaion [1].

## 4.1 Introduction

In recent years, a number of approaches toward error control and protection for audio and video communication over packet networks have been reported in the literature [29, 76, 77, 78, 79, 80]. A majority of these works have chosen the receiver-only strategies to avoid retransmission and to reduce the communication delay. These include various FEC schemes (and its variations) and error concealment techniques that exploit the residual correlation in the encoded data.

For example, in Jayant's subsample-interpolation [76] scheme, odd and even samples of the input speech signal are sent in different packets. Thus if a packet is lost, the missing samples can be interpolated using the neighboring samples that

[1] Part of this chapter represents work published before, see [74, 75].

were received correctly. Unequal error protection schemes have been studied by Davis et al. [35], Danskin et al. [36] and Sherwood et al. [37], in which error correction codes are applied according to the importance of the data to provide different levels of protection. A robust audio tool (RAT) is proposed by Hardman et al. for multicast teleconferencing [29]. In RAT, each packet carries explicitly a redundant version of the previous packet, which can be used for loss recovery. Similar ideas have also been explored and extended to video coding by Bolot at al. [77, 78, 79] and Podolsky et al. [80]. To deal with the network heterogeneity issue in broadcast/multicast applications, a hierarchical/layered FEC scheme is proposed by Tan et. al. [81] and Chou et al. [82] in which end users can subscribe to different number of channel layers for error protection based on their own observed packet loss statistics.

Recently, there has been renewed interest in using the technique of multiple description coding (MDC) for error protection and control over the packet network [83, 84, 85, 86, 76, 87, 88]. As formulated by El-Gamal and Cover in the 1980s [83], the basic idea of MDC is to encode the input signal into multiple descriptions with the constraint that any one of the descriptions can render an acceptable reconstruction of the signal. Furthermore, it requires that the more the descriptions, the better the reconstruction quality. Assume each description is sent through one packet, then an acceptable signal reconstruction can be guaranteed as long as one packet is received correctly. This is reasonable for networks where QoS is not available and therefore there is no way to give more priority to certain packets. This assumption is well matched to the MDC philosophy, where all packets carry equally important information. Among recently proposed MDC techniques we cite the DPCM diversity system by Ingle et al. [89] for packet speech and the wavelet image MDC coding scheme by Servetto et al. [90] which both use the multiple description scalar quantizer (MDSQ) designed by Vaishampayan [91, 92], the multiple description perceptual audio coder (MDPAC) presented by Arean et al. [30], which uses the Multiple Description Transform Coding (MDTC) technique proposed by Wang et al. [62] and Orchard et al. [63], and further developed by Goyal et al. [66]. However, these MDC techniques usually involve complex system designs and implementations which may affect their effectiveness

for real-time applications. For example, MDSQ requires careful index assignment, which becomes difficult if more than two descriptions are necessary [91]. To build a complete transform coding system, the MDTC approach necessitates another correlating transform besides the conventional decorrelating transform [63, 66].

In this chapter, we propose a new MDC scheme for packet loss recovery over packet networks. Different from previous MDC works, data loss recovery is achieved using the redundancy *explicitly* carried by the encoded data, an idea inspired by the work of Hardman et al. [29] and Bolot et al. [78] on robust packet speech/audio over the Internet. In the proposed scheme, the input is first split into different components and each component is quantized separately at different resolutions. In this work, a polyphase transform is used for source splitting. Each polyphase component is coded independently at relatively high quality (i.e., finely quantized). Redundancy is then *explicitly* added to each description by coding other polyphase components at a lower coding rate. In case of packet losses, i.e., descriptions are missing, the lost data is recovered using the redundancy carried in the correctly received packets. The approach of adding redundancy *explicitly* leads to a very simple system design and implementation. The use of polyphase transform also enables us to incorporate context-adaptive coding techniques [93] to further improve the system coding efficiency. We will show that our proposed MDC system, although simple, can yield very competitive coding performance as compared to previously published work. More recently (at the time of writing this chapter), much improved performance results have been reported by Moguel et al. by increasing the *granularity* of redundancy addition, i.e., varying the amount of redundancy with the importance of the data for unequal loss protection [94].

One may have noticed that MDC approaches can actually be combined with FEC schemes for a better overall error protection. For example, one can send redundant packets following the data packets for loss recovery, where each packet can be protected using the error-correcting codes. Indeed, there have been such hybrid error control schemes reported recently, e.g., the MDC via FEC work by Puri et al. [95] and the generalized MDC scheme through unequal error loss protection by Mohr et al. [96]. Nevertheless, improving the performance through the use of error protection codes goes outside the scope of this chapter.

The remainder of this chapter is organized as follows. In the next section, we provide a characterization of the erasure channel and we define the problem. In Section 4.3, the proposed MDC system using polyphase transform and selective quantization is detailed. The performance analysis in terms of optimal redundancy bit allocation and comparisons for Gaussian i.i.d sources are presented in Section 4.4. Section 4.5 provides image and speech coding results using the proposed MDC technique. Finally, we conclude our work in Section 4.6.

## 4.2 Erasure Channel Model and Problem Definition

The erasure channel model which we are going to study in this work is a simplified one. Each packet of transmitted data is assumed either to be completely lost in case of channel failures or received correctly when the channel is good. In terms of the seven-layer OSI network model, in essence, we do not consider bit errors introduced in the physical layer and we assume that these have been corrected or the whole packet has been discarded if it was not possible to correct the errors. Instead, we consider an alternative data recovery technique for the application layer, where the smallest data unit is a single packet of the information source. That is, the channel is a *packet erasure channel*. These erasures will occur largely due to the network congestion and link failures.

To begin with, we make the following two assumptions:

- *The event of channel failure occurs independently*
  Examples include mobile channels spaced beyond the coherence bandwidth, time slots separated larger than the coherence time and packets routed through different paths in the network.

- *The receiver knows which channel fails*
  That is, the receiver knows which description is lost. This can be achieved by inserting synchronization points in the bitstream or tagging packets with sequence numbers.

The independent packet failure assumption, though a simplification, can serve as a good approximation when the network traffic is sufficiently multiplexed from different sources and a random dropping policy is used to relieve the congestion (for more details, see [78]). Though the failure probability of a single channel is large in some cases, the first assumption ensures that the probability of simultaneous multiple independent channel failures will tend to be small. A multiple description coding system, taking advantage of this reasonable assumption, generates independent bitstreams (descriptions), each of which is sent over different channels. This channel diversity thus helps greatly to increase the probability of the event that at least one description arrives successfully at the receiver. To recover the lost description(s), each description has to carry, in addition to the information about itself, extra information (redundancy) about other descriptions.

Assume total $M$ descriptions are generated for the source. Let us define

*Central Distortion* $D_c$ : the reconstruction error using all $M$ descriptions.

*Side Distortion* $D_s$ : the reconstruction error using only $k$ descriptions ($k < M$).

For a fixed bit budget, one would prefer descriptions independent of each other to minimize $D_c$. However, for loss recovery, one has to introduce redundancy among descriptions to reduce $D_s$. Our goal is then

**Objective** *For a given total bit rate $R$ and a channel failure model, design a multiple description coding system to minimize the average central distortion $D_c$ as well as the average side distortion $D_s$.*

## 4.3 MDC Based On Explicit Redundancy

### 4.3.1 Base System

The proposed MDC system is shown in Fig. 4.1. The two quantizers $Q_1$ and $Q_2$ are respectively the fine (high rate $R_0$) and coarse quantizers (low rate $\rho$). The input source $X$ is first split into two subsources $Y_1$ and $Y_2$. In this case, a polyphase transform is applied but other choices are also possible. In a two description coding system, $Y_1$ consists of all even indexed samples and $Y_2$ consists

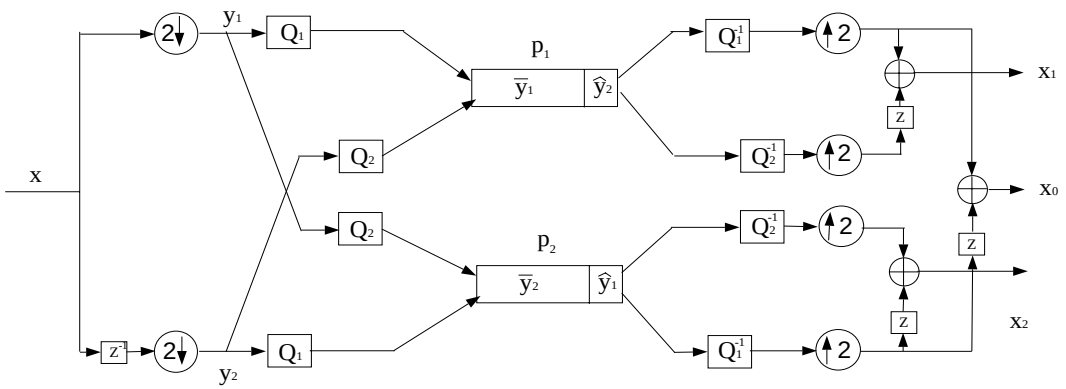



Figure 4.1: The proposed MDC system.

of all odd indexed samples. The two polyphase components, $Y_1$ and $Y_2$, are then finely quantized using $Q_1$ and packed into corresponding packets $P_1$ and $P_2$. For error protection and recovery, each packet also carries a coarsely quantized version of the other polyphase component. For example, $P_1$ consists of $\bar{y}_1$, finely quantized $Y_1$, and $\hat{y}_2$, coarsely quantized $Y_2$. The other packet $P_2$ is obtained similarly, i.e., $(\bar{y}_2, \hat{y}_1)$. At the receiver, if only one packet is received, then one finely quantized polyphase component and one coarsely quantized polyphase component are used for reconstruction. For example, if only $P_1$ is received, $\bar{y}_1$ and $\hat{y}_2$ are used for reconstruction. However, if both packets are received, then only finely quantized data $\bar{y}_1$ and $\bar{y}_2$ are used in the signal reconstruction. In this sense, $\hat{y}_1$ and $\hat{y}_2$ are called *redundant information* whose function is solely to provide packet loss protection.

### 4.3.2 Context-Adaptive Extension

Obviously, to improve the bandwidth efficiency, one would like to use as few bits as possible to encode the redundant information. This motivates us to consider context-adaptive coding techniques. The basic idea of a context-based coding technique is to make use of the knowledge of the neighborhood statistics for the data to be encoded [93]. Since the polyphase components $Y_1$ and $Y_2$ are generated from the same input $X$, strong correlation is expected to exist between $Y_1$ and $Y_2$. This is true for natural speech or image signals and this correlation has been taken advantage of to achieve compression gain in a number of coding standards (e.g.,

G.721 for speeches and JPEG for images). If the input $X$ is the subband data from a DCT or a wavelet transform output, linear correlation is approximately removed in most cases. However, strong structural similarities still exist between polyphase components. One example is that large magnitude coefficients tend to cluster together and so is the case for smaller magnitude coefficients. This feature has been exploited extensively and proven to be very successful in context adaptive codecs developed recently for image coding applications [93, 97]. To incorporate the idea of context adaptive coding, the redundant information, $\hat{y_2}$ and $\hat{y}_1$ in our proposed MDC system, is quantized and coded conditioned on the dequantized data, $\bar{y}_1$ and $\bar{y}_2$. This is shown in Fig. 4.2 where the coarse quantizer $Q_2$ also has as an input the dequantized data from $Q_1$. As a result, more efficient quantization of the redundant information can be achieved. Note that, because we limit our context-based coding to operate within each packet, a packet loss does not affect the decoding of other packets.

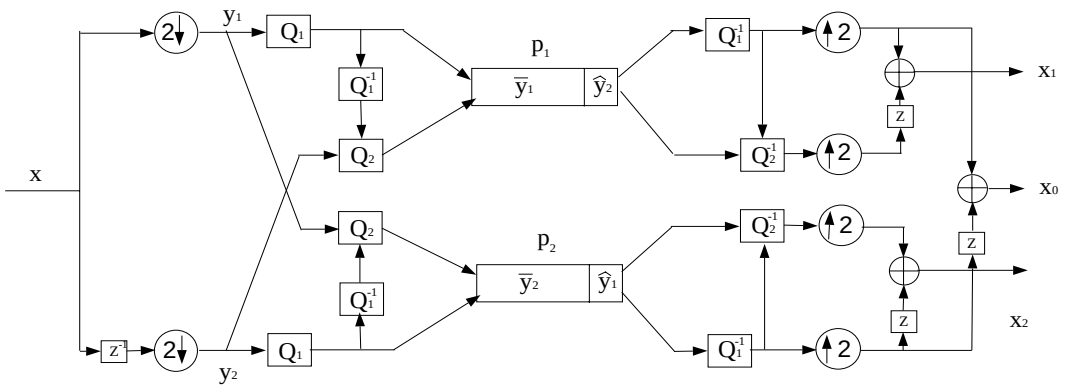



Figure 4.2: Context-based MDC System

## 4.4 Optimal Redundancy Bit Allocation for Gaussian Sources

We now give a performance analysis of our proposed MDC system for i.i.d Gaussian random sources. We answer the following question: for a given total bit

rate, what is the optimal bit allocation between the source coding (primary information) and the channel coding (redundant information) to minimize both the central distortion and side distortions?

### 4.4.1 Implicit Channel Modeling

Consider first the traditional MDC problem without explicit channel modeling. For the problem of two descriptions coding, the goal is to find the operational optimal regions for the 5-tuple $(R_{s,1}, R_{s,2}, D_c, D_{s,1}, D_{s,2})$ where $D_{s,i}$ is the side distortion of description $i$ at rate $R_{s,i}$ [83]. We only study equal rate MDC systems in which all descriptions are coded with the same bit rate, i.e., $R_{s,1} = R_{s,2}$.

Assume $X$ is an i.i.d zero mean random source with pdf $f(x)$ and variance $\sigma^2$. Under the high resolution quantization assumption, its rate-distortion function can be approximated as $D = h\sigma^2 2^{-2R}$, with $h$ an integral constant defined as $h = \frac{1}{12}\left\{\int_{-\infty}^{\infty}[f(x)]^{1/3}dx\right\}^3$[98]. For Gaussian sources, $h = \sqrt{3}\pi/2$. Obviously, applying the polyphase transform on a memoryless source does not change the distortion-rate function. Any subsampling, scrambling, etc., would not change the statistics, since the data is random. Therefore, the polyphase components $Y_1, Y_2$ will have same distortion-rate functions as $X$.

Let the bit rate for primary information be $R_0$ and the redundant bit rate be $\rho$, the side distortion (when one description is lost) and effective bit rates are then

$$D_{s,i} = \frac{1}{2}h\sigma^2 2^{-2\rho} + \frac{1}{2}h\sigma^2 2^{-2R_0} \tag{4.1}$$

$$R_{s,i} = \frac{1}{2}(R_0 + \rho) \tag{4.2}$$

$$i = 0, 1 \tag{4.3}$$

The corresponding central distortion achieved is

$$D_c = h\sigma^2 2^{-2R_0} \tag{4.4}$$

at a total bit rate $R = R_0 + \rho$. With this simple formulation, the optimal bit allocation between $R_0$ (primary information) and $\rho$ (redundant information) then becomes a constrained optimization problem. Using a Lagrange multiplier, define

the cost function $J$ as

$$J = D_c + \lambda D_s \tag{4.5}$$

$$= h\sigma^2 2^{-2(R-\rho)} + \lambda(\frac{1}{2}h\sigma^2 2^{-2\rho} + \frac{1}{2}h\sigma^2 2^{-2(R-\rho)}) \tag{4.6}$$

The optimal redundancy bit rate $\rho^*$ can be analytically solved by having $\frac{\partial J}{\partial \rho}|_{\rho=\rho^*} = 0$ which leads to

$$\rho^* = \frac{1}{2}R + \frac{1}{4}\log_2(\frac{\lambda}{2+\lambda}) \tag{4.7}$$

This optimal redundancy $\rho^*$ has a very intuitive explanation. Since $\frac{\lambda}{2+\lambda} \leq 1$, $\rho^*$ is in the range of $[0, \frac{1}{2}R]$ for a given total bit rate $R$. If only side distortion counts (i.e., one description will be lost with very high probability), then the optimal redundancy rate is $\frac{1}{2}R$. This shows that both the primary information and the redundant information part are coded at the same rate, i.e., half of the total bit rate. This is in fact the minimum possible achievable side distortion $D_s$ for our system, which however, is obtained at the expense of the maximum possible central distortion $D_c$. If only central distortion counts (i.e., both descriptions will arrive at the destination with very high probability), then the redundancy $\rho$ should always be set to zero. This means that each channel will carry half of the total information. Upon receiving data from both channels, one can get the minimum possible central distortion for the given bit rate $R$. However, the side distortion will be its maximum in this case since there is no redundancy. In between these two extreme cases, one has the freedom to fine tune the side distortion with respect to the central distortion by choosing different redundancy bit allocations. With this redundancy allocation, the achievable pair of side and central distortion $(D_s, D_c)$ for a given total bit rate $R$ is

$$D_c = h\sigma^2 2^{-R+\frac{1}{2}\log_2(\frac{\lambda}{2+\lambda})} \tag{4.8}$$

$$D_s = h\frac{\sigma^2}{2}2^{-R-\frac{1}{2}\log_2(\frac{\lambda}{2+\lambda})} + h\frac{\sigma^2}{2}2^{-R+\frac{1}{2}\log_2(\frac{\lambda}{2+\lambda})} \tag{4.9}$$

As a practical example, we consider MDC for a unit-variance zero mean memoryless Gaussian source. In our simulation, we first generate a sequence of Gaussian

i.i.d samples. Then the even samples are quantized by a Lloyd-Max quantizer at a source coding rate $R_0$ and the odd samples are quantized by the same Lloyd-Max quantizer at a redundancy coding rate $\rho$. This is our description 1. Description 2 is formed in the same way except that odd samples are quantized at rate $R_0$ and even samples are quantized at rate $\rho$. The central distortion $D_c$ is the MSE achieved at rate $R_0$ of the original source and the side distortion $D_s$ is the average of the MSEs achieved using only description 1 or description 2. We use fixed-length codes for the index coding so the bit allocation is very simple with the only constraint of fixed total coding rate $R = R_0 + \rho = constant$. For example, if the total coding rate is $5bps$, the possible bit allocations are $(R_0, \rho) = (5, 0), (4, 1), (3, 2)$ at which we measure central distortions and side distortions.

The optimal lower bound of the achievable set of 5-tuple $(R_{s,1}, R_{s,2}, D_{s,1}, D_{s,2}, D_c)$ for a unit-variance zero mean Gaussian source has been given by Ozarow [85] as

$$D_{s,1} \geq 2^{-2R_{s,1}} \tag{4.10}$$

$$D_{s,2} \geq 2^{-2R_{s,2}} \tag{4.11}$$

$$D_c \geq \frac{2^{-2(R_{s,2}+R_{s,1})}}{1-(\sqrt{\Pi}-\sqrt{\Delta})^2} \tag{4.12}$$

where $\Pi = (1 - D_{s,1})(1 - D_{s,2})$ and $\Delta = D_{s,1}D_{s,2} - 2^{-2(R_{s,2}+R_{s,1})}$. The total bit rate is given by $R = R_{s,1} + R_{s,2}$. The asymptotic results using the optimal level-constrained MDSQ given by Vaishampayan et al. [92] are

$$D_c \approx \frac{1}{4}h2^{-R(1+a)} \tag{4.13}$$

$$D_s \approx h2^{-R(1-a)} \tag{4.14}$$

where $0 < a < 1$ and $h = \frac{\sqrt{3}\pi}{2}$. The comparisons are shown in Fig.4.3 for different bit rates.

As one can see, the results of our proposed system using a Lloyd-Max quantizer on an average are comparable to those achieved with an optimal level-constrained MDSQ. They can be better than MDSQ when redundancy rates become very low, for relatively low total bit rates. It would be interesting to determine whether we can improve the proposed system performance by using more efficient quantizers

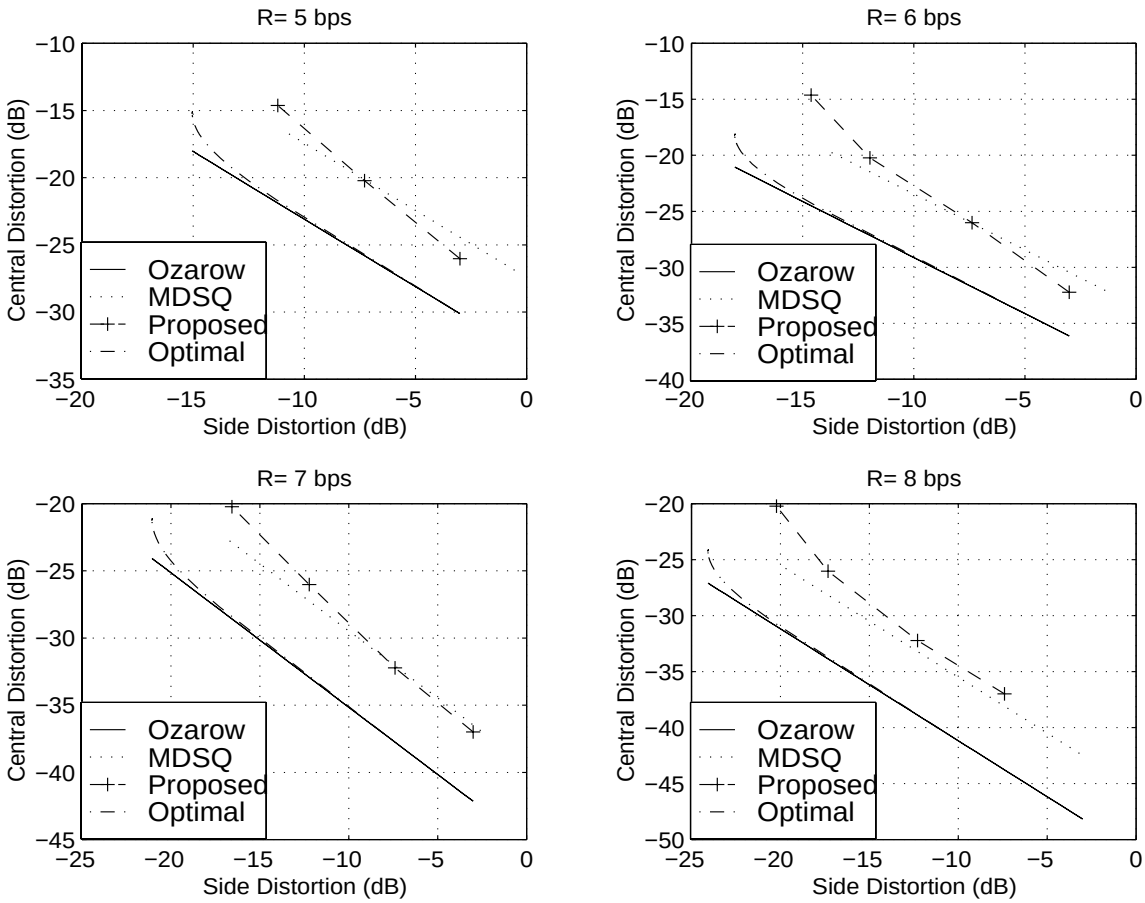



Figure 4.3: Rate-distortion performances comparison for a Gaussian source $\mathcal{N}(0,1)$: (1) Ozarow: optimal bound. (2) MDSQ: optimal level-constrained results. (3) Proposed: Lloyd-Max quantizer results with fixed length code. (4) Optimal: results using the rate-distortion function of the Gaussian source.

other than the Lloyd-Max quantizer. Moreover, it would also be useful to establish what is the best achievable performance within the proposed MDC system framework? For example, for a Gaussian source, if we can design a quantizer which operates exactly on the rate distortion function, can we approach the Ozarow's MDC bounds? Using the optimal bit allocation derived before, the achievable central and side distortions are

$$D_c = \sigma^2 2^{-R+\frac{1}{2}\log_2(\frac{\lambda}{2+\lambda})} \tag{4.15}$$

$$D_s = \frac{\sigma^2}{2} 2^{-R-\frac{1}{2}\log_2(\frac{\lambda}{2+\lambda})} + \frac{\sigma^2}{2} 2^{-R+\frac{1}{2}\log_2(\frac{\lambda}{2+\lambda})} \tag{4.16}$$

These optimal results are plotted in the same figure (see Fig. 4.3). As one can see, the performance gap narrows drastically and almost approaches the Ozarow's lower bounds at lower redundancy rates. This indicates that, while the proposed system cannot achieve the lower bounds within the whole operational range, its performance can be greatly improved if we can design better quantizers. Moreover that at low redundancy rates MDSQ is not as efficient (even it does not introduce explicit redundancy) as using two standard quantizers. We mention that here the quantizer design is exactly the same as that for single description coding. Therefore we can make use of the state-of-art results from single description coding to reach our multiple description coding goals. As a result, the system design and implementation complexity are expected to be reduced compared to specially designed MDC systems, such as the MDSQ [91] and MDTC [63, 66] systems. Our experimental results on MDC for image transmission will further illustrate this point.

### 4.4.2 Explicit Channel Modeling

Next we consider the channel model in the analysis. Assume that the two descriptions are sent over two different channels with independent channel failure probability $p$. There are four different situations at the receiver: (a) both descriptions are received, which happens with probability $(1-p)^2$ and results in distortion $D_c = h\sigma^2 2^{-2R_0}$; (b) one description is lost, which happens with probability $2p(1-p)$ and distortion $D_s = \frac{1}{2}h\sigma^2 2^{-2\rho} + \frac{1}{2}h\sigma^2 2^{-2R_0}$; and (c) both descriptions are lost, which happens with probability $p^2$ and distortion $D_b = \sigma^2$. The average distortion at the receiver for this channel model is

$$\begin{aligned} D &= (1-p)^2 D_c + 2p(1-p)D_s + p^2 D_b \\ &= (1-p)^2 h\sigma^2 2^{-2R_0} + 2p(1-p)[\frac{1}{2}h\sigma^2 2^{-2\rho} + \frac{1}{2}h\sigma^2 2^{-2R_0}] + p^2\sigma^2 \end{aligned}$$

Since the reconstruction distortion in case (c) will not be affected by the redundancy allocation, the optimal bit allocation only needs to minimize the first two terms in $D$. Notice that the total rate $R = R_0 + \rho$. Taking the derivative of $D$ with respect to the redundancy rate $\rho$ and solving the equation $\frac{\partial D}{\partial \rho}|_{\rho=\rho^*} = 0$, it

can be found that, to minimize the average distortion in the presence of channel failures, the optimal bit allocation is

$$\rho^* = \frac{1}{2}R + \frac{1}{4}\log_2 p \tag{4.17}$$

One may notice that the optimal bit allocation $\rho^*$ can actually be computed directly using the result derived before for the case of implicit channel modeling. Notice that the average distortion $D$ for the independent loss channel can be written as

$$D = C_1(D_c + \lambda D_s) + p^2 D_b \tag{4.18}$$

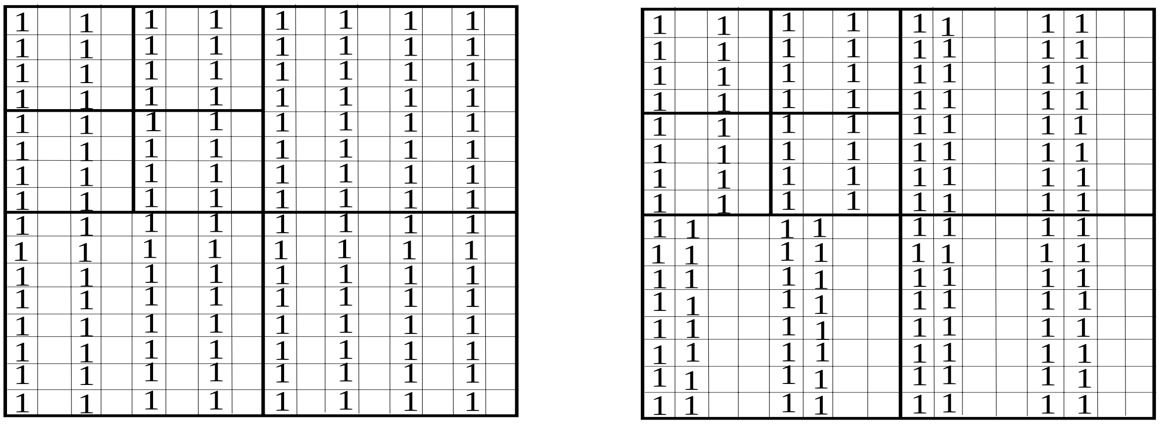
where $C_1 = (1-p)^2$ and $\lambda = \frac{2p(1-p)}{(1-p)^2}$. So the optimization of $D$ is the same as that of $J$ (4.5). That is, the optimal redundancy can be computed from (4.7) using the new $\lambda$. This indicates that the use of an explicit channel model only changes the weighting factor of the side distortion in the cost function $J$ (4.5), i.e., the $\lambda$. This is true that, for a MDC system, once the encoding is completed, all possible decoding scenarios can be enumerated and corresponding side distortions can be computed. The channel model then determines the probability of each decoding scenario, i.e., providing a weighting function to compute the average distortion. By using an arbitrary multiplier $\lambda$, the optimization of cost function $J$ (4.5) actually subsumes all cases using explicit channel models. In this sense, we can say that the proposed MDC system can also be easily extended for other non-independent-loss channel models, e.g., the bursty erasure channel modeled with a Markov chain loss process.

## 4.5 Experimental Results

### 4.5.1 An Image MDC Example

In this section, we present a wavelet image MDC example. As shown by our analysis, the more efficient the quantization scheme, the better performance of our MDC system. Among these state-of-art wavelet coders [93, 99, 100, 101], we

choose the Said-Pearlman wavelet coder due to its simplicity and to the fact that the code is available [2].



(a)

(b)

Figure 4.4: An example configuration of two different polyphase transforms using a two-level wavelet decomposition on a 16x16 input matrix. In all the subbands, all the 1 samples constitute one polyphase component and the remaining samples constitute the other polyphase component. (a) plain polyphase transform. (b) vector polyphase transform.

In our experiment, the input image is first wavelet transformed and its polyphase components are extracted. In this case the polyphase transform of the wavelet coefficients, not that of the original image data, is extracted. Two different types of polyphase transforms are tested on the wavelet coefficients (see Fig. 4.4). One is the plain polyphase transform which, for two descriptions coding, consists simply in grouping all the even coefficients into one description and all the odd coefficients into the other description. This is done for each row in each subband. The second is not a polyphase transform in strict sense but can be viewed as a generalized polyphase transform in vector form. Each subband is first grouped into vectors of same size and these vectors' indices are used in the polyphase transform. For example, if every two successive wavelet coefficients are grouped into a vector of size 2, then there are 4 such vectors in each row in in each subband at the first decomposition level, refer to Fig. 4.4. Denoted these 4 vectors as $\{\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}\}$.

---

[2] The author would like to thank Dr. A. Said and Prof. W. A. Pearlman for providing the SPIHT image coder.

Polyphase component 1 will have $\{\mathbf{v_0}, \mathbf{v_2}\}$ and polyphase component 2 will have $\{\mathbf{v_1}, \mathbf{v_3}\}$. The vector size is increased across subbands as $2^j, j = 0, 1, \cdots, J-1$. The motivation for introducing such a generalized polyphase transform is to preserve spatial structures across subbands, which is used in the SPIHT coder to improve the coding efficiency [101].

Let the two polyphase components be $y_1$ and $y_2$. Then $(y_1(R_0), y_2(\rho))$ constitutes our first description and $(y_2(R_0), y_1(\rho))$ the second description. The Said-Pearlman wavelet coder [101] is used to quantize and entropy code the polyphase components. For example, $y_1(R_0)$ means that $y_1$ is coded at a bit rate $R_0$ with the Said-Pearlman coder. If one description is lost, reconstruct from the received data $((y_1(R_0), y_2(\rho))$ or $(y_2(R_0), y_1(\rho)))$ which gives the side distortion. The central distortion is derived using $(y_1(R_0), y_2(R_0))$. The total coding rate is $R_0 + \rho$.

Since the Said-Pearlman coder makes use of the zerotree structure among subbands, the second type of polyphase transform generates slightly better coding results. In Fig. 4.5 we show MDC results for Lena gray-level image (size 512x512) using the Said-Pearlman wavelet coder [101]. The results of two descriptions are plotted and the comparison with a recent MDSQ-based MDC wavelet coder by Servetto et al. [90] is also given. With a fixed total coding rate, our MDC coder achieves better rate-distortion performance in the whole redundancy range. In Fig. 4.6 we show the reconstructed Lena images using only one channel information with different redundancy rates at a total coding rate $0.5bps$ (the vector form polyphase transform is used).

In the second experiment, we measure the average achieved PSNRs when there are independent packet losses. The input image is first wavelet transformed and then a polyphase transform (the zerotree vector form) is implemented on the wavelet coefficients. The downsampling factor is 16 so we have a total of 16 polyphase components. To protect from channel failures, the redundancy is carried in a sequential way like that used in RAT [29]. That is, packet 1 carries redundancy to protect packet 2 while packet 2 carries redundancy to protect 3 and so on.

Each polyphase component constitutes the primary part in each packet while it also carries redundancy to protect the next polyphase component in sequence.

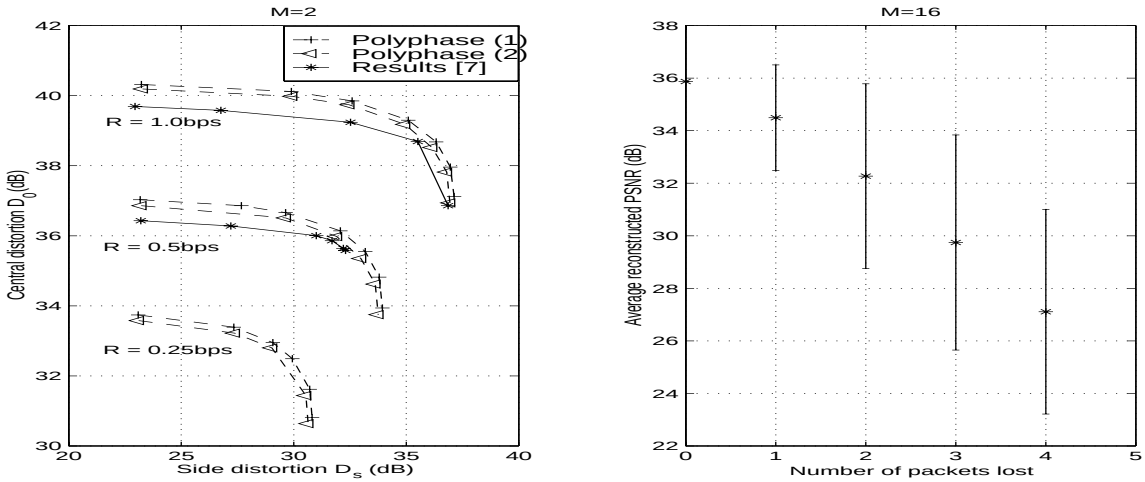



Figure 4.5: Experimental results with Lena gray-level image. (a) Two descriptions. Polyphase (1): plain polyphase transform. Polyphase (2): vector polyphase transform. (b) Performances with independent packet losses.

For example, packet 0 carries two parts of information:(1) polyphase component 0 coded at rate $R_0$ and (2) polyphase component 1 coded at rate $\rho$. We emphasize that $R_0$ and $\rho$ apply to all pixels, and therefore the total rate is indeed $R_0 + \rho$.

In this experiment, we fix the coding rates with $R_0 = 0.4bps$ and $\rho = 0.1bps$ so that the total coding rate is $R = 0.5bps$. Since the total number of wavelet coefficients in each packet is the same, all 16 packets have the same size. We then measure the reconstruction error assuming independent packet losses. For example, assume there are 4 packets lost during the transmission, we first generate the loss pattern independently with 4 erasures. Let one loss pattern be 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 with $1s$ representing received packets while $0s$ represent lost packets. In this case, packet 4 can be reconstructed by using the redundancy carried by packet 3. The same is true for packet 9 and 11 while packet 5 will be lost without reconstruction. We tested 1000 loss patterns for each case when there are 1/2/3/4 packets lost.

In Fig. 4.5 we show the image MDC results at total rate $0.5bps$ ($0.1bps$ redundancy rate). With different number of independent packets losses, the average PSNRs are plotted using star symbols with the standard deviations in vertical bars. As one can see, reconstructed PSNRs deviate from the mean values with an average standard deviation about 3dB. The quality changes are due to changes


Figure 4.6: Reconstructed Lena images at total rate $0.5bps$ using only one channel information. (a) Original image; (b) Redundancy bit rate $0.05bps$, PSNR 29.64dB; (c) Redundancy bit rate $0.10bps$, PSNR 31.91dB; (d) Redundancy bit rate $0.15bps$, PSNR 32.98dB

in the different loss patterns with consecutive losses lead to the worst reconstructions. For the case of only one packet loss, the average PSNR is 34.5dB with standard deviation about 2dB. This is due to the fact that we assume that the first packet loss is not recovered in the experiment. If the first packet can also be recovered using the redundancy carried by the last packet, then the average PSNR becomes 34.99dB with standard deviation 0.12dB.

The experimental results show that the system performance degrades gradually as the packet loss rate increases. However, it also indicates that simple sequential packet protection does not perform well in cases of consecutive losses. Recently, Miguel et al. have shown that better error protection can be achieved by increasing

the *granularity* of redundancy addition [94] by incorporating the idea of unequal error protection. In their work, the amount of redundancy is varied according to the importance of the data, i.e., more redundancy is given to important data and less for unimportant data. For future improvement of our proposed MDC technique, the idea of unequal error protection is certainly worth to be further explored.

### 4.5.2 A Speech MDC Example

In this experiment, we show an example context-adaptive MDC system for robust speech coding over a lossy packet network and compare the results with those of the RAT scheme [29, 78]. The speech materials consist of two sentences recorded at 16KHz and 16bps, one male speaker with "A tamed squirrel makes a nice pet" and one female speaker with "Draw every outer line first, then fill in the interior". Each 20ms speech segment is sent in one packet with 320 samples in each packet. A 14 bits per sample PCM coder is used as the fine quantizer and a 2-bit ADPCM coder is used as the coarse quantizer in the simulation. The 14bps PCM coder is obtained by removing the two LSB bits from the original speech.

A schematic plot of speech coding and packetization is shown in Fig.4.7. In the proposed scheme, the two polyphase components $Y_1$ (all even samples) and $Y_2$ (all odd samples) of the input vector $X$ are independently quantized using a fine scalar quantizer $Q_1$ and packed, respectively, into packets $P_1$ and $P_2$. To achieve loss protection, $P_1$ needs to carry also a coarse version of polyphase component $Y_2$. Rather than quantize $Y_2$ directly using a coarse quantizer $Q_2$, we first predict $Y_2$ using the dequantized data $\bar{Y}_1$ and only quantize the prediction residue $r_2(n)$.

$$r_2(n) = y_2(n) - (\bar{y}_1(n) + \bar{y}_1(n+1))/2 \tag{4.19}$$

$$r_1(n) = y_1(n) - (\bar{y}_2(n-1) + \bar{y}_1(n))/2 \tag{4.20}$$

where $r_2(n)$ and $r_1(n)$ are the prediction residues of $Y_2$ using $\bar{Y}_1$ and $Y_1$ using $\bar{Y}_2$ respectively. These prediction residues are then quantized using the coarse quantizer $Q_2$. As one can see, a simple average interpolation is used for prediction though more more sophisticated techniques can be applied as well.

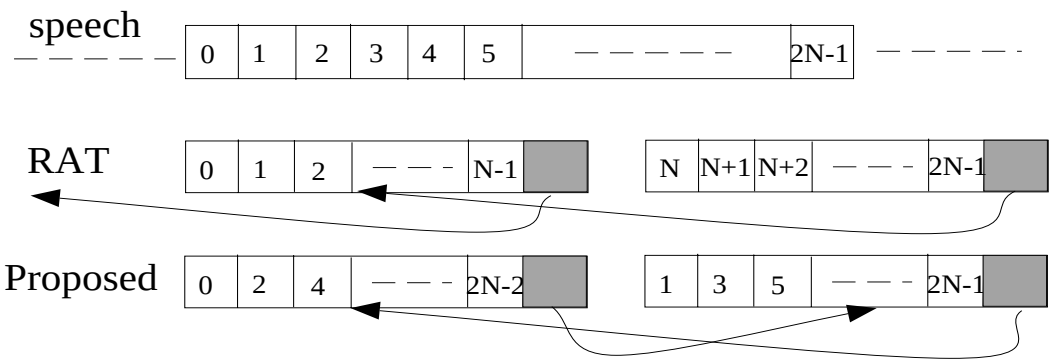



Figure 4.7: The proposed and the RAT schemes for robust packet speech coding.

In the same figure, we also show the RAT scheme [29], which segments the input $X$ into two N sample frames. Each frame is quantized, coded, and packed independently into one packet. Each packet also carries a coarsely quantized data of previous $N$ samples for loss protection.

Three different schemes are implemented and tested in the simulation [3].

1. The subsample-interpolation method by Jayant [76]. The interpolation is the average of two neighboring samples as shown before. Each frame is 16bps PCM coded.

2. The RAT scheme in which each packet carries a 2-bit ADPCM coded data of the previous packet for loss protection and the main part is 14bps PCM coded.

3. Every two frames are polyphase transformed and coded by the 14bps PCM coder. The interpolation is the average of two neighboring samples as shown before. The prediction residues are then coded using the 2-bit ADPCM.

To measure the reconstruction quality of speech signal, we use the Noise-to-Mask Ratio (NMR) which measures the relative energy of noise components above

[3]In this experiment, for illustration purpose, the PCM coder is chosen for encoding the primary information and the ADPCM coder is chosen for encoding the redundancy. More advanced coders can certainly be used and more gains are expected, though, more delicate context adaptive techniques are needed.

the signal's audible masking threshold [102]. The NMR is defined as [102]

$$NMR = \frac{10}{M}\sum_{i=0}^{M-1}\log 10\frac{1}{B}\sum_{b=0}^{B-1}\frac{1}{C_b}\frac{\sum_{k=k_l}^{k=k_h}|D(i,k)|^2}{T_b^2(i)} \tag{4.21}$$

where $M$ is the total number of frames, $B$ is the number of Critical Bands (CB), $C_b$ is the number of frequency components for CB b, and $|D(i,k)|^2$ is the power spectrum of the noise at frequency bin $k$ and frame $i$. The $k_l, k_h$ are respectively the low and high frequency bin indices corresponding to CB b.

We choose NMR rather than other criteria (e.g., Mean Opinion Score) as performance measurement for a number of reasons. One is that NMR is an objective measure based on human hearing system and it has been found to have a high degree of correlation with subjective tests [103]. Second, although the MOS result is subjectively a better indication of the speech audio quality, most MOS results published are based on limited tests within the authors' research group or department and it is difficult to compare. Third and the most important point is that our focus here is to see the relative performance variations between different algorithms and NMR is easy to compute.

Table 4.1: "Squrriel" reconstruction NMR comparison (dB)

| *Loss Prob* | *JAY* | | *RAT* | | *Proposed* | |
|---|---|---|---|---|---|---|
| | *mean* | *std* | *mean* | *std* | *mean* | *std* |
| 10% | 3.36 | 7.20 | 4.86 | 3.37 | 0.36 | 7.02 |
| 15% | 6.41 | 10.37 | 7.49 | 7.25 | 3.40 | 10.36 |
| 20% | 8.61 | 11.44 | 9.24 | 10.36 | 5.99 | 11.41 |
| 30% | 12.34 | 19.96 | 12.68 | 19.65 | 10.67 | 19.98 |

In Table 4.5.2 we show the reconstruction NMR results for the *Squirrel* sentence under independent packet losses at different loss rates. Each loss rate is simulated 100 times and the results are taken as the ensemble averages. It can be seen that the proposed scheme achieves lowest mean NMR at all loss rates. However, the standard deviation of the reconstruction NMR is larger compared to that of the RAT scheme. The reason is that when two consecutive packets are lost, the proposed scheme can not recover if these two packets are both polyphase

components from the same speech frame. However, the RAT scheme can still recover one packet. The same observations can also be seen from the NMR results of the *Draw* sentence given in Table 4.5.2 and Fig.4.8. One possible solution is to introduce more than two descriptions coding, for example, three or four descriptions coding as long as the delay constraints are observed. Provided the number of consecutive packet losses is smaller than the number of descriptions, one can avoid the catastrophic case when all descriptions are lost.

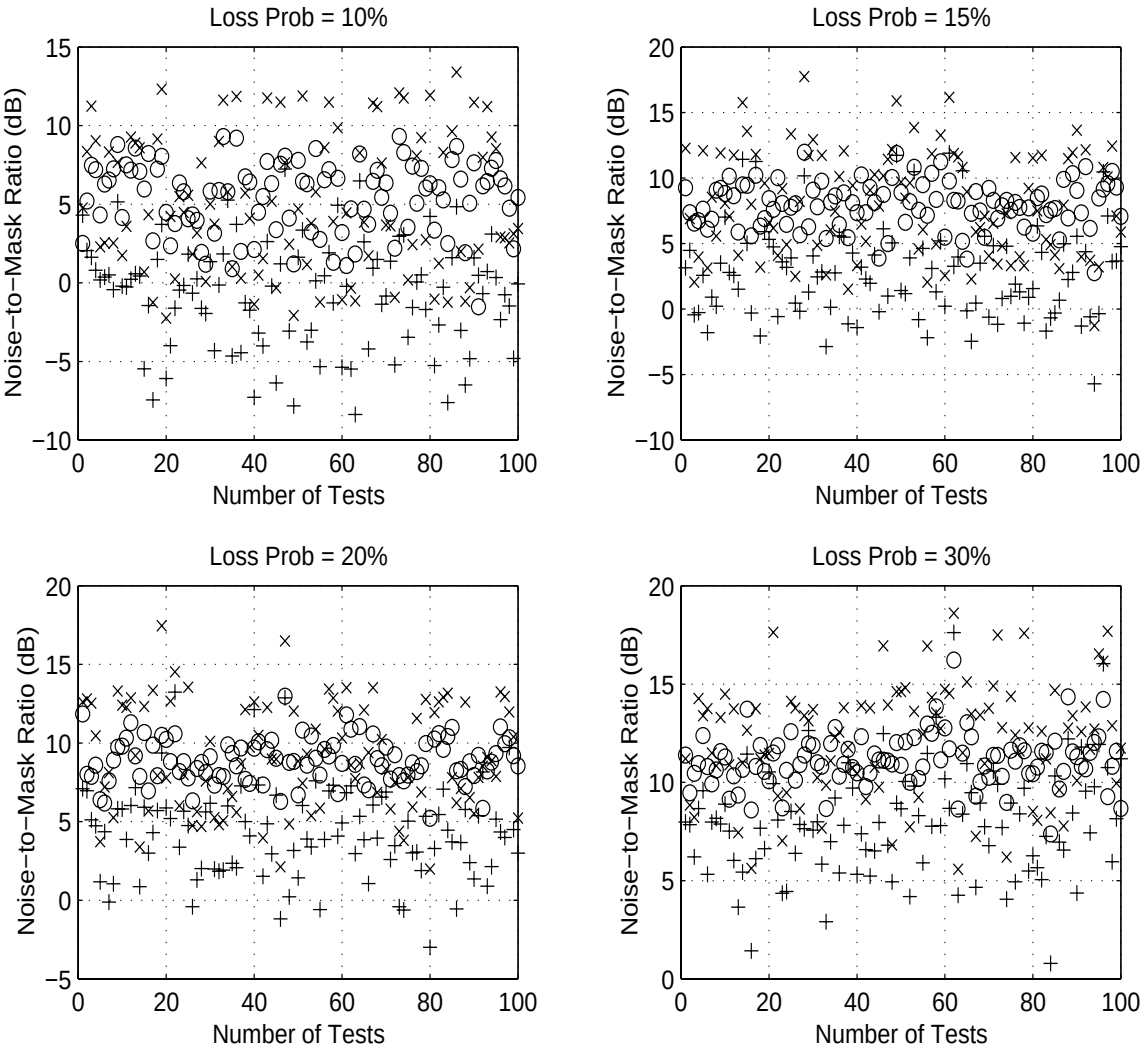



Figure 4.8: Reconstructed NMR distribution plot for the *Draw* sentence under different packet loss probabilities. RAT: o; Proposed: +; JAY: x

Table 4.2: "Draw" reconstruction NMR comparison (dB)

| *Loss Prob* | *JAY* | | *RAT* | | *Proposed* | |
|---|---|---|---|---|---|---|
| | *mean* | *std* | *mean* | *std* | *mean* | *std* |
| 10% | 5.07 | 0.43 | 5.42 | 2.71 | 0.76 | 7.03 |
| 15% | 8.01 | 9.79 | 7.95 | 4.30 | 2.77 | 5.01 |
| 20% | 9.11 | 9.45 | 8.89 | 4.43 | 4.28 | 5.05 |
| 30% | 12.11 | 11.28 | 11.10 | 6.84 | 7.92 | 8.59 |

## 4.6 Summary and Future Work

In this chapter, a MDC system using polyphase transform and selective quantization has also been proposed in this chapter. For i.i.d Gaussian sources, we give detailed analysis of optimal bit allocation to achieve minimum average central distortion and side distortion for a fixed total coding rate. Our experimental results have shown that our system implementation, compared to previous proposed systems, is simple yet the achieved MDC results for image coding is better, especially at lower redundancy rates.

One interesting question is to consider that whether MDC techniques are applicable to robust multicast applications for multimedia communications, Our preliminary work has shown that MDC provides better end-to-end reconstruction signal quality compared to that of layered coding [104] when the network is heavily loaded and delay constraints become critical. We believe that, the MDC system can also be designed and implemented in a way to provide a *layered* coding scheme to cope with the network heterogeneity issue. Each description can be sent through one multicast group and end users can subscribe to a different number of groups based on their bandwidth availability and observed channel statistics. In other words, end users can choose to receive from one to $M$ descriptions by subscribing to corresponding multicast groups. By doing so, not only can one deal with the bandwidth heterogeneity issue (i.e., the goal of LC [104]) but also with the packet loss heterogeneity issue (i.e., the goal of the hierarchical and layered FEC schemes [81, 82].

# Appendix A

# Overlap-Add and Overlap-Save

Standard techniques for linear FIR filtering over a long data sequences include *overlap-save* and *overlap-add*, both of which are block-based approaches [49]. The input sequence is segmented into blocks, each of which is filtered in the frequency domain using DFT and IDFT. The outputs from each block processing are concatenated together to form the final result which is identical to the sequence obtained as if the whole input sequence had been processed in the time-domain.

Let $x(n)$ be the input sequence. Denote the block length as $M$ and the filter length is $L$.

**Overlap-Save** In this method, each block consists of:$(i)$ $(L-1)$ samples from the previous block; and $(ii)$ $(M-1)$ new samples from the sequence. The data blocks are

$$\begin{aligned}
b_1(n) &= \{\underbrace{0,0,\cdots,0}_{L-1}, x(0), x(1), \cdots, x(M-1)\} \\
b_2(n) &= \{\underbrace{x(M-L+1,\cdots,x(M-1)}_{L-1}, \underbrace{x(M),\cdots,x(2M-1)}_{M}\} \\
b_3(n) &= \{\underbrace{x(2M-L+1,\cdots,x(2M-1)}_{L-1}, \underbrace{x(2M),\cdots,x(3M-1)}_{M}\}
\end{aligned}$$

and so on. As one can see, the actual block size needed is $N = M + L - 1$ since $(L-1)$ samples have to be overlapped from the previous block. The filtering output is $y_i(n) = IDFT(FFT(b_i(n)) * FFT(h))$. The first $(L-1)$ samples are discarded due to aliasing and the remaining $M$ samples constitute the desired result from linear convolution.

**Overlap-Add** In this method, each block consists of only $(L-1)$ nonoverlapping samples from the input sequence. The data blocks are

$$b_1(n) = \{x(0), x(1), \cdots, x(M-1), \underbrace{0,0,\cdots,0}_{L-1}\}$$

$$
\begin{aligned}
b_2(n) &= \{x(M), \cdots, x(2M-1), \underbrace{0, 0, \cdots, 0}_{L-1}\} \\
b_3(n) &= \{x(2M), \cdots, x(3M-1), \underbrace{0, 0, \cdots, 0}_{L-1}\}
\end{aligned}
$$

and so on. As one can see, though the input block is nonoverlapped, the actual block size is still $N = M + L - 1$ becasue of the $(L-1)$ padded zero samples. The filtering output $y_i(n) = IDFT(FFT(b_i(n)) * FFT(h))$ is free of aliasing. The last $(L-1)$ points, however, must be overlapped and added to the first $(L-1)$ points of the succeeding block to form the final result. That is

$$
\begin{aligned}
y(n) &= \{y_0(0), y_0(1), \cdots, y_0(M-1), y_0(M) + y_1(0), \\
&\quad y_0(M+1) + y_1(1), \cdots, y_0(N-1) + y_1(L-1), y_1(L), \cdots\}
\end{aligned}
$$

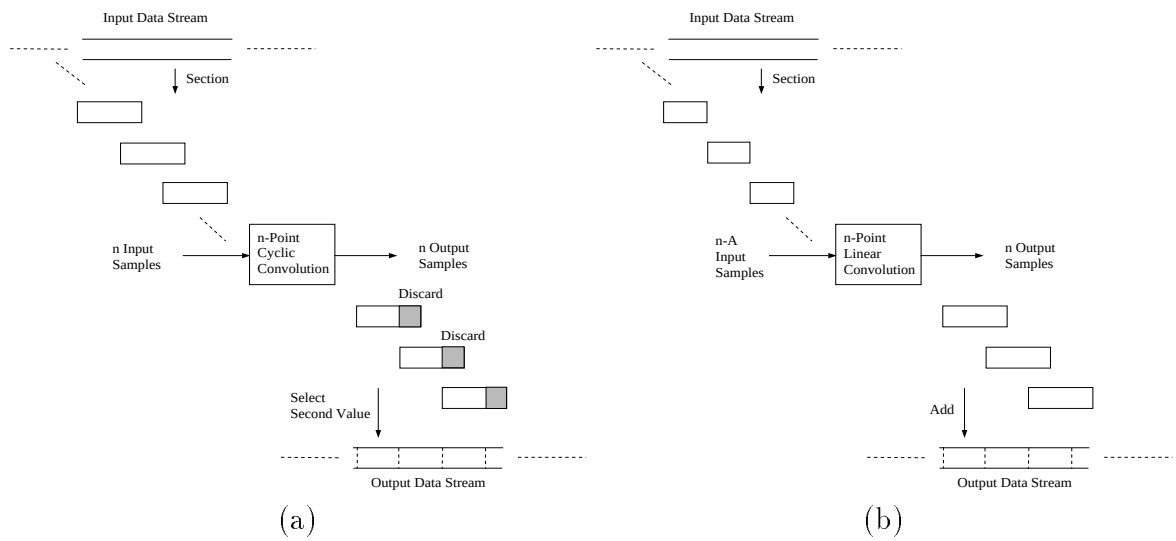



Figure A.1: (a) Overlap-save. (b) Overlap-add.

# Appendix B

# DWT FSM Examples

**Daubechies (9,7) filters**
This filterbank has been used extensively in the image compression algorithms proposed in the literature. The factorization of the analysis polyphase matrix (adapted from [48]) is

$$\mathbf{P}_a(z)=\begin{bmatrix} \zeta & 0 \\ 0 & 1/\zeta \end{bmatrix}\begin{bmatrix} 1 & \delta(1+z^{-1}) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \gamma(1+z) & 1 \end{bmatrix}\begin{bmatrix} 1 & \beta(1+z^{-1}) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \alpha(1+z) & 1 \end{bmatrix}$$

where $\alpha = -1.586134342, \beta = -0.05298011854, \gamma = 0.8829110762, \delta = 0.4435068522$, and $\zeta = 1.149604398$.

Based on this factorization, the forward transform is

$$\begin{aligned}
x_0^0(n) &= x(2n) \\
x_1^0(n) &= x(2n+1) \\
x_1^1(n) &= x_1^0(n) + \alpha(x_0^0(n) + x_0^0(n+1)) \\
x_0^1(n) &= x_0^0(n) + \beta(x_1^1(n) + x_1^1(n-1)) \\
x_1^2(n) &= x_1^1(n) + \gamma(x_0^1(n) + x_0^1(n+1)) \\
x_0^2(n) &= x_0^1(n) + \delta(x_1^2(n) + x_1^2(n-1)) \\
x_0^3(n) &= \zeta x_0^2(n) \\
x_1^3(n) &= x_1^2(n)/\zeta
\end{aligned}$$

and the inverse transform is

$$\begin{aligned}
x_1^2(n) &= \zeta x_1^3(n) \\
x_0^2(n) &= x_0^3(n)/\zeta \\
x_0^1(n) &= x_0^2(n) - \delta(x_1^2(n) + x_1^2(n-1)) \\
x_1^1(n) &= x_1^2(n) - \gamma(x_0^1(n) + x_0^1(n+1)) \\
x_0^0(n) &= x_0^1(n) - \beta(x_1^1(n) + x_1^1(n-1))
\end{aligned}$$

$$
\begin{aligned}
x_1^0(n) &= x_1^1(n) - \alpha(x_0^0(n) + x_0^0(n+1)) \\
x(2n+1) &= x_1^0(n) \\
x(2n) &= x_0^0(n)
\end{aligned}
$$

As one can see, using the (9,7) wavelet filters, total 4 state transitions are needed to transform a raw input sample into a wavelet coefficient. This process is ahown in Fig. B.1. Assume there are 9 samples, $\{x^0(0), x^0(1), \cdots, x^0(8)\}$, loaded in memory initially. The first elementary matrix $\mathbf{e}^0(z)$ is lower triangular, so the state transition is to update odd samples with two neighboring even samples. For example, $x^0(1)$ is updated into $x^1(1) = x^0(1) + \alpha(x^0(0) + x^0(2))$. The same updating also occurs for samples $\{x^0(3), x^0(5), x^0(7)\}$. Notice that samples $\{x^0(0), x^0(8)\}$ remain un-updated because they are needed by neighboring blocks, e.g., $x^0(8)$ is needed by $x^0(9)$ and $x^0(0)$ is needed by $x^0(-1)$ (not shown in the figure). Consequently, $x^0(0)$ and $x^0(8)$ are preserved as the *state* information at state 0.

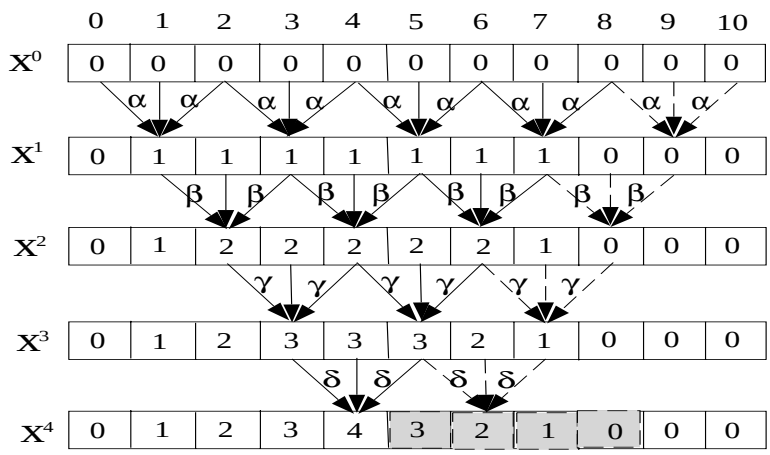



Figure B.1: State transitions of the Daubechies (9,7) filters using factorization (B.1). The state information consists of 4 samples (the overlap buffer size) in shaded boxes. Dashed lines represent operations necessary for the transform of the new input sample pair $\{x^0(9), x^0(10)\}$.

The next elementary matrix $\mathbf{e}^1(z)$ is upper triangular so it updates even samples using odd samples. For example, $x^1(2)$ is updated into $x^2(2) = x^1(2) + \beta(x^1(1) + x^1(3))$ and so are samples $\{x^1(4), x^1(6)\}$. Again, $x^1(1)$ and $x^1(7)$ are preserved as the *state* information at state 1. The same process continues until $x^0(4)$ is updated into the final transform coefficient $x^4(4)$.

The *state* information near the right boundary consists of samples shown in shaded boxes in the figure, i.e., $\{x^3(5), x^2(6), x^1(7), x^0(8)\}$. So the overlap buffer size for one level of wavelet decomposition using the Daubechies (9,7) filters is 4 samples. These *partially updated samples* constitutes the only information one

needs to buffer for the transform of the new input data pair $\{x^0(9), x^0(10)\}$. The operations are shown as dashed lines in the figure. As one can see, all these operations are based on the *state* information which is preserved in the memory buffer.

**(2,10) filters**

This filter has been found to give excellent performances for lossless image compression. The factorization of the analysis polyphase matrix is

$$\mathbf{P}_a(z) = \begin{bmatrix} 1 & 0 \\ \frac{3}{64}(z^2 - z^{-2}) + \frac{22}{64}(z - z^{-1}) & 1 \end{bmatrix} \begin{bmatrix} 1 & 1/2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

Based on this factorization, the forward transform is

$$\begin{aligned}
x_0^1(n) &= x(2n) \\
x_1^1(n) &= x(2n+1) - x(2n) \\
x_0^2(n) &= x_0^1(n) + x_1^1(n)/2 \\
x_1^2(n) &= x_1^1(n) \\
x_0^3(n) &= x_0^2(n) \\
x_1^3(n) &= x_1^2(n) + \frac{3}{64}(x_0^2(n-2) - x_0^2(n+2)) + \frac{22}{64}(x_0^2(n-1) - x_0^2(n+1))
\end{aligned}$$

and the inverse transform is

$$\begin{aligned}
x_1^2(n) &= x_1^3(n) - \frac{3}{64}(x_0^2(n-2) - x_0^2(n+2)) + \frac{22}{64}(x_0^2(n-1) - x_0^2(n+1)) \\
x_0^2(n) &= x_0^3(n) \\
x_1^1(n) &= x_1^2(n) \\
x_0^1(n) &= x_0^2(n) - x_1^1(n)/2 \\
x(2n) &= x_0^1(n) \\
x(2n+1) &= x_1^1(n) + x(2n)
\end{aligned}$$

As one can see,, first two state transitions are basically the same as that of the (9,7) filters. Assume initially there are 10 samples in the memory as shown in Fig. B.2. The last transition is more interesting which is detailed here.

The elementary matrix $\mathbf{e}^2(z)$ is

$$\mathbf{e}^2(z) = \begin{bmatrix} 1 & 0 \\ \frac{3}{64}(z^2 - z^{-2}) + \frac{22}{64}(z - z^{-1}) & 1 \end{bmatrix}$$

This is a lower triangular matrix so odd samples get updated. For example, $x^2(5)$

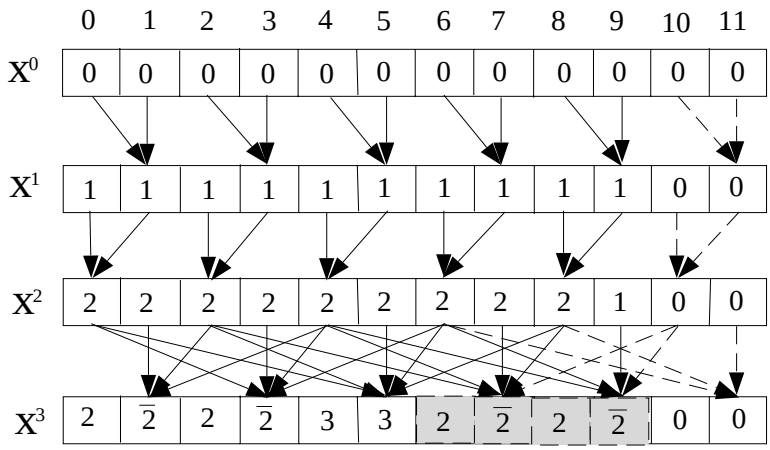



Figure B.2: State transitions of the (2,10) filters using factorization (B.1). The state information consists of 4 samples in shaded boxes. Dashed lines represent operations necessary for the transform of the new input sample pair $\{x^0(10), x^0(11)\}$.

is updated into

$$x^3(5) = x^2(5) + \frac{3}{64}(x^2(8) - x^2(0)) + \frac{22}{64}(x^2(6) - x^2(2)) \tag{B.1}$$

On the other hand, $x^2(7)$ can not be fully updated because $x^0(10)$ is not available (not in buffer yet). However, it is partially updated as

$$\bar{x}^2(7) = x^2(7) + \frac{3}{64}(-x^2(2)) + \frac{22}{64}(x^2(8) - x^2(4)) \tag{B.2}$$

This partial updating then frees sample $x^2(2)$ from the buffer. In other words, to fully update $\bar{x}^2(7)$, no samples with indices smaller than 7 are needed. Same partially updating is also performed for sample $x^2(9)$ as

$$\bar{x}^2(9) = x^2(9) + \frac{3}{64}(-x^2(6)) + \frac{22}{64}(-x^2(4)) \tag{B.3}$$

The only samples which have to be buffered are $\{x^2(6), \bar{x}^2(7), x^2(8), \bar{x}^2(9)\}$. So the overlap buffer size is 4 samples.

When the next new input pair $\{x^0(10), x^0(11)\}$ comes, operations in dashed lines are executed. As a result, samples $\{x^2(6), \bar{x}^2(7)\}$ are completely transformed thus can be removed from the buffer. However, samples $\{x^0(10), x^0(11)\}$ can only be partially updated and thus have to be buffered. This process continues until all inputs are transformed.

**CDF(4,2) filters**

The scaling function of CDF(4,2) filters is a cubic B-spine which is used frequently

in computer graphics for interpolation. The factorization of the analysis polyphase matrix (adapted from [48]) is

$$\mathbf{P}_a(z) = \begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & \frac{3}{16}(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{4}(1+z^{-1}) \\ 0 & 1 \end{bmatrix}$$

Based on this factorization, the forward transform is

$$\begin{aligned}
x_0^0(n) &= x(2n) \\
x_1^0(n) &= x(2n+1) \\
x_0^1(n) &= x_0^0(n) - \frac{1}{4}(x_1^0(n) + x_1^0(n-1)) \\
x_1^1(n) &= x_1^0(n) - (x_0^1(n) + x_0^1(n+1)) \\
x_0^2(n) &= x_0^1(n) + \frac{3}{16}(x_1^1(n) + x_1^1(n-1))
\end{aligned}$$

and the inverse transform is

$$\begin{aligned}
x_0^1(n) &= x_0^2(n) - \frac{3}{16}(x_1^1(n) + x_1^1(n-1)) \\
x_1^0(n) &= x_1^1(n) + (x_0^1(n) + x_0^1(n+1)) \\
x_0^0(n) &= x_0^1(n) + \frac{1}{4}(x_1^0(n) + x_1^0(n-1))
\end{aligned}$$

In this case, the state transition is basically the same as that of the (9,7) filters. The overlap buffer size is 3 samples as shown in Fig. B.3.

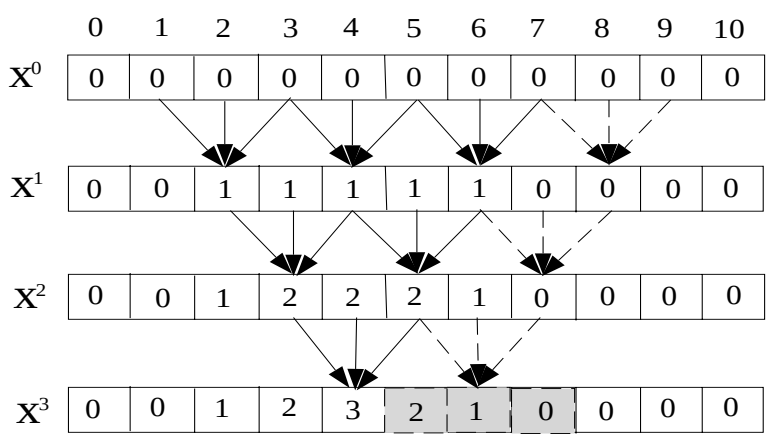



Figure B.3: State transitions of the CDF (4,2) filters using factorization (B.4). The state information consists of 3 samples in shaded boxes. Dashed lines represent operations necessary for the transform of the new input sample pair $\{x^0(8), x^0(9)\}$.

# Appendix C

# MDTC Transform Design Algorithm

The basic algorithm we use is the same as that described in [64]. Here we provide a different derivation for a special channel model. We assume that channel fails independently with an equal channel failure probability $p$ (description loss probability). One example is the best-effort network where independent packet loss is a reasonable model for each individual connection [78]. We mention that it is straightforward to generalize the algorithm to cases of unequal channel failures probabilities.

The above constrained optimization problem can be transformed into a unconstrained one using a Lagrangian multiplier $\lambda$. The cost function $J$ can be written as

$$J = D_s(\mathbf{A}, \mathbf{\Theta}) + \lambda\rho(\mathbf{A}, \mathbf{\Theta})$$

Let $R_X = \{\sigma_{ii}^2\}, i = 1, 2, \cdots, M$ be the correlation matrix of input vector $X$. Let $R_Y = \{r_{ij}\}, i, j = 1, 2, \cdots, M$ be the correlation matrix of the transform output $Y$. Then we have

$$\begin{aligned} R_Y &= T(\mathbf{A}, \mathbf{\Theta}) R_X T(\mathbf{A}, \mathbf{\Theta})^t \\ &= R(\mathbf{\Theta}) S(\mathbf{A}) R_X S(\mathbf{A})^t R(\mathbf{\Theta})^t \\ &= R(\mathbf{\Theta}) R_S R(\mathbf{\Theta})^t \end{aligned}$$

where $R_S = S(\mathbf{A}) R_X S(\mathbf{A})^t = \{a_{ii}^2 \sigma_{ii}^2\}, i = 1, 2, \cdots, M$ is a diagonal matrix.

### C.0.1 Redundancy bit rate $\rho$

Let us first derive the redundancy rate $\rho$ for this correlating transform $T$. Assume the coding rate is $R_X$ for input vector $X$ and components of $X$ are identically distributed. By optimal bit allocation and high resolution quantization, the achieved distortion is [98]

$$D_0 = h\rho_X^2 2^{-2R_X}$$

where $h = \frac{1}{12}\left\{\int_{-\infty}^{\infty}[f(x)]^{1/3}dx\right\}^3$ is a constant integral determined by the component density function $f(x)$ and $\rho_X^2 = (\prod_{i=1}^{M}\sigma_i^2)^{\frac{1}{M}}$ is the geometric mean of the component variances. After transformation, the distortion will remain unchanged since $T$ is implemented losslessly. However, the bit rate will be increased to $R_Y$. We have

$$D_0 = h\rho_Y^2 2^{-2R_Y}$$

where $\rho_Y^2 = (\prod_{i=1}^{M} r_{ii}^2)^{\frac{1}{M}}$. The redundancy bit rate $\rho$ is derived as the increased bit rate from $R_X$ to $R_Y$ as

$$\begin{aligned}\rho(\mathbf{A},\mathbf{\Theta}) &= R_Y - R_X \\ &= \frac{1}{2M}\log\frac{\prod_{i=1}^{M} r_{ii}}{\prod_{i=1}^{M}\sigma_i^2}\end{aligned}$$

### C.0.2 Side Distortion $D_s$

Now we derive the side distortion when there are erasures, i.e., some descriptions lost. We first need to recover the lost descriptions from received descriptions.

Assume $m$ out of $M$ descriptions are lost. Without loss of generality, we can partition $Y$ into *received* and *lost* portions [64], i.e., $Y = [y_r\ y_l]^t$ For notational simplicity, we will assume $r = m$ and $l = M - m$ so $y_r$ is a m-dimensional vector which is available at the decoder while $y_l$ is a $(M-m)$-dimensional vector which is not available at the decoder. Denote the decoder reconstructed descriptions as $\hat{Y} = [\hat{y}_r\ \hat{y}_l]^t$, then $\hat{Y} = [y_r\ \hat{y}_l]^t$. The MMSE estimator of $y_l$, $\hat{y}_l$, based on $y_r$ is the conditional mean, i.e., $\hat{y}_l = E[y_l|y_r]$. For jointly Gaussian vectors, this reduces to a LMMSE estimator, that is

$$\hat{y}_l = R_{lr}R_{rr}^{-1}y_r$$

with mean squared estimation error (MSE)

$$\begin{aligned}E\|e_y\|^2 &= E\|y_l - \hat{y}_l\|^2 \\ &= tr(R_{ee})\end{aligned}$$

where $tr(Q)$ is the trace of matrix $Q$ and

$$\begin{aligned} R_{rr} &= E[y_r y_r^t] \\ R_{ll} &= E[y_l y_l^t] \\ R_{lr} &= R_{rl}^t = E[y_r y_l^t] \\ R_{ee} &= E[e_y e_y^t] = R_{ll} - R_{lr} R_{rr}^{-1} R_{rl} \end{aligned}$$

Once $\hat{Y}$ is found, the estimated $\hat{X} = T^{-1}\hat{Y}$ can be obtained via the inverse correlating transform $T^{-1}$. For simplicity, denote the transpose of $T^{-1}$ as $T^{-t}$. The final mean-squared reconstruction error between $X$ and $\hat{X}$ is

$$\begin{aligned} E(\|e_X\|^2) &= E\|X - \hat{X}\|^2 \\ &= E\|T^{-1}(Y - \hat{Y})\|^2 \\ &= tr(E[T^{-1}(Y - \hat{Y})(Y - \hat{Y})^t T^{-t}]) \\ &= tr(T^{-1} E[(Y - \hat{Y})(Y - \hat{Y})^t] T^{-t}) \end{aligned}$$

Write $Y$ and $\hat{Y}$ in their partitioned forms, we have

$$\begin{aligned} E(e_Y e_Y^t) &= E[(Y - \hat{Y})(Y - \hat{Y})^t] \\ &= \begin{bmatrix} 0 & 0 \\ 0 & R_{ee} \end{bmatrix} \end{aligned}$$

Partition $T^{-1}$ correspondingly,

$$T^{-1} = \begin{bmatrix} T_{rr} & T_{rl} \\ T_{lr} & T_{ll} \end{bmatrix} \qquad T^{-t} = \begin{bmatrix} T_{rr}^t & T_{lr}^t \\ T_{rl}^t & T_{ll}^t \end{bmatrix}$$

After some simplifications, we finally get the reconstruction mean-squared error, the side distortion $D_m$ when there are $m$ descriptions lost, as

$$\begin{aligned} D_m &= E\|e_X\|^2 \\ &= tr(T_{rl} R_{ee} T_{rl}^t) + tr(T_{ll} R_{ee} T_{ll}^t) \end{aligned}$$

The average side distortion $D_s$ is a weighted sum

$$D_s = \sum_{m=1}^{M} P_m D_m$$

with $P_m = \begin{pmatrix} M \\ m \end{pmatrix} p^m (1-p)^{M-m}$, the probability of $m$ descriptions lost.

To summarize, we list steps for side distortion calculation.

1. For $m = 1:1:M$, $m$ is the number of lost descriptions.
   - Calculate the side distortion $D_m$.
     - Partition $R_Y$ to find $R_{rr}$, $R_{ll}$, and $R_{lr}$. Partition $T^{-1}$ to find $T_rl$ and $T_{ll}$.
     - Calculate correlation of estimation error $R_{ee} = R_{ll} - R_{lr}R_{rr}^{-1}R_{rl}$.
     - Calculate side distortion as $D_m = tr(T_{rl}R_{ee}T_{rl}^t) + tr(T_{ll}R_{ee}T_{ll}^t)$.
   - Calculate the event probability $P_m = \binom{M}{m}p^m(1-p)^{M-m}$ ( $m$ descriptions are lost).
2. Calculate the average side distortion $D_s = \sum_{m=1}^{M} P_m D_m$.

# Appendix D

# KLVT Vector Space Partition

Let $< \mathbf{a_1}, \mathbf{a_2}, \cdots, \mathbf{a_N} >$ denote a vector space $\mathbf{A}$ with basis set $\{\mathbf{a_i}, 1 \leq i \leq N\}$. That is, any vector in $\mathbf{A}$ can be represented as a linear combination of the basis vectors. $\mathbf{A}$ is also called the expansion of this basis set. Consider a random sequence $\mathbf{x} = \{x_i, 1 \leq i \leq N\}$. Let $\mathbf{R}$ be an $N \times N$ (assuming $N$ is even) correlation matrix and its eigenmatrix $\mathbf{\Phi}$ is

$$\mathbf{\Phi} = [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_N]$$

where $\{\mathbf{e}_i, 1 \leq i \leq N\}$ are eigenvectors and $\{\lambda_i, 1 \leq i \leq N\}$ are corresponding eigenvalues. Hence $\mathbf{R}$ can be written as

$$\begin{aligned} \mathbf{R} &= \lambda_1 \mathbf{e}_1 {\mathbf{e}_1}^H + \lambda_2 \mathbf{e}_2 {\mathbf{e}_2}^H + \cdots + \lambda_N \mathbf{e}_N {\mathbf{e}_N}^H \\ &= \sum_{i=1}^{N} \lambda_i \mathbf{e}_i {\mathbf{e}_i}^H \end{aligned}$$

which is also called the spectral decomposition of $\mathbf{R}$ on its eigenspace

$$\mathbf{E} = (\mathbf{e}_1 \quad \mathbf{e}_2 \quad \cdots \quad \mathbf{e}_N)$$

in the context of signal processing. In fact, each $\mathbf{e}_i {\mathbf{e}_i}^H$ constitutes an eigen-subspace of dimensionality 1 with no correlation between each other.

Now we merge each two consecutive eigen-subspaces to form a new set of eigen-subspaces as following

$$\mathbf{E}_1 = (\underline{\mathbf{e}}_1, \underline{\mathbf{e}}_2) \qquad \mathbf{E}_2 = (\underline{\mathbf{e}}_3, \underline{\mathbf{e}}_4) \qquad \cdots \qquad \mathbf{E}_{N/2} = (\underline{\mathbf{e}}_{N-1}, \underline{\mathbf{e}}_N)$$

We will have another set of uncorrelated eigensubspaces $\{\mathbf{E_i}, 1 \leq i \leq N/2\}$, each of which with dimensionality 2 and their union is still the eigenspace $\mathbf{E}$.

$$\mathbf{E} = \bigcup_{i=1}^{N/2} \mathbf{E_i} \qquad \mathbf{E_i} \perp \mathbf{E_j}, i \neq j, 1 \leq i, j \leq N/2$$

Projected onto this new set of eigensubspaces, $\mathbf{R}$ can be expanded as following:

$$\begin{aligned} \mathbf{R} &= \mathbf{E_1 \Lambda_1 E_1^H} + \mathbf{E_2 \Lambda_2 E_2^H} + \cdots + \mathbf{E_{N/2} \Lambda_{N/2} E_{N/2}^H} \\ &= \sum_{i=1}^{N/2} \mathbf{E_i \Lambda_i E_i}^{\mathbf{H}} \end{aligned}$$

where $\{\mathbf{E_i}, 1 \leq i \leq N\}$ are $N \times 2$ matrices with rank 2, i.e., the column space of dimensionality of 2, and $\mathbf{\Lambda_i}$ are $2 \times 2$ eigenmatrices.

One notices that, $\{\mathbf{E_i}, 1 \leq i \leq N/2\}$ is a very special partition of the eigenspace $\mathbf{E}$ with each subspace having eigenvectors in the basis set and $\mathbf{\Lambda_i}, 1 \leq i \leq N$ are all diagonal matrices. As a matter of fact, these eigen-subspaces $\{\mathbf{E_i}, 1 \leq i \leq N/2\}$ can have other basis set other than the eigenvectors. In other words, $\mathbf{\Lambda_i}, 1 \leq i \leq N$ need not to be diagonal either. As long as these subspaces are a valid decorrelation partition, $\mathbf{R}$ will be block-diagonalized.

# References


[1] C. Chrysafis and A. Ortega, "Line based, reduced memory, wavelet image compression," in *Proc. of Data Compress. Conf.*, 1998.

[2] P. Cosman, T. Frajka, and K. Zeger, "Image compression for memory constrained printers," in *Proc. of ICIP*, Oct. 1998.

[3] P. Cosman and K. Zeger, "Memory constrained wavelet-based image coding," *IEEE Signal Processing Letters*, Sept. 1998, to appear.

[4] M. Misra and V. K. Prasanna, "Parallel computation of 2-D wavelet transforms," in *Proc. of 11th Intl. Conf. on Pattern Recognition*, 1992.

[5] O. Rioul and P. Duhamel, "Fast algorithms for discrete and continuous wavelet transforms," *IEEE Trans. on Information Theory*, vol. 38, no. 2, pp. 569–586, Mar. 1992.

[6] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," *IEEE Trans. on Signal Proc.*, vol. 42, no. 3, Mar. 1994.

[7] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: From single chip implementations to mappings on SIMD array computers," *IEEE Trans. on Signal Proc.*, vol. 43, no. 3, pp. 759–771, Mar. 1995.

[8] R. E. Van Dyck, T. G. Marshall, M. Chin, and N. Moayeri, "Wavelet video coding with ladder structures and entropy-constrained quantization," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no. 5, pp. 483–494, Oct. 1996.

[9] H. Sava, A. C. Downtown, and A. F. Clark, "Parallel pipeline implementation of wavelet transform," *Proc. Inst. Elect. Eng., Vision Image Process.*, vol. 144, no. 6, pp. 355–359, Dec. 1997.

[10] J. Fridman and E. S. Manolakos, "Discrete wavelet transform: Data dependence analysis and synthesis of distributed memory and control array

architectures," *IEEE Trans. on Signal Processing*, vol. 45, no. 5, pp. 1291–1308, May 1997.

[11] J. Fridman and E. S. Manolakos, "On the scalability of 2-D discrete wavelet transform alogrithms," *Multidimensional Systems and Signal Processing*, , no. 8, pp. 185–217, 1997.

[12] M. A. Trenas, J. Lopez, and E. L. Zapata, "A memory system supporting the efficient SIMD computation of the two dimensional DWT," in *Proc. of ICASSP*, 1998, pp. 1521–1524.

[13] C. Chakrabarti and C. Mumford, "Efficient realizations of encoders and decoders based on the 2-d discrete wavelet transform," *IEEE Transactions on VLSI Systems*, 1998.

[14] W. Jiang and A. Ortega, "A parallel architecture for DWT based on lifting factorization," in *Proc. of SPIE in Parallel and Distributed Methods for Image Processing III*, July 1999.

[15] A. Ortega, W. Jiang, P. Fernandez, and C. Chrysafis, "Implementations of the discrete wavelet transform: complexity, memory, and parallelization issues," in *Proc. of SPIE: Wavelet Applications in Signal and Image Processing VII*, July 1999.

[16] W. Jiang and A. Ortega, "Efficient DWT system architecture design using filterbank factorizations," in *Proc. of ICIP*, Oct. 1999.

[17] F. Marino, V. Piuru, and E. Swartzlander Jr., "A parallel implementation of the 2D discrete wavelet transform without interprocessor communication," *IEEE Trans. Signal Proc.*, vol. 47, no. 11, pp. 3179–3184, Nov. 1999.

[18] *Proc. of The IEEE: Special Issue on Wavelets*, Number 4. 1996.

[19] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. on VLSI System*, vol. 1, no. 2, pp. 191–202, June 1993.

[20] A. S. Lewis and G. Knowles, "VLSI architecture for 2D daubechies wavelet transform without multipliers," *Electronic Letters*, vol. 27, no. 2, pp. 171–173, Jan. 1991.

[21] T. C. Denk and K. K. Parhi, "VLSI architectures for lattice structure based orthonormal wavelet transforms," *IEEE Trans. on CAS*, vol. 44, no. 2, pp. 129–132, Feb. 1997.

[22] L. Yang and M. Misra, "Coarse-grained parallel algorithms for multi-dimensional wavelet transforms," *The Journal of Supercomputing*, vol. 12, no. 1/2, pp. 99–118, 1998.

[23] J. Bowers, L. Keith, N. Aranki, and R. Tawel, "IMAS integrated controller electronics," Tech. Rep., Jet Propulsion Laboratory, Pasadena, CA, USA, Jan. 1998.

[24] O.M. Nielsen and M. Hegland, "TRCS9721: A scalable parallel 2D wavelet transform algorithm," Tech. Rep., The Australian National University, Dec. 1997.

[25] L. Yang and M. Misra, "Parallel wavelet transforms for image processing," in *ASAE Workshop on Application of Parallel Processing for Image Processing*, Aug. 1997.

[26] T. E. Anderson, D. E. Culler, D. A. Patterson, and the NOW team, "A case for NOW (Networks of Workstations)," *IEEE Micro*, Feb. 1995.

[27] *http://www.osc.edu/lam.html*.

[28] F. Kossentini, "Spatially segmented wavelet transform," Tech. Rep., UBC, 1998, ISOIEC JTC 1SC29WG1 WG1N868.

[29] V. Hardman M. A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the Iternet," in *Proc. INET*, 1995.

[30] R. Arean, J. Kovacevic, and V. K. Goyal, "Multiple description perceptual audio coding with correlating transforms," *IEEE Trans. on Speech and Audio Processing*, 2000.

[31] W.-T. Tan and A. Zakhor, "Multicast transmission of scalable video using receiver-driven hierarchical FEC," in *Proc. Packet Video Workshop*, New York, Apr. 1999.

[32] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource ReSerVation Protocol," *IEEE Network Mag.*, vol. 7, no. 5, pp. 8–18, Sept. 1993.

[33] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice Hall, 1983.

[34] A. S. Tanenbaum, *Computer Networks*, Prentice Hall, 1996.

[35] G. Davis and J. Danskin, "Joint source and channel coding for internet image transmission," in *Proc. of SPIE Conf. on Wavelet Applications of Digital Image Processing XIX*, Aug. 1996.

[36] J. M. Danskin, G. M. Davis, and X. Song, "Fast lossy internet image transmission," in *Proc. of 1995 ACM Multimedia Conference*, San Francisco, CA, 1995.

[37] P. G. Sherwood and K. Zeger, "Joint source and channel coding for internet image transmission," in *Proc. of DCC'97*, 1997.

[38] M. R. Karim, "Packetizing voice for mobile radio," *IEEE Trans. on Communications*, vol. 42, no. 2/3/4, pp. 377–385, Feb./Mar./Apr. 1994.

[39] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE Journal on Selected Areas in Communications, Special Issue on Multimedia Network Radios*, vol. 17, no. 5, pp. 756–773, May 1999.

[40] W. Jiang and A. Ortega, "A parallel architecture for dwt based on lifting factorization," in *Proc. of SPIE in Parallel and Distributed Methods for Image Processing III*, Denver, CO, July 1999.

[41] A. Ortega, W. Jiang, P. Fernandez, and C. Chrysafis, "Implementations of the discrete wavelet transform: complexity,memory, and parallelization issues," in *Proc. of SPIE in Wavelet Applications in Signal and Image Processing VII*, Denver, CO, Jul. 1999.

[42] W. Jiang and A. Ortega, "Efficient DWT system architecture design using filterbank factorizations," in *Proc. of Intl. Conf. on Image Processing*, Koba, Japan, Oct 1999.

[43] W. Jiang and A. Ortega, "Lifting-based discrete wavelet transform system architecture design," *submitted to IEEE Trans. on CSVT*, Oct 1999.

[44] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 11, no. 7, pp. 674–693, 1989.

[45] P. P. Vaidyanathan and P.-Q. Hoang, "Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, no. 1, pp. 81–94, Jan. 1988.

[46] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proceedings of The IEEE*, vol. 78, no. 1, pp. 56–93, Jan. 1990.

[47] T. G. Marshall, "Zero-phase filter bank and wavelet coder matrices: Properties, triangular decompositions, and a fast algorithm," *Multidimensional Systems and Signal Processing*, vol. 8, pp. 71–88, 1997.

[48] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247–269, 1998.

[49] R. Blahut, *Fast Algorithms for Digital Signal Processing*, Addison-Wesley Publishing Company, 1985.

[50] "Report on core experiment codeff1 "Complexity reduction of SSWT"," Tech. Rep., Motorola Australia, UBC, 1998, ISOIEC JTC 1SC29WG1 WG1N881.

[51] C. Chrysafis, "Low memory line-based wavelet trasform using lifting scheme," Tech. Rep., HP Labs, 1998, ISOIEC JTC 1SC29WG1 WG1N978.

[52] P. Onno, "Low memory line-based wavelet transform using lifting scheme," Tech. Rep., Canon Research Center France, 1998, ISOIEC JTC 1SC29WG1 WG1N1013.

[53] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels, and I. Bolsens, "Optimal memory organization for scalable texture codecs in MPEG-4," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, no. 2, pp. 218–243, Mar. 1999.

[54] D. Taubman, "Adaptive non-separable lifting transforms for image compression," in *Proc. ICIP*, Kobe, Japan, Oct. 1999.

[55] C. Chrysafis and A. Ortega, "Line based, reduced memory, wavelet image compression," *IEEE Trans. on Image Processing*, May 1999, Accepted for publication.

[56] G. Beylkin, R. Coifman, and V. Rokhlin, "Fast wavelet transforms and numerical algorithms I," *CPAM*, vol. 44, pp. 141–183, 1991.

[57] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Prentice-Hall PTR, Englewood-Cliffs, New Jersey, 1995.

[58] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Wavelet Applications in Signal and Image Processing III*, A. F. Laine and M. Unser, Eds. 1995, pp. 68–79, Proc. SPIE 2569.

[59] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1979.

[60] W. Jiang and A. Ortega, "Multiple description coding via scaling rotation transform," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing*, Phoenix, AZ, Mar. 1999.

[61] W. Jiang and A. Ortega, "Karhunen-loeve vector transform KLVT for vector quantization," Tech. Rep., Signal and Image Processing Institute, University of Southern California, 1996.

[62] Y. Wang, M. Orchard, and A. R. Reibman, "Multiple description image coding for noisy channels by paring transform coefficients," in *Proc. IEEE 1997 First Workshop on Multimedia Signal Processing*, 1997.

[63] M. T. Orchard, Y. Wang, V. Vaishampayan, and A. R. Reibman, "Redundancy rate-distortion analysis of multiple description coding using pairwise correlating transforms," in *ICIP'97*, 1997.

[64] V. K. Goyal and J. Kovacevic, "Optimal multiple description transform coding of Gaussian vectors," in *Proc. of IEEE Data Compression Conference*, 1998.

[65] V. K. Goyal, J. Kovacevic, and M. Vetterli, "Multiple description transform coding: Robustness to erasures using tight frame expansions," in *Proc. of IEEE int. Symp. Info. Th.*, Aug. 1998.

[66] V. K. Goyal, J. Kovacevic, R. Arean, and M. Vetterli, "Multiple description transform coding of images," in *ICIP'98*, 1998.

[67] J. Kovacevic, "Multiple descriptions as joint source channel codes," 1998.

[68] W. Li, "Vector transform and image coding," *IEEE Trans. on CAS for VT*, vol. 1, no. 4, pp. 297–307, Dec 1991.

[69] X. Xia and B.W. Suter, "On vector KL transform," *IEEE Trans. on CAS for VT*, vol. 5, no. 4, pp. 372–374, Aug 1995.

[70] W. Ding, "Optimal vector transform for vector quantization," *IEEE Signal Processing Letters*, vol. 1, no. 7, pp. 110–113, July 1994.

[71] G.H. Golub and C.F.Van Loan, *Matrix Computations*, John Hopkins University Press, 1996.

[72] Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall Information and System Sciences Series. Prentice Hall, Englewood Cliffs, NJ 07632, 1989.

[73] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 2nd edition, 1992.

[74] W. Jiang and A. Ortega, "Multiple description coding via polyphase transform and selective quantization," in *Proc. of Visual Communication and Image Processing*, San Jose, CA, Jan. 1999.

[75] W. Jiang and A. Ortega, "Multiple description speech coding for robust communication over lossy packet networks," in *Proc. of Intl. Conf. on Multimedia Engineering*, New York, NY, July 2000.

[76] N. S. Jayant and S. W. Christensen, "Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure," *IEEE Trans. Communications*, vol. COM-29, no. 2, pp. 101–109, Feb. 1981.

[77] J-C. Bolot and A. V. Garcia, "Control mechanisms for packet audio in the Internet," in *Proc. IEEE INFOCOM'96*, 1996, pp. 232–239.

[78] J.-C. Bolot and A. V. Garcia, "The case for FEC-based error control for packet audio in the internet," in *to appear ACM Multimedia Systems*, 1997.

[79] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for Internet telephony," in *Proc. IEEE INFOCOMM'99*, 1999, vol. 3, pp. 1453–1460.

[80] M. Podolsky, C. Romer, and S. McCanne, "Simulation of FEC-based error control for packet audio on the internet," in *INFOCOM'98*, San Francisco, CA, Mar. 1998.

[81] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. on Multimedia*, vol. 1, no. 2, pp. 172–186, June 1999.

[82] P. A. Chou, A. E. Mohr, S. Mehrotra, and A. Wang, "FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video," in *Proc. of The Data Compression Conf.*, Mar. 2000.

[83] A. A. El-Gamal and T. M. Cover, "Achievable rates for multiple descriptions," *IEEE Trans. Information Theory*, vol. IT-28, no. 6, pp. 851–857, Nov. 1982.

[84] J. K. Wolf, A. D. Wyner, and J. Ziv, "Source coding for multiple descriptions," *The Bell System Technical Journal*, vol. 59, no. 8, pp. 1417–1426, Oct. 1980.

[85] L. Ozarow, "On a source-coding problem with two channels and three receivers," *The Bell System Technical Journal*, vol. 59, no. 10, pp. 1909–1921, Dec. 1980.

[86] N. S. Jayant, "Subsampling of a DPCM speech channel to provide two "self-contained" half-rate channels," *The Bell System Technical Journal*, vol. 60, no. 4, pp. 501–509, Dec. 1981.

[87] Z. Zhang and T. Berger, "New results in binary multiple descriptions," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 502–521, 1987.

[88] Z. Zhang and T. Berger, "Multiple description source coding with no excess marginal rate," *IEEE Trans. Inform. Theory*, vol. 41, no. 2, pp. 349–357, 1995.

[89] A. Ingle and V. A. Vaishampayan, "DPCM system design for diversity systems with applications to packetized speech," *IEEE Trans. Speech and Audio Processing*, vol. 3, no. 1, pp. 48–58, 1995.

[90] S. D. Servetto, K. Ramchandran, V. Vaishampayan, and K. Nahrstedt, "Multiple description wavelet based image coding," in *ICIP'98*, 1998.

[91] V. A. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Trans. Information Theory*, vol. 39, no. 3, pp. 821–834, 1993.

[92] V. A. Vaishampayan and J.-C. Batllo, "Asymptotic analysis of multiple description quantizers," *IEEE Trans. on Information Theory*, vol. 44, no. 1, pp. 278–284, Jan. 1998.

[93] C. Chrysafis and A. Ortega, "Efficient context-based entropy coding for lossy wavelet image compression," in *Proc. of DCC'97*, Snowbird, UT, Mar. 1997.

[94] A. C. Miguel, A. E. Mohr, and E. A. Riskin, "SPIHT for generalized multiple description coding," in *Proc. of ICIP*, 1999.

[95] R. Puri, K. Ramchandran, and I. Kozintsev, "Multiple description source coding using forward error correction (fec) codes," 1999.

[96] A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Generalized multiple description coding through unequal loss protection," in *Proc. of ICIP*, 1999.

[97] A. Said and W. A. Pearlman, "Low-complexity waveform coding via alphabet and sample-set partitioning," in *Proc. of VCIP'97*, San Jose, CA, Jan. 1997.

[98] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Prentice Hall Information and System Sciences Series. Prentice Hall, Englewood Cliffs, NJ 07632, 1991.

[99] S. M. LoPresto, K. Ramchandran, and M. T. Orchard, "Wavelet image coding using rate-distortion optimized backward adaptive classification," in *Proc. of VCIP'97*, San Jose, CA, Jan. 1997.

[100] X. Wang, G. Wu, and X. Lin, "A zerotree based multirate wavelet image coding method," *Acta Electronica Sinica*, vol. 25, no. 4, pp. 48–51, Apr. 1997.

[101] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 4, pp. 243–250, June 1996.

[102] D. E. Tsoukalas, J. N. Mourjopoulos, and G. Kokkinakis, "Speech enhancement based on audio noise suppression," *IEEE Trans. on speech and audio processing*, pp. 497–514, 1997.

[103] W. Jiang and H. Malvar, "Adaptive speech noise removal," Tech. Rep., Microsoft Research, Aug. 1999.

[104] S. R. McCanne, "Scalable compression and transmission of internet multicast video," Tech. Rep. Ph.D. Thesis, UCB/CSD-96-928, University of California, Berkeley, 1996.