

國立臺灣大學電機資訊學院電子工程學研究所

碩士論文

Graduate Institute of Electronics Engineering

College of Electrical Engineering & Computer Science

National Taiwan University

Master Thesis

分散式視訊解碼器之設計與實現

Design and Implementation of a
Distributed Video Decoder

何孟宣

Ho, Meng-Hsuan

指導教授：陳少傑 博士

Advisor: Chen, Sao-Jie, Ph.D.

中華民國 103 年 10 月

October, 2014





誌 謝

可以完成論文並且順利取得碩士學位，要特別感謝指導教授陳少傑教授，多虧指導教授精闢的見解與指導建議，使研究過程中的種種難題都得以迎刃而解，感謝簡韶逸教授提供我們研究的資源和建議，感謝口試委員們在學位考試中對於研究主題所提出的寶貴意見使本篇論文更佳完善。

研究所在學期間受到各位學長姐以及同學們的照顧與協助，感謝鑫平學長在研究上不吝指導與耐心鼓勵，以及實驗室的學長姐：盈如、逸霏的教導。還有 99 級的學長們：和昇、居翰、翊耀，100 級的學長們：仁豪、俊儒、承志、懷傑，在課業和研究上的幫助和鼓勵。101 級的同学們：勝翔、偉澤、瀚漳、榮鴻、柏叡，在課業或研究上都是很好的夥伴，一起度過煎熬的時刻。102 級的學弟們：晟豪、詩堯、岱鑫、鼎鈞，有你們的加入讓實驗室充滿朝氣與活力。

最後要感謝我的父母和家人，在求學期間給予的支持，以及鼓勵我完成這份學業。

謹以本文感謝所有關心我的人，願與你們分享這份榮耀。





中文部分





摘要

在傳統視訊編碼中有許多標準化的視訊編碼格式，像是動態影像專家組 (Moving Picture Experts Group) 提出的 MPEG 格式、國際電信聯盟電信標準化部門提出的 H. 26X 格式等等，這些傳統視訊編碼的特點是盡量讓編碼器探索冗餘的時間和空間資訊，也就是使用影格內編碼或影隔間編碼來壓縮視訊，在這樣架構下的編碼器負擔很大。但是傳統的視訊編碼不適用於只擁有低運算複雜度的編碼器或考量到低功耗的編碼器，像是無線視訊感測器或監視系統，分散式視訊編碼技術因此被提出，分散式視訊編碼的根本以 Slepian-Wolf 理論和 Wyner-Ziv 理論為基礎。

在此論文中，我們設計並實現一個分散式視訊解碼器，提出的架構包含累積低密度奇偶檢查碼解碼器、相關雜訊模型、軟輸入計算、邊資訊產生，實作採用 TSMC 90nm GUTM 製程完成分散式視訊解碼器，針對 QCIF 大小的測試序列可解碼達到每秒 30 張的影像，此晶片工作頻率可達到 100MHz，使用的邏輯閘數目為 690K，晶片大小為 4.67 mm^2 。

關鍵字：分散式視訊解碼器、累積低密度同位檢查碼解碼器、邊資訊產生。





目 錄

第一章	簡介	伍
第二章	分散式視訊編碼概述	陸
第三章	架構和實現.....	柒
第四章	實驗結果.....	捌
第五章	結論	玖





第一章

簡介

在傳統視訊編碼中，許多編碼技術被提出用來探索時間或空間上冗余的資訊，這些編碼技術主要著重在盡可能壓縮視訊資料，像是 H.264/AVC 為了盡量縮減時間或空間上冗余的資訊，所有複雜度高的運算會集中於編碼器，H.264/AVC 的架構適用於編碼一次而多次解碼的視訊廣播服務，但是卻不適用於具有許多運算資源有限的編碼器和少量運算能力較強的解碼器的架構，像是無線視訊感應器，這些考量到低運算能力或低功耗需求的設備，因此才有一種新的視訊架構—分散式視訊編碼器被提出。

本論文著重在分散式視訊解碼器的設計與實現，我們會在第二章回顧先前的分散式視訊編解碼器的架構，第三章會呈現我們設計的分散式視訊解碼器的架構，第四章則是實驗結果以及晶片設計流程，第五章對研究做總結。



第二章

分散式視訊編碼概述

此章節會回顧具有代表性的分散式視訊編碼的架構，包含第一個被提出的像素域(Pixel Domain)的分散式視訊編碼架構、及後來發展成轉換域(Transform Domain)的架構，從其結果看出轉換域編解碼器的失真率表現會比像素域編解碼器的失真率表現佳，這是因為使用離散餘弦轉換探索空間上的相關性。之後具代表性的 DISCOVER(Distributed Coding for Video Services)被提出，近來混合型分散式視訊(Hybrid DVC)編碼器被提出，其編碼效率以及失真率表現更上一層。在此章節會依序描述以下的編解碼器架構：第一個被提出的分散式視訊編解碼器架構、DISCOVER 分散式視訊編解碼器架構、混合型分散式視訊編解碼器架構。



第三章 架構和實現

許多分散式視訊編解碼器的架構被提出，這些架構主要藉由增加解碼器的元件來改善失真率表現，但同時也增加解碼過程的複雜度，高複雜度會造成解碼器無法即時解碼視訊。因此我們根據混合式分散式視訊編解碼器提出的架構，並且以實現硬體的方式來改善問題，在之後的章節會展示所提出分散式視訊解碼器的架構，以及各元件的架構，分別是：累積低密度同位檢查碼解碼器、邊資訊產生器、離散餘弦反變換和反量化器、離散餘弦變換和量化器、相關雜訊模型、軟輸入計算。



第四章 實驗結果

在此章節會呈現提出的分散式視訊解碼器的實驗結果，首先會先描述如何從演算法階段至晶片製作的設計流程，接著是如何使用機台測試我們的晶片，以及呈現晶片的規格，最後則是不同測試序列的實驗結果。



第五章

結論

本論文實現即時解碼的分散式視訊解碼器，並呈現此解碼器的硬體架構，根據模擬結果我們架構的失真率表現比 DISCOVER 佳，在此研究的關鍵挑戰是邊資訊產生器和累積低密度同位檢查碼解碼器的運算複雜度高，因此會考量到硬體成本和失真率表現的取捨。晶片透過國家晶片系統設計中心下線並使用 TSMC 90nm GUTM 製程，解碼器晶片可解碼每秒 30 張 QCIF 大小的視訊，製作的晶片時脈可達 100MHz，面積為 4.67mm^2 ，邏輯閘數目為 690K。





英文部分





ABSTRACT

In conventional video coding, lots of coding techniques were proposed to exploit redundancy of the spatial and temporal signals in the encoder. These coding techniques are focused on compressing video data as high as possible, such as MPEG (Moving Picture Experts Group) and H.26X proposed by International Telecommunication Union. These conventional techniques tried to extremely reduce spatial and temporal redundancies, so all the computing efforts were majorly spent on the encoder. Conventional video coding is not suitable for the situations where little encoders and big decoders are used, like wireless video sensors that have a low computing capability and need a low power consumption. To meet the requirement of a low-complexity and low-power encoder, a new video coding paradigm, distributed video coding, based on Slepian-Wolf Theorem and Wyner-Ziv Theorem, was proposed.

In this Thesis, we design and implement a distributed video decoder which is majorly composed of LDPCA, correlation noise modeling, soft input computation, and side information creation. Our proposed DVC decoder, implemented in TSMC 90nm GUTM process technology, can meet the requirement of decoding a QCIF video with a speed of 30fps. The maximum operation frequency is 100MHz, the chip area is 4.67 mm², and gate count is 690K.

Keywords: Distributed Video Decoder, LDPCA Decoder, Side Information Creation.

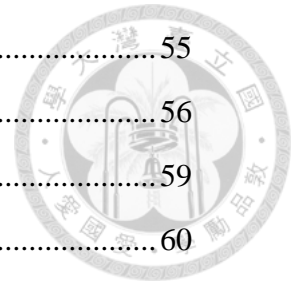




TABLE OF CONTENTS

ABSTRACT	i
LIST OF FIGURES	v
LIST OF TABLES	vii
CHAPTER 1 INTRODUCTION	1
1.1 Slepian-Wolf Theorem and Wyner-Ziv Theorem	1
1.2 Wyner-Ziv Coder	3
1.3 Distributed Video Coding	5
1.4 Motivation	6
1.5 Thesis Organization	7
CHAPTER 2 OVERVIEW OF DISTRIBUTED VIDEO CODING	9
2.1 First Practical DVC	9
2.2 DISCOVER DVC	11
2.3 Hybrid DVC	15
CHAPTER 3 PROPOSED ARCHITECTURE AND IMPLEMENTATION ...	21
3.1 Proposed Architecture of DVC	21
3.2 LDPCA Decoder	25
3.3 Side Information Creation	38
3.3.1 Spatial Motion Smoothing	39
3.3.2 BiDirectional Motion Compensation	44
3.3.3 Interpolation	46
3.4 CAVLC Decoder	47
3.5 DCT, IDCT, Quantization, and De-Quantization	49
3.6 Correlation Noise Modeling and Soft Input Computation	52
3.7 Reconstruction	53
CHAPTER 4 EXPERIMENT RESULTS	55

4.1	Design Flow.....	55
4.2	Testing Consideration	56
4.3	Chip Implementation Result.....	59
4.4	Simulation Results.....	60
CHAPTER 5	CONCLUSION.....	65
REFERENCE	67

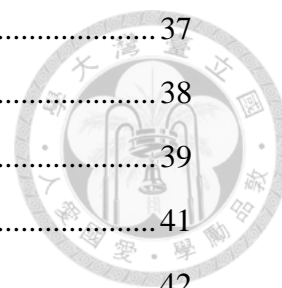




LIST OF FIGURES

Fig. 1-1.	Correlated Source Coding Configuration	2
Fig. 1-2.	Achievable Rate Region for Dependent X and Y	2
Fig. 1-3.	Entropies of X and Y	3
Fig. 1-4.	A Practical Wyner-Ziv Coder.....	3
Fig. 1-5.	Tanner Graph and Accumulated Check Nodes	4
Fig. 1-6.	Parity-Check Matrix of the LDPCA in Fig. 1-5.....	5
Fig. 1-7.	Achieving Different Source Rates by Accumulating Check Nodes	5
Fig. 2-1.	First Practical DVC Codec.....	11
Fig. 2-2.	DISCOVER Codec	12
Fig. 2-3.	Frame Interpolation Framework	12
Fig. 2-4.	Motion Vectors (a) Before and (b) After Spatial Motion Smoothing	13
Fig. 2-5.	The Hybrid DVC Framework	15
Fig. 3-1.	Proposed Architecture of DVC	22
Fig. 3-2.	Proposed System Architecture and Data Flow.....	24
Fig. 3-3.	Tanner Graphs	26
Fig. 3-4.	Tanner Graphs with Accumulated Syndrome Bits.....	27
Fig. 3-5.	Parity-Check Matrix of the LDPCA shown in Fig. 3.2 (a)	28
Fig. 3-6.	Parity-Check Matrix of the LDPCA shown in Fig. 3.3.....	28
Fig. 3-7.	Proposed Architecture of Comparing Tree	31
Fig. 3-8.	Schematic Diagram of Wires Connected to VNs and BNs.....	33
Fig. 3-9.	A 3-Input Minimum-Value Generator.....	33
Fig. 3-10.	A 4-Input Minimum-Value Generator.....	34
Fig. 3-11.	An Example of Comparing Tree Consisting of Check Nodes	35
Fig. 3-12.	An Example of Merging Check Nodes in Code Rate of 4/6.....	36
Fig. 3-13.	An Example of Merging Check Nodes in Code Rate of 5/6.....	36

Fig. 3-14. Proposed Architecture of LDPCA Decoder	37
Fig. 3-15. State Diagram of Proposed LDPCA Decoder	38
Fig. 3-16. Proposed Framework of Side Information Creation	39
Fig. 3-17. Proposed Architecture of Weighted Vector Median Filter	41
Fig. 3-18. Architecture of Datapath of Distance	42
Fig. 3-19. Architecture of Datapath of SAD	43
Fig. 3-20. Architecture of Weighing & Minimum Selector.....	44
Fig. 3-21. Motion Vectors of (a) Forward and (b) Backward Compensations	45
Fig. 3-22. Architecture of Motion Compensation	46
Fig. 3-23. Architecture of Interpolation.....	47
Fig. 3-24. Decoding Flow of CAVLC Decoder	48
Fig. 3-25. Block Diagram of CAVLC Decoder	49
Fig. 3-26. Hardware of Direct 2-D Transform.....	51
Fig. 3-27. Architectures of DCT, IDCT, Quantization, and De-Quantization.....	52
Fig. 3-28. Hardware of Correlation Noise Modeling and Soft Input Computation....	53
Fig. 3-29. Hardware of Reconstruction	53
Fig. 4-1. Cell-based Design Flow	56
Fig. 4-2. Uncollapsed Stuck Fault Summary Report.....	57
Fig. 4-3. Measurement Considerations	58
Fig. 4-4. Measurement Flow.....	58
Fig. 4-5. Result of Place and Route	59
Fig. 4-6. RD Performance of Coastguard QCIF 15fps	61
Fig. 4-7. RD Performance of Foreman QCIF 15fps	62
Fig. 4-8. RD Performance of Hall QCIF 15fps	62
Fig. 4-9. Rate Performance of Regular Degree-3 LDPCA of Length 396	63





LIST OF TABLES

Table. 1-1.	Decoding Time for Foreman QCIF 150 Frames.....	6
Table. 2-1.	Coding Modes and Decision Rules.....	18
Table. 3-1.	Transmitted Nodes in Different Code Rates.....	30
Table. 4-1.	Specification Table	60
Table. 4-2.	Synthesis Result of LDPCA	60





CHAPTER 1

INTRODUCTION

In conventional video coding, lots of coding techniques were proposed to exploit redundancy of the spatial and temporal signals in the encoder. These coding techniques are focused on compressing video data as high as possible, such as H.264/AVC [1]. To extremely reduce spatial and temporal redundancy, all the computing efforts are majorly spent on the encoder. The H.264/AVC architecture is suitable for the situation of broadcasting video services that are encoded once and decoded many times, while they are not suitable for the situation of using little encoders and big decoders, like wireless video sensors that have a low computing capability and need a low power consumption. To meet the requirement of low-complexity and low-power encoders, a new video coding paradigm, distributed video coding (DVC) [2], was proposed.

1.1 Slepian-Wolf Theorem and Wyner-Ziv Theorem

DVC is based on two theorems, Slepian-Wolf Theorem [3] and Wyner-Ziv Theorem [4], where separately encoded and correlated sources could be jointly decoded with a rate of the joint entropy in theory.

Consider two correlated sources X and Y , which are respectively encoded with R_X and R_Y bit rates and transmitted to the decoder from the encoders as shown in Fig. 1-1.

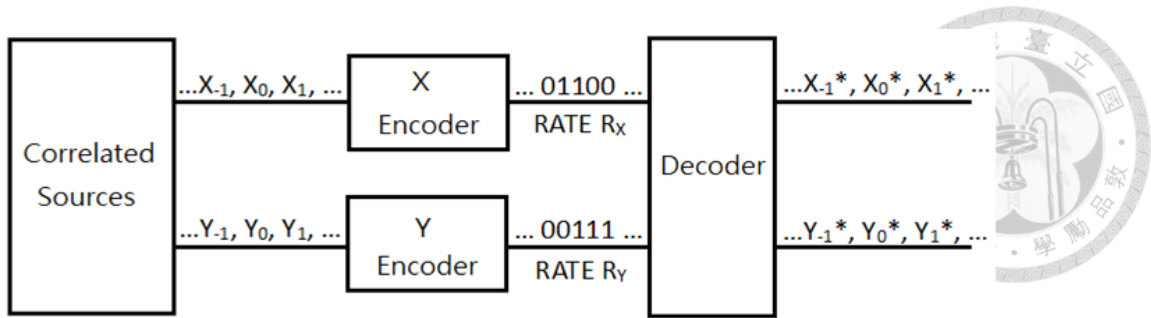


Fig. 1-1. Correlated Source Coding Configuration [3].

The rate region of the Slepian-Wolf Theorem is shown in Fig. 1-2, where $H(X)$ and $H(Y)$ are respectively the individual entropies of X and Y , $H(X|Y)$ and $H(Y|X)$ are respectively the conditional entropies of X and Y , and $H(X,Y)$ is the joint entropy of X and Y .

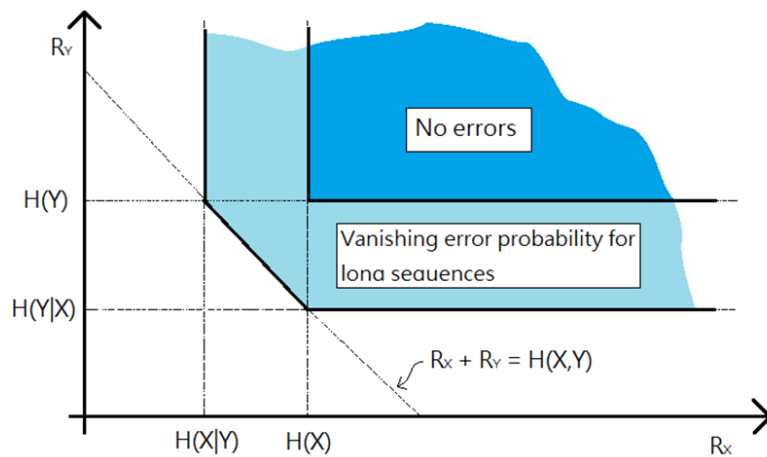


Fig. 1-2. Achievable Rate Region for Dependent X and Y [2].

Figure 1-3 states the relationship of different entropies where $I(X;Y)$ is the mutual entropy of X and Y . The region as shown in Fig. 1-2 can be expressed by the following equations:

$$R_X + R_Y \geq H(X,Y) \quad (1-1)$$

$$R_X \geq H(X|Y), R_Y \geq H(Y|X) \quad (1-2)$$

In spite of encoding X and Y separately, Slepian-Wolf Theorem states that the total rate can be reduced to a joint entropy by joint decoding.

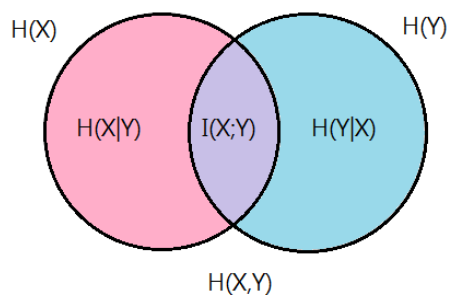


Fig. 1-3. Entropies of X and Y .

Slepian-Wolf Theorem can be extended to perform lossy coding within a bounded distortion. Assume that source X is encoded in the encoder without referring to the source Y , Wyner-Ziv Theorem proved that there is a lower bounded distortion between source X and the reconstructed data X' in the decoder. In practical applications, Wyner-Ziv coder consists of a Slepian-Wolf coder, a quantizer, and a distortion reconstructor as shown in Fig. 1-4.

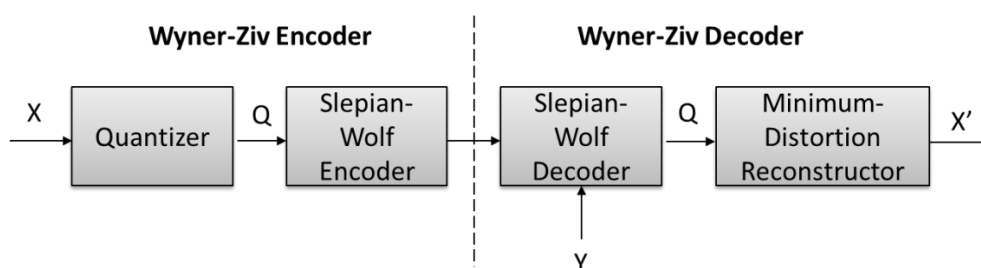


Fig. 1-4. A Practical Wyner-Ziv Coder [2].

1.2 Wyner-Ziv Coder

To implement a practical Wyner-Ziv coder and to obtain less RD performance loss, choosing a proper error correcting code is important. Low-density parity-check accumulate (LDPCA) [4] will be the core of Wyner-Ziv coder in our work for its

ability of error correcting can result in better Rate-Distortion (RD) performance [5].

LDPCA is presented by a tanner graph where solid circles and solid squares are used to represent bit nodes and check nodes respectively, and dotted circles are for accumulating check nodes as shown in Fig. 1-5. And Fig. 1-6 depicts its parity-check matrix. The reason of using LDPCA is that different video sequences require different proper source rates. Though high code rate always guarantees decoding the side information (SI) successfully, but it can cause excessive bit rate if the conditional entropy of the encoded information and SI is low.

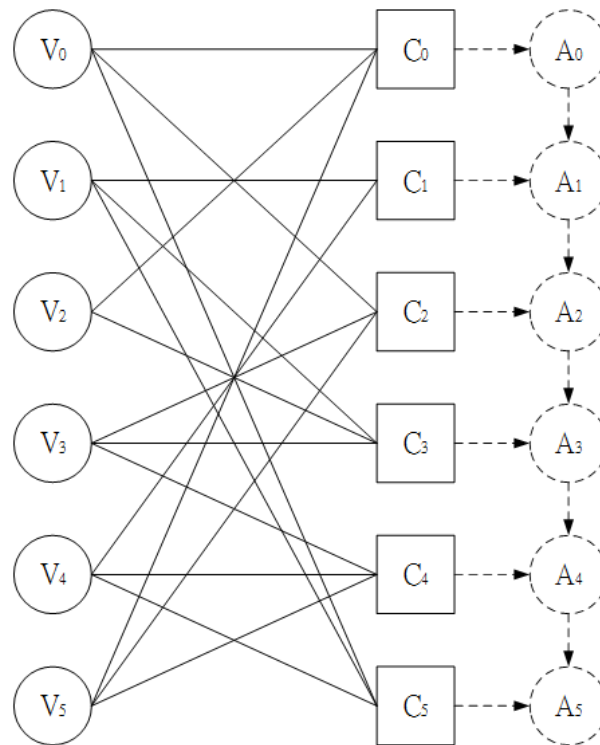


Fig. 1-5. Tanner Graph and Accumulated Check Nodes.

$$H = \begin{matrix} & V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} & C_0 & C_1 & C_2 & C_3 & C_4 & C_5 \end{matrix}$$



Fig. 1-6. Parity-Check Matrix of the LDPCA in Fig. 1-5.

The method of accumulating syndromes make decoding SI with syndromes of different source rates possible. For example, we can change the source rate from 6/6 to 5/6 by accumulating C0 and C1 in Fig. 1-7.

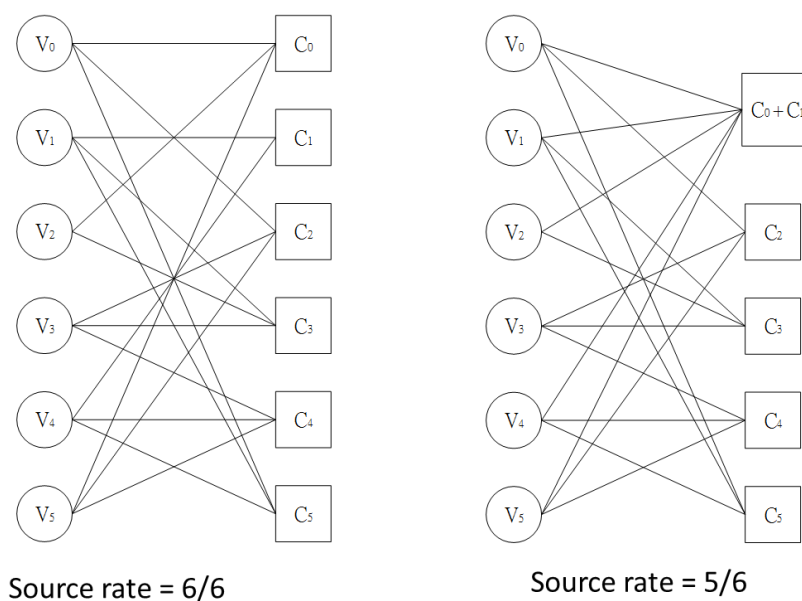


Fig. 1-7. Achieving Different Source Rates by Accumulating Check Nodes.

1.3 Distributed Video Coding

In a typical DVC, there are two types of coding frames, Wyner-Ziv (WZ) frames and key frames. The key frames are coded by an H.264/AVC [1] Intra encoder. The

WZ frames are coded through a WZ encoder. In the DVC encoder, WZ frames will be transformed by a discrete cosine transform (DCT). The quantizer keeps the data of different frequencies according to a selected quantization factor. Then, syndromes created by the LDPCA encoder will be stored in a buffer. In the decoder, periodically inserted key frames will be referred to generate SI using bidirectional motion estimation and motion compensation. Then, frames are reconstructed by the LDPCA decoder from SI and received syndromes.

1.4 Motivation

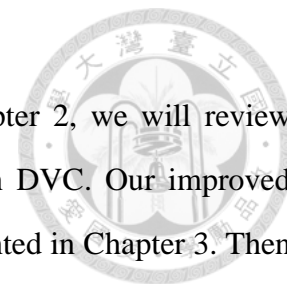
The architecture of distributed video coding is different from the architecture of conventional video coding since many parts with high computing complexity like LDPCA decoder and side information creation are used in the distributed video decoder. We evaluate respective computing complexities by running a Hybrid DVC decoding software with a test sequence of Foreman in QCIF format and 150 frames on a server with an Intel i7-2600 CPU. Table 1-1 shows the decoding time with different quality parameters (QPs). When QP is 8, frames can be only decoded in 0.156 fps, 150 frames divided by 958.9205 seconds. Hence, we will focus on implementing a hardware of DVC decoder to decode frames in real time.

Table 1-1. Decoding Time for Foreman QCIF 150 Frames.

Quality Parameter	Decoding Time (Sec.)
1	42.3858
2	75.6814
3	75.9458
4	123.5452
5	123.3386
6	186.7416
7	390.6234
8	958.9205

1.5 Thesis Organization

The rest of this Thesis is organized as follows. In Chapter 2, we will review previous architectures of DVC including components used in DVC. Our improved architecture of DVC with better RD performance will be presented in Chapter 3. Then, Chapter 4 will discuss the experiment results of our proposed DVC implementation. In Chapter 5, we make a conclusion on our work.







CHAPTER 2

OVERVIEW OF DISTRIBUTED VIDEO CODING

In this chapter, we will review some previous, representative architectures of DVC including components used in DVC. One of the first practical pixel domain DVC codec was proposed in [6]. Then, the pixel domain codec was evolved into a transform domain scheme [7]. The results of above designs showed that the RD performance of a transform domain codec is better than a pixel domain codec by using DCT to exploit spatial correlation of each frame and truncating less important bit planes to save bit rate. A more efficient DVC codec, DISCOVER [5] (Distributed Coding for Video Services), a project resulted from the IST (Information Society Technologies) FET (Future and Emerging Technologies) program of the European Union, was proposed later. Recently, a new Hybrid DVC design with residual coding and frame level coding mode selection [8] was proposed and achieved high-efficiency coding compared to DISCOVER. In the following sections, we will review advance of transform domain codecs from the first practical DVC [7], DISCOVER DVC [5], to Hybrid DVC [8].

2.1 First Practical DVC

The first practical DVC was proposed in [7] which is a basic Wyner-Ziv codec consisting of an intraframe encoder and an interframe encoder as shown in Fig. 2-1, where video sequences are classified into Wyner-Ziv frames and Key frames. Key frames are encoded and decoded by a conventional intra codec, while Wyner-Ziv frames are coded by a Wyner-Ziv codec. Wyner-Ziv frames will first be processed by Discrete Cosine Transform (DCT) to separate the frequencies into spatial signals. Signals of high frequency can usually be truncated, for most of the signals with a larger energy are located in the lower frequency range. After truncated by the

Quantizer, bit planes split by Extract Bit-planes will be coded by a channel encoder, the Turbo Encoder, then being stored in a Buffer. The whole encoding process only exploits the spatial correlation within each frame, hence called Intraframe Encoder. In Interframe Decoder, side information is generated by Interpolation and Extrapolation. Interpolation is a method where the side information is generated by referring to the previous key frame and the next key frame, while the method of Extrapolation is where the side information is generated by referring to two previous key frames. The key frames are encoded by a Conventional Intraframe Encoder and are decoded by a Conventional Intraframe Decoder. And both of Interpolation and Extrapolation use reference frames to estimate motion vectors, then generate the side information by motion compensation. The methods of generating side information is the basic idea proposed by the first practical DVC, which provided significant improvement of RD performance in some test sequences. After generating the side information, a channel decoder, the Turbo Decoder, will be used to correct errors in the side information. Finally, each bit plane of the Wyner-Ziv frames is reconstructed by Reconstruction, processed by an Inverse Discrete Cosine Transform (IDCT) and used to make up the decoded Wyner-Ziv Frames. This architecture provides a basis of DVC with the idea of generating side information through Interpolation and Extrapolation.

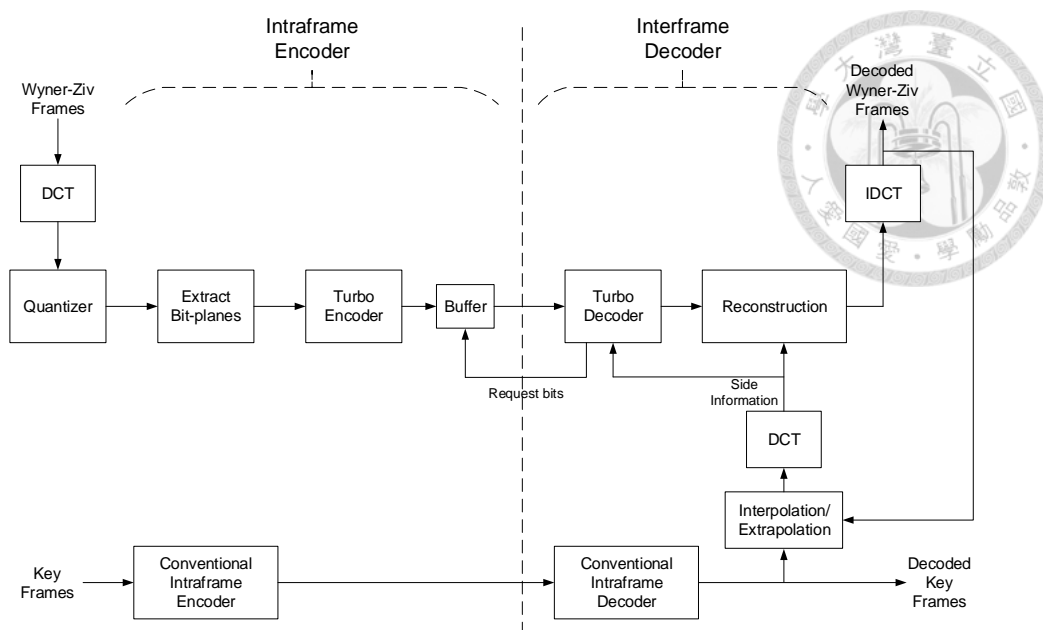


Fig. 2-1. First Practical DVC Codec [7].

2.2 DISCOVER DVC

The performance of DISCOVER DVC [5] is improved significantly for adopting some new methods including LDPCA codec, Soft Input Computation, and Virtual Channel Model. The architecture of DISCOVER DVC is shown in Fig. 2-2.

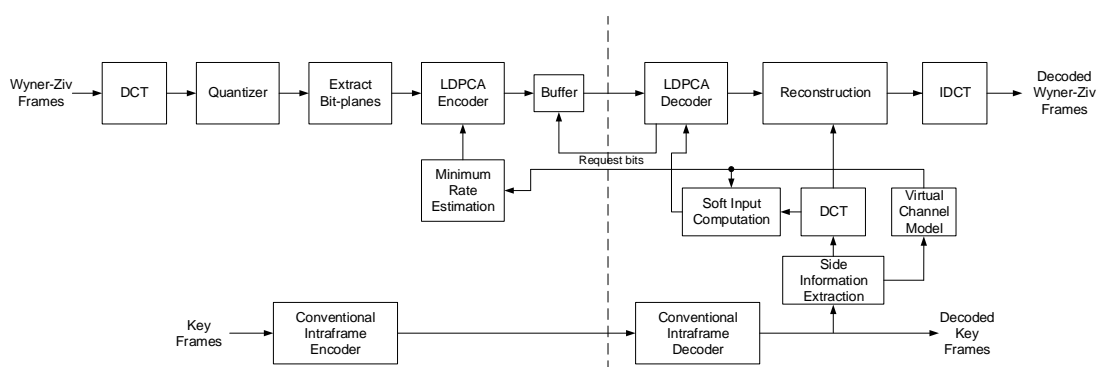


Fig. 2-2. DISCOVER Codec [5].

The parts of encoder including DCT, Quantizer, Extract Bit-planes, Buffer, and Conventional Intraframe Encoder are the same as the first practical DVC mentioned in Section 2.1. And the parts of decoder including Reconstruction, IDCT, and Conventional Intraframe Decoder are also the same as the first practical DVC.

Channel coding uses an LDPCA coder which consists of an LDPCA Encoder and an LDPCA Decoder because of its better ability of correcting errors. And Minimum Rate Estimation is used to estimate the initial source rate of LDPCA such that the total number of iterations in the LDPCA Decoder could be reduced.

The quality of side information has a great impact on the RD performance of Wyner-Ziv frames, so choosing appropriate techniques to implement Side Information Extraction and to generate side information is important. Thus, a Frame Interpolation Framework including some techniques proposed in [9] was applied to improve the accuracy of motion-compensated side information as shown in Fig. 2-3.

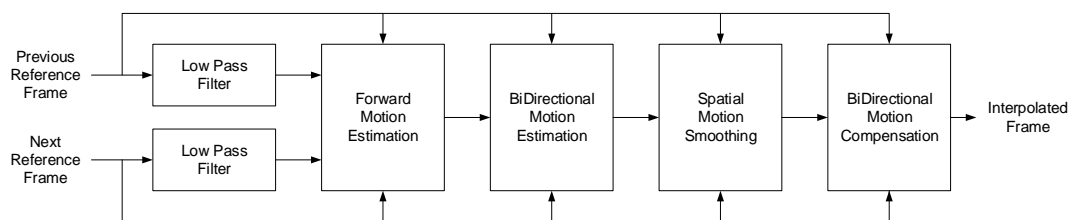


Fig. 2-3. Frame Interpolation Framework [9].

First, the input Previous Reference Frame and Next Reference Frame are processed by a Low Pass Filter. The function of Low Pass Filter in DVC is to average the values in pixel domain within a specific region. Then, Forward Motion Estimation is used to find motion vectors approximate to the true motion vectors between low pass filtered Previous Reference Frame and low-pass filtered Next Reference Frame. The Bidirectional Motion Estimation refines the motion vectors within a small offset which could be half pixel or quarter pixel of the approximate motion vectors. The method used to find appropriate motion vectors of each block is to choose the smallest SAD of each block between the compensated previous reference frame and the compensated next reference frame. Then, an adaptively weighted vector-median filter [10] is used for Spatial Motion Smoothing. The function of Motion Smoothing is to find more smoothing motion vectors for each block within a specified region. The technique could be applied to the situations as depicted in Fig. 2-4, where the original

motion vectors were just obtained by minimum SAD (Fig. 2-4(a)) while Spatial Motion Smoothing considered not only minimum SAD but also adjacent motion vectors to refine motion vectors (Fig. 2-4 (b)).

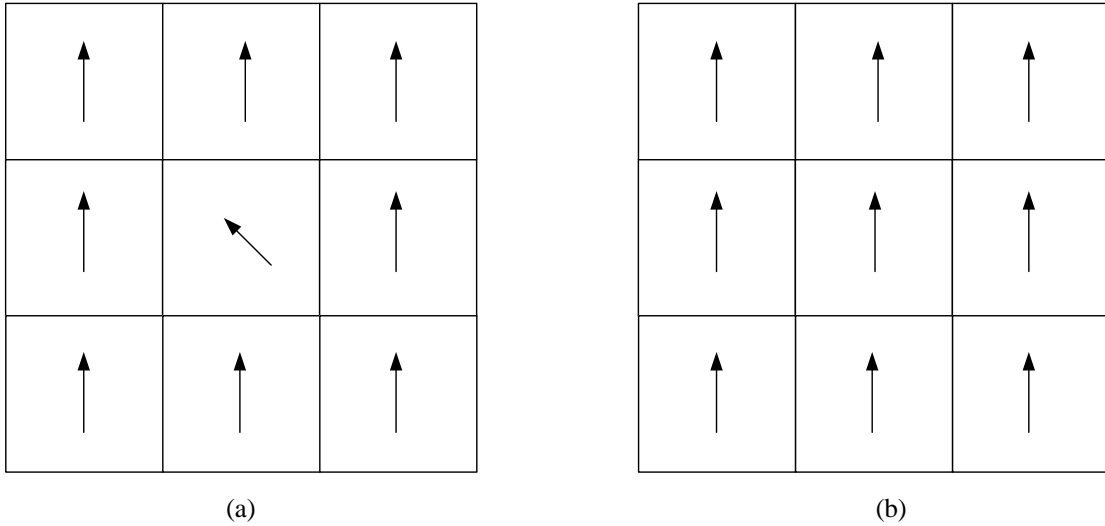


Fig. 2-4. Motion Vectors (a) Before and (b) After Spatial Motion Smoothing.

The criterion is defined by the following equation:

$$\sum_{j=1}^N MSE(x_{wvmf}, B)(|X_{wvmf} - X_j| + |Y_{wvmf} - Y_j|) \leq \quad (2-1)$$

$$\sum_{j=1}^N MSE(x_c, B)(|X_c - X_j| + |Y_c - Y_j|),$$

where x_{wvmf} is the motion vector obtained at the output of the weighted vector-median filters, N is the number of motion vectors within a specified region, x_c is the current appropriate motion vector, X and Y are x-dimension and y-dimension values of motion vectors respectively, $MSE(x, B)$ means the mean square error of the motion vector x of current block compensated by previous and next reference frames. Finally, Bidirectional Motion Compensation is applied, where the side information consists of a Previous Reference Frame compensated by forward motion and a Next Reference Frame compensated by backward motion, and the motion vectors are the

output of this Bidirectional Motion Estimation.

The Virtual Channel Model adopted in DISCOVER DVC is a Laplacian distribution as proposed in [11, 12]. The Laplacian distribution used in correlation noise modeling is described by the following probability density function:

$$p[WZ(x, y) - SI(x, y)] = \frac{\alpha}{2} \exp[-\alpha |WZ(x, y) - SI(x, y)|], \quad (2-2)$$

where $WZ(x, y)$ is the pixel of Wyner-Ziv frames in the position of (x, y) , $SI(x, y)$ is the pixel of side information in the position of (x, y) , and α is the Laplacian distribution parameter. The Laplacian distribution parameter is calculated at both the DCT band level and the coefficient level where the former means the parameter being associated with a DCT band of each frame and the latter means the parameter being associated with a DCT band of each block in each frame. We will describe calculation of the Laplacian distribution parameter at the DCT band level. To calculate the Laplacian distribution parameter for the transform domain noise model, the residual frame will be transformed by DCT. Then, the variance can be computed through the transformed residual. The variance is defined by the following equations:

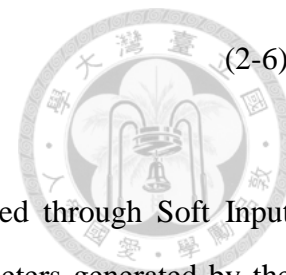
$$\sigma_b^2 = E_b[|T|_b^2] - (E_b[|T|_b])^2 \quad (2-3)$$

$$E_b[|T|_b] = \frac{1}{J} \sum_{j=1}^J |T|_b(j) \quad (2-4)$$

$$E_b[|T|_b^2] = \frac{1}{J} \sum_{j=1}^J [|T|_b(j)]^2 \quad (2-5)$$

where σ_b is the variance of a DCT-band b , $|T|_b$ is an absolute transformed residual $|T|$ of a DCT-band b , $E_b[|T|_b]$ is the expectation of an absolute transformed residual of band b , $E_b[|T|_b^2]$ is the expectation of the square of absolute transformed residual of band b , and J is the number of transform blocks in a frame. The Laplacian distribution parameter of the DCT-band b is defined by Equation 2-6:

$$\alpha_b = \sqrt{\frac{2}{\sigma_b^2}} \quad (2-6)$$



The probability inputted to LDPCA Decoder is calculated through Soft Input Computation by using the above Laplacian distribution parameters generated by the Virtual Channel Model.

Errors of bit planes in side information are corrected by LDPCA Decoder. And these bit planes are reconstructed by Reconstruction to form a transformed frame. Finally, the transformed frame will be processed by IDCT to become a pixel-domain frame, a decoded Wyner-Ziv frame.

2.3 Hybrid DVC

Hybrid distributed video coding with frame level coding selection was proposed in [8] and its framework is shown in Fig. 2-5.

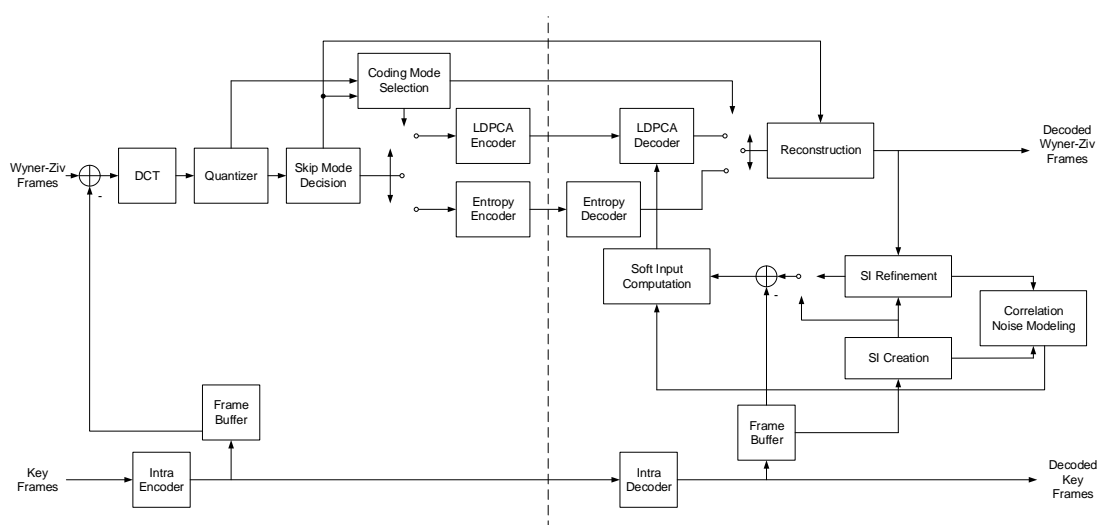


Fig. 2-5. The Hybrid DVC Framework [8].

Correlation Noise Modeling, Intra Encoder, and Intra Decoder are respectively the same as Virtual Channel Model, Conventional Intraframe Encoder, and Conventional Intraframe Decoder in DISCOVER as mentioned in Section 2.2. Besides the parts of DCT, Quantizer, LDPCA Encoder, LDPCA Decoder, Reconstruction, Soft Input

Computation, Correlation Noise Modeling, Intra Encoder, and Intra Decoder as mentioned in Section 2.2, there are some new blocks in Hybrid DVC including Skip Mode Decision, Coding Mode Selection, Entropy Encoder, Entropy Decoder, and Side Information (SI) Refinement. The main idea of the Hybrid DVC is that the authors applied conventional video techniques including residual coding, entropy coding, and skip mode to the framework. Therefore, they need a Coding Mode Selection in their frame level coding.

In the encoder, applying a residual codec to DVC can efficiently reduce the data needed to be transmitted from encoder to decoder by exploring the temporal redundancy between key frames and WZ frames at the cost of a little increase in computing complexity. The residual is calculated by the following equation:

$$r(x, y) = f_{WZ}(x, y) - f_{key}(x, y), \quad (2-7)$$

where r is the residual value, f_{WZ} is the current WZ frame, f_{key} is the reference key frame, and (x, y) is the coordinate of each frame. In order to coordinate residual coding, a new method of quantization step size was applied to the Hybrid DVC to fit the balance of RD performance. The new quantization step size is defined by Equation 2-8:

$$Q_i = \max\left(\frac{2^{|max(b_i)|}}{2^{M-1}}, Q_{i,min}\right), \quad (2-8)$$

where Q_i is the quantization step size of band i , b_i is the value of band i , $Q_{i,min}$ is the pre-defined minimum quantization step size of band i , and M is the number of bitplanes.

The residual frames processed by DCT and Quantizer will be input to Skip Mode Decision used in the encoder, which will help to reduce largely the transmitted data but add little error to the reconstructed frame. Unit of skip mode is a 4x4 block and the threshold of skip mode is defined by the following equation by calculating the transformed residual value:

$$Dist(n) = \sum_{u=3}^3 \sum_{v=0}^3 q_n^2(u, v), \quad (2-9)$$

where n is the index of block in the residual frame, and $q_n^2(u, v)$ is the square of transformed residual value in the block. If the value calculated by Equation (2-9) is lower than a defined threshold, the block will be skipped, which is controlled by Skip Mode Decision. And the skip flag will be coded by a run-length encoder to further reduce the transmitted data.

Then, Coding Mode Selection will decide the coding mode of non-skipped blocks and this is a new idea proposed in [8]. This decision is made for every frame meaning that all non-skipped blocks in a frame will be coded by the same coding mode. There are totally four modes including Channel Coding, Hybrid Mode 1, Hybrid Mode 2, and Entropy Coding. Channel Coding in the Hybrid DVC uses a low-density parity-check accumulate (LDPCA) coder and Entropy Coding uses a context-adaptive variable-length coder (CAVLC). Hybrid Modes combined with Channel Coding and Entropy Coding are used to code different frequency bands. In Channel Coding mode, all bands are coded by LDPCA. In Entropy Coding mode, all bands are coded by CAVLC. In Hybrid Mode 1, the lower three bands are coded by LDPCA and CAVLC is used to code the other bands. In Hybrid Mode 2, the lower six bands are coded by LDPCA and CAVLC is used to code the other bands. The rule of choosing a proper coding mode is shown in Table 2-1:

Table 2-1. Coding Modes and Decision Rules [8].

Coding mode	Decision rule
Channel Coding	$E_0 > \Gamma_0, E_1 > \Gamma_1, E_2 > \Gamma_2$
Hybrid Mode 1	$E_0 > \Gamma_0, E_1 \leq \Gamma_1$
Hybrid Mode 2	$E_0 > \Gamma_0, E_1 > \Gamma_1, E_2 \leq \Gamma_2$
Entropy Coding	$E_0 \leq \Gamma_0$



In Table 2-1, E_0, E_1, E_2 are the energy of three different groups and $\Gamma_0, \Gamma_1, \Gamma_2$ are defined thresholds of these three groups of energy. The energy E_0 is the summation of average amplitudes of the lower three zigzag-scan ordered bands, E_1 is the summation of average amplitudes of the next three zigzag-scan ordered bands, and E_2 is the summation of the other average amplitudes. The energy in each of the three groups is defined by the following equations:

$$E_0 = b_0 + b_1 + b_2 \quad (2-10)$$

$$E_1 = b_3 + b_4 + b_5 \quad (2-11)$$

$$E_2 = b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + b_{15} \quad (2-12)$$

The average amplitude b of each band i is denoted as:

$$b_i = \frac{1}{N} \sum_{n=1}^N q_n(i), \quad (2-13)$$

where N is the number of blocks, $q_n(i)$ is the transformed residual value of band i within block n .

In the decoder, WZ frames will be reconstructed through three paths which are skipped block, entropy decoding, and channel decoding. If the block is a skipped block, the values of the block are completely copied from the reference frame. Then,

the bands coded by Entropy Coding are reconstructed through the path of Entropy Decoder. The bands coded by Channel Coding will be reconstructed through the path of a Channel Decoder. The path of the Channel Decoder is the most complex for the side information must be generated by Side Information (SI) Creation and refined by Side Information Refinement [13]. The Side Information Creation includes Low Pass Filter, Forward Motion Estimation, Bidirectional Motion estimation, Spatial Motion Smoothing, and Bidirectional Motion Compensation as mentioned in Section 2.2. And the side information will be refined after the channel decoding of a bit plane. Some blocks of WZ frames flagged as skipped block are reconstructed through directly copying from reference frames. Other blocks of WZ frames are reconstructed by Entropy Decoding or Channel Decoding according the four coding modes. Then, decoded WZ frames are reconstructed by these decoded blocks.





CHAPTER 3

PROPOSED ARCHITECTURE AND IMPLEMENTATION

Lots of architectures of DVC have been proposed, and their goal is to improve the RD performance by adding extra components in the decoder, which also increases the complexity of the decoding process. High complexity of the decoding process causes that the decoded video cannot be played in real time. Hence, we propose an architecture of DVC based on Hybrid DVC [8] to improve these situations and implement this architecture in hardware. In the following sections, we will describe the proposed architectures of DVC, LDCPA Decoder, Side Information Creation including Motion Compensation and Spatial Motion Smoothing, CAVLC Decoder, DCT, IDCT, Quantization, De-Quantization, Correlation Noise Modeling, Soft Input Computation, and Reconstruction.

3.1 Proposed Architecture of DVC

The proposed DVC decoder is shown in Fig. 3-1. A new idea of our proposed architecture is that the motion vector is predicted in the DVC encoder in a small range such that more precise motion vectors can be input to the DVC decoder.

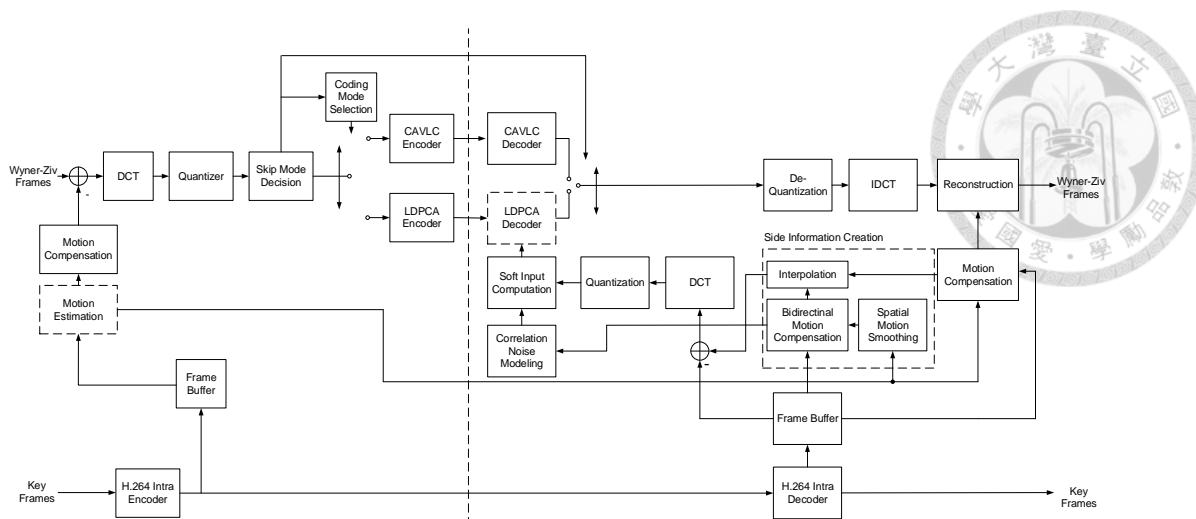


Fig. 3-1. Proposed Architecture of DVC.

Adding motion estimation in the DVC decoder will result in an increase of time complexity in the encoding process, but could improve RD performance. And compared to software simulation, it will only increase about ten percent of time complexity in the encoding process.

Most blocks of our proposed DVC decoder, including LDPCA Decoder, CAVLC Decoder, Soft Input Computation, Correlation Noise Modeling, Quantization, De-Quantization, DCT, IDCT, Bidirectional Motion Compensation, Spatial Motion Smoothing, Frame Buffer, and H.264 Intra Decoder, have been depicted in previous chapter, but there are some new blocks such as Motion Compensation and Interpolation to be presented. Since the residue frames in the proposed DVC encoder come from motion-compensated key frames and WZ frames, we need a Motion Compensation in the DVC decoder. And the reason of adding Interpolation in Side Information Creation is that the frames generated by Bidirectional Motion Compensation sometimes are less precise than the frames generated by Motion Compensation. Using Interpolation can help us to get better SI because of its averaging the errors of motion vectors from DVC encoder and Spatial Smoothing. Besides, the input data to the Reconstruction of the proposed DVC decoder are motion-compensated key frames and residue frames generated from motion-compensated key frames and WZ frames. And they could reduce the data

needed to be transmitted from encoder to decoder compared to those residue frames generated from key frames and WZ frames.

Most blocks of our proposed DVC encoder have been described in previous chapter, except the following two blocks, Motion Estimation and Motion Compensation. The reason why we added these two blocks is that the amplitudes of residue frames can be reduced and motion vectors predicted in the encoder can be used to create more precise side information in the decoder.

The proposed system architecture and data flow is shown in Fig. 3-2. The proposed system consists of external memory and blocks described in the previous paragraph. Dotted lines present that a frame-based process is within this region. Until a frame-based process is finished, the followed process will start. In each frame-based process, the external memory can only be accessed by a certain block. Hence, we schedule the system that only one certain block is running at a time. The encoded bits of H.264 are decoded by H.264 Intra Decoder to generate Key Frames. The Wyner-Ziv Frames are decoded by the Wyner-Ziv Decoder consisting of Spatial Motion Smoothing, Bidirectional Motion Compensation, Motion Compensation, Interpolation, I/DCT & De/Quant., Correlation Noise Modeling & Soft Input Computation, LDPCA Decoder, CAVLC Decoder, and Reconstruction. In decoding Wyner-Ziv Frames, input data is Key Frames, Motion Vectors, Syndrome Bits, and Encoded Bits of CAVLC, and some intermediate Transform-Domain Residue Frames.

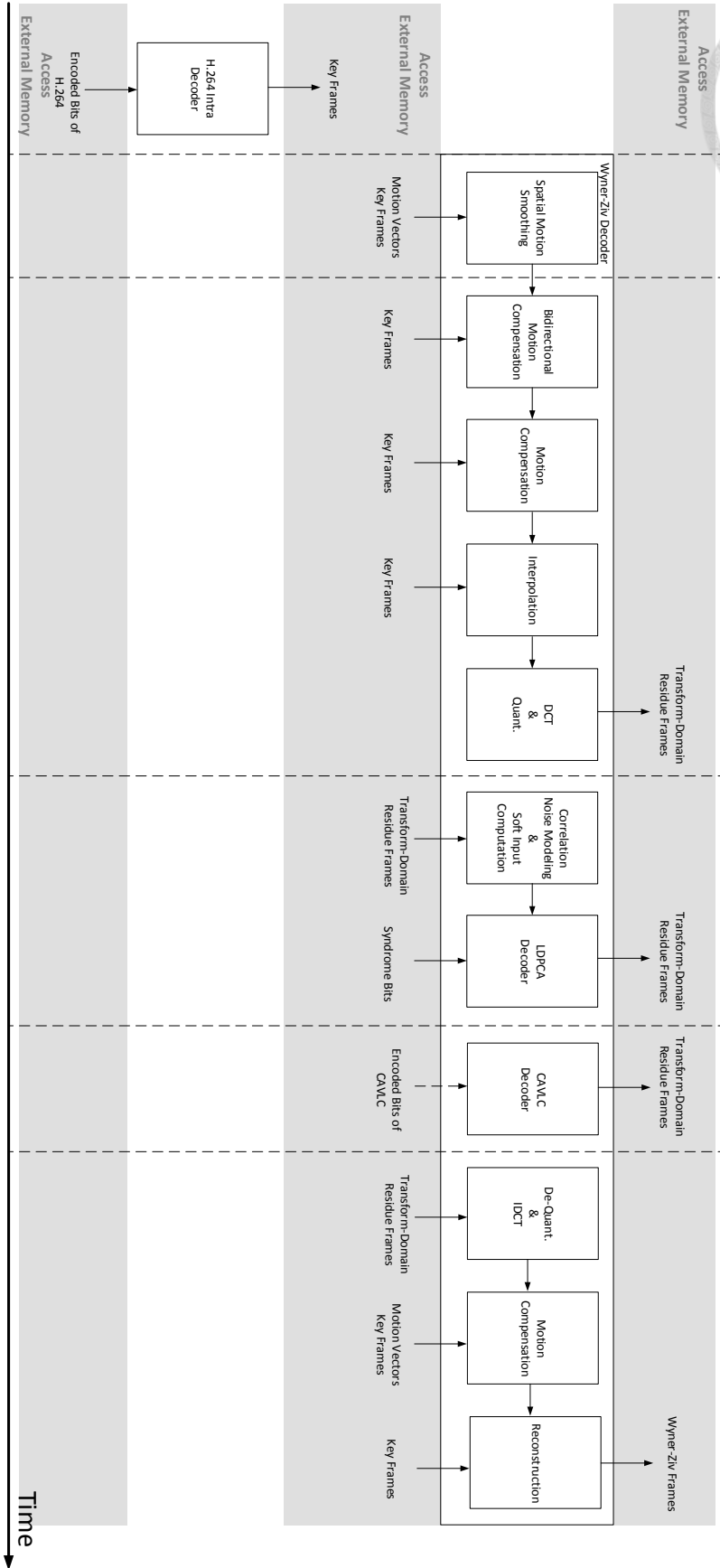
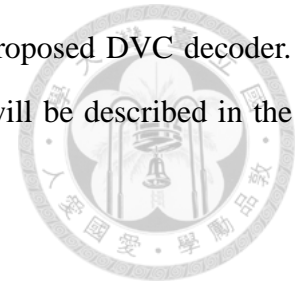


Fig. 3-2. Proposed System Architecture and Data Flow.

In this Thesis, we focus on the hardware design of our proposed DVC decoder. And the design of each block in the proposed DVC decoder will be described in the following sections.



3.2 LDPCA Decoder

Low Density Parity Check (LDPC) code, a linear block code, was invented by Gallager [14] in 1962. Then, Tanner's graph [15] transformed from the sparse parity-check matrix was proposed to clearly depict the concept of LDPC. Urbanke [16] proved that the error-correcting performance of LDPC is only 0.0045 dB far from Shannon limit. With its outstanding error-correcting performance, LDPC code has been applied to some digital communication systems as channel coding. Besides, LDPC code is used in fixed-rate distributed source coding [17]. We can use LDPC in a syndrome code form to transmit different amount of syndromes to achieve the goal of data compression. Though LDPC in syndrome code form can be used to compress data, there is a disadvantage that not every variable nodes are connected to check nodes in high compression rate as shown in Fig. 3-3, where V_{0-5} present the variable nodes and C_{0-5} present the check nodes.

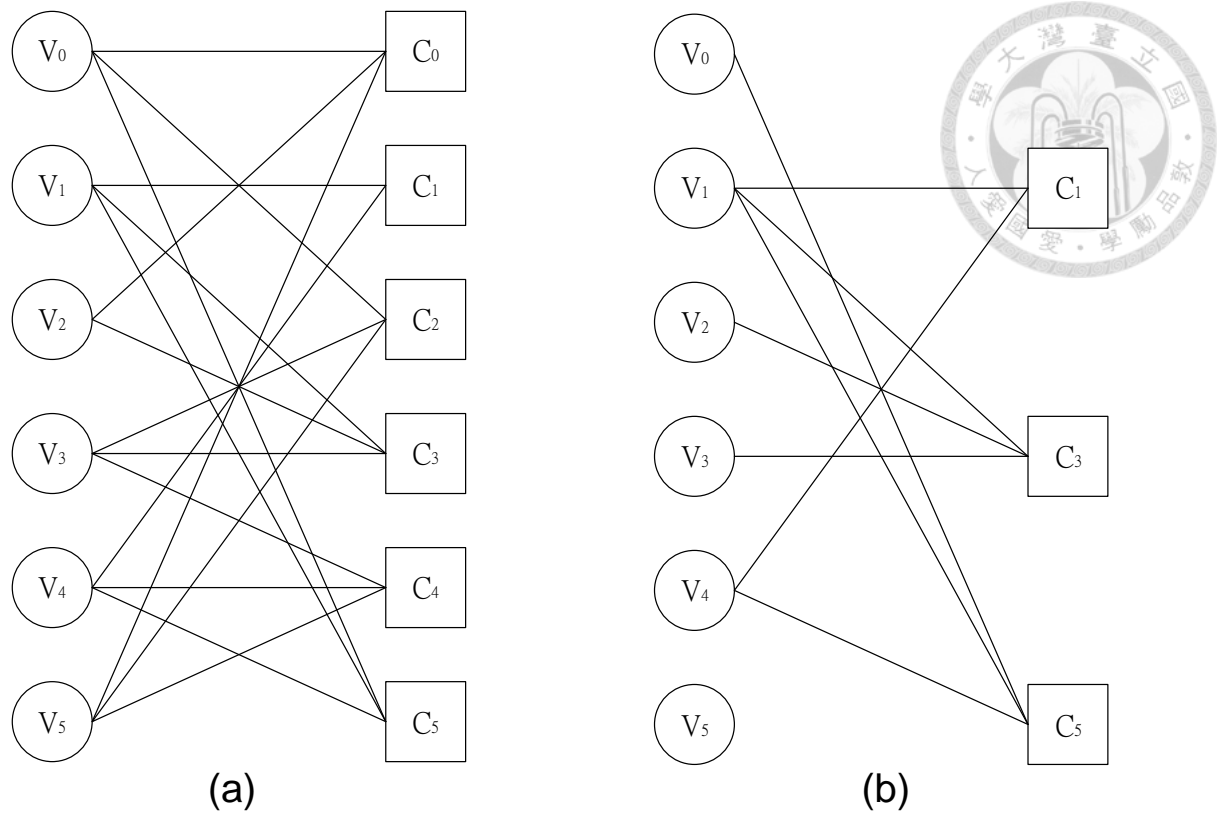


Fig. 3-3. Tanner Graphs.

Figure 3-3(a) shows that the encoder transmitted the entire syndrome bits and Fig. 3-3(b) shows that the encoder only transmitted odd-index syndrome bits. There are some variable nodes in Fig. 3-3(b) not connected to any check nodes or just connected to one check node. To solve this problem, low density parity check accumulate (LDPCA) was proposed [4]. The idea of LDPCA is that the syndrome generated by the encoder will be accumulated such that variable nodes will not be unconnected to check nodes. Fig. 3-4 shows the decoding graph with accumulated syndrome bits.

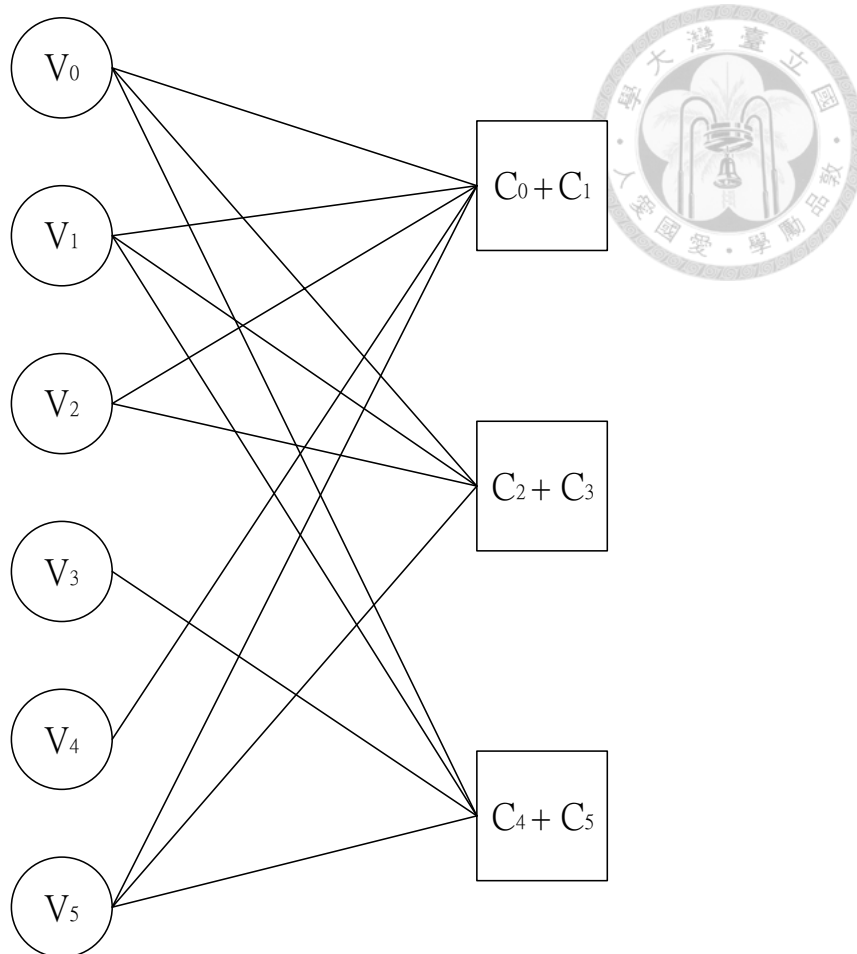


Fig. 3-4. Tanner Graphs with Accumulated Syndrome Bits.

Parity-check matrices of Fig. 3-3(a) and Fig. 3-4 are respectively shown in Fig. 3-5 and Fig. 3.6. We can see that two rows are merged when two check nodes are merged. Merging rows means that the elements from the same position in each of the two rows are binary added.

$$\mathbf{H} = \begin{array}{cccccc}
 & V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & \\
 \left[\begin{array}{cccccc}
 1 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 0 & 0 & 1 & 0 \\
 1 & 0 & 0 & 1 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 1 \\
 1 & 1 & 0 & 0 & 1 & 0
 \end{array} \right] & \begin{array}{l} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{array}
 \end{array}$$



Fig. 3-5. Parity-Check Matrix of the LDPCA shown in Fig. 3.3(a).

$$\mathbf{H} = \begin{array}{cccccc}
 & V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & \\
 \left[\begin{array}{cccccc}
 1 & 1 & 1 & 0 & 1 & 1 \\
 1 & 1 & 1 & 0 & 0 & 1 \\
 1 & 1 & 0 & 1 & 0 & 1
 \end{array} \right] & \begin{array}{l} C_0 + C_1 \\ C_2 + C_3 \\ C_4 + C_5 \end{array}
 \end{array}$$

Fig. 3-6. Parity-Check Matrix of the LDPCA shown in Fig. 3.4.

For example, Row 1 in Fig. 3-6 is obtained by merging the elements from Row 1 and Row 2 in Fig. 3-5. Binary addition of rows is shown below:

$$[1 \ 1 \ 1 \ 0 \ 1 \ 1] = [1 \ 0 \ 1 \ 0 \ 0 \ 1] + [0 \ 1 \ 0 \ 0 \ 1 \ 0]$$

There are some properties of LDPCA:

1. The Log-Likelihood Ratios (LLR) of message and syndrome bits are inputs of LDPCA, while the LLR of a code word (original message bits and redundancy bits) is the input of LDPC.
2. The total decoding iteration number of LDPCA is directly proportional to the number of different code rates.
3. The advantage is on its ability of compressing data and error correcting.
4. The disadvantage is on its high computation complexity.

After introduction of LDPCA, we are going to define the equations of check nodes merging. Only syndrome bits are generated by the LDPCA encoder, which form the

parity-check matrix used by the LDPCA encoder. Syndrome bits are defined in the following equation:

$$C_k = \sum_{i=0}^{n-1} H_{ki} \cdot V_i, \quad (3-1)$$

where C_k is the k^{th} syndrome bit, n is the code length, V_i is the i^{th} source bit, and H_{ki} is the (k,i) element of the parity-check matrix. The accumulated syndrome bits are defined in Equation 3-2:

$$A_k = \sum_{i=0}^k C_i, \quad (3-2)$$

where A_k is the k^{th} accumulated syndrome bit. The merged syndrome bit from check node j to check node k is defined in Equation 3-3:

$$C_{j\dots k} = \begin{cases} A_{j-1} \oplus A_k, & \text{if } j \neq 0 \\ A_k, & \text{else} \end{cases} \quad (3-3)$$

To simplify the transmitting process of different code rates, every 66 syndrome bits are grouped together. If data cannot be decoded completely in a current code rate, the encoder just transmits additional syndrome bits in a pre-determined order. The transmitted nodes in different code rates are listed in Table 3-1:

In this work, we implement the LDPCA using a Min-Sum Algorithm [18]. Since the check nodes of LDPCA could be merged in different compression rates and the syndrome bits are accumulated in every 66 check nodes, we proposed an architecture of Comparing Tree consisting of basic check nodes and stacking nodes to find the first and second minimum LLRs to be merged into Check Nodes (CNs) as shown in Fig. 3-7.

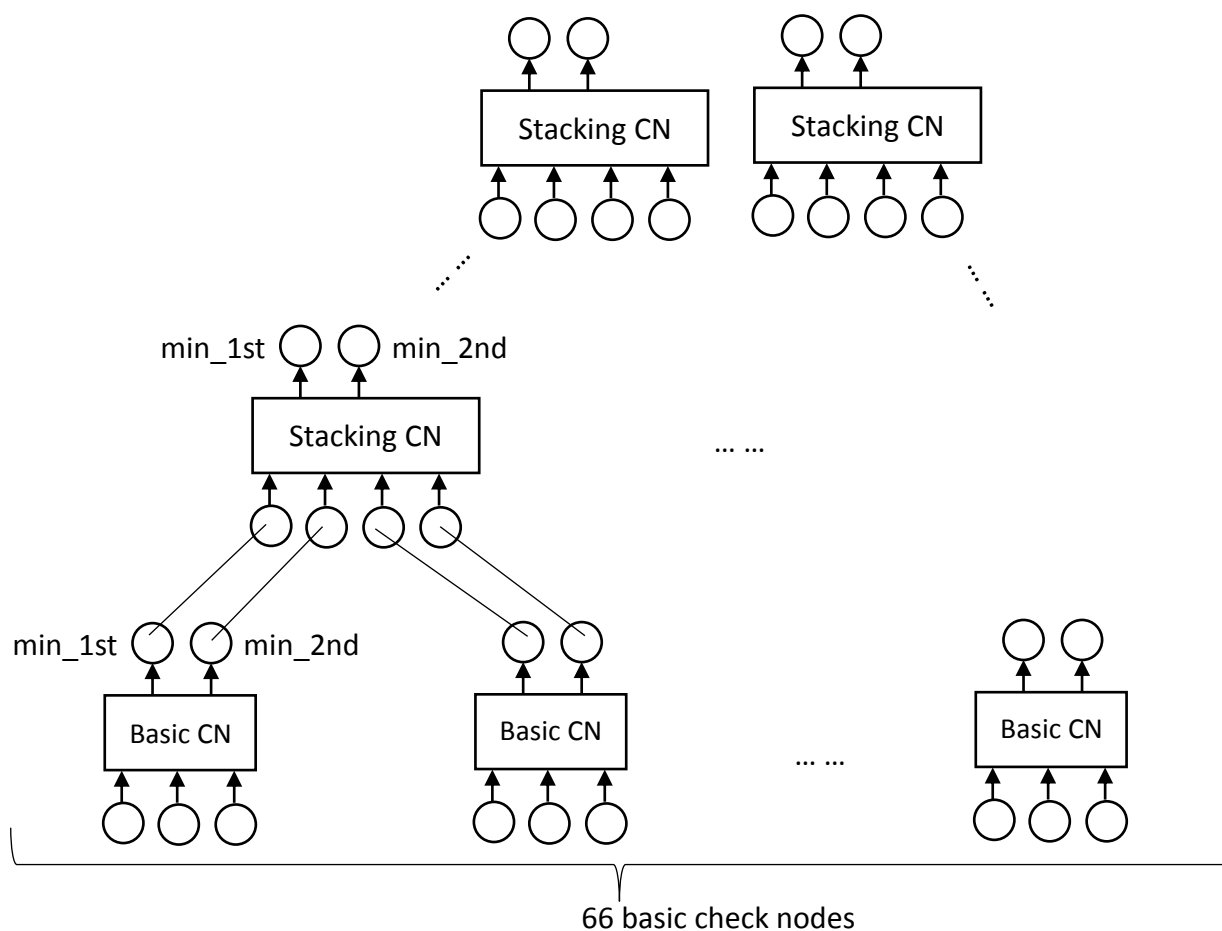


Fig. 3-7. Proposed Architecture of Comparing Tree.

Each wire in the Comparing Tree is used to transmit minimum values and parity of variable nodes to check nodes. According to the parity-check matrix proposed by Varodayan *et al.* [4], we design an algorithm to generate wires connecting Basic CNs and Stacking CNs. Besides, to implement the hardware, we change the number of

accumulated syndrome bits to 66 only, without accumulating all the syndrome bits.

The flow of creating a Comparing Tree based on 66 basic check nodes is as follows:

- (1) Allocate an array for Basic CNs and an array for pointers which are initially pointing to Basic CNs.
- (2) From the code rate of 66/66 to the code rate of 2/66, every time the code rate is changed, a Stacking CN will be inserted to the Comparing Tree. The pointers of the to-be-merged two check nodes will point to the new inserted Stacking CN. The pointer of the pointer array in the position of a transmitting node will point to the new inserted Stacking CN, while the pointer in the position of the to-be-merged node of the pointer array will point to null.
- (3) After iterations of code rates from 66/66 to 2/66 in Step 2 are done, we can traverse from the array of Basic CNs by breadth-first search to know how many Stacking CNs are inserted and wires are connected to Basic CNs and Stacking CNs.

The schematic diagram of wires connected to Variable Nodes (VNs), Basic CNs, and Stacking CNs is shown in Fig. 3-8.



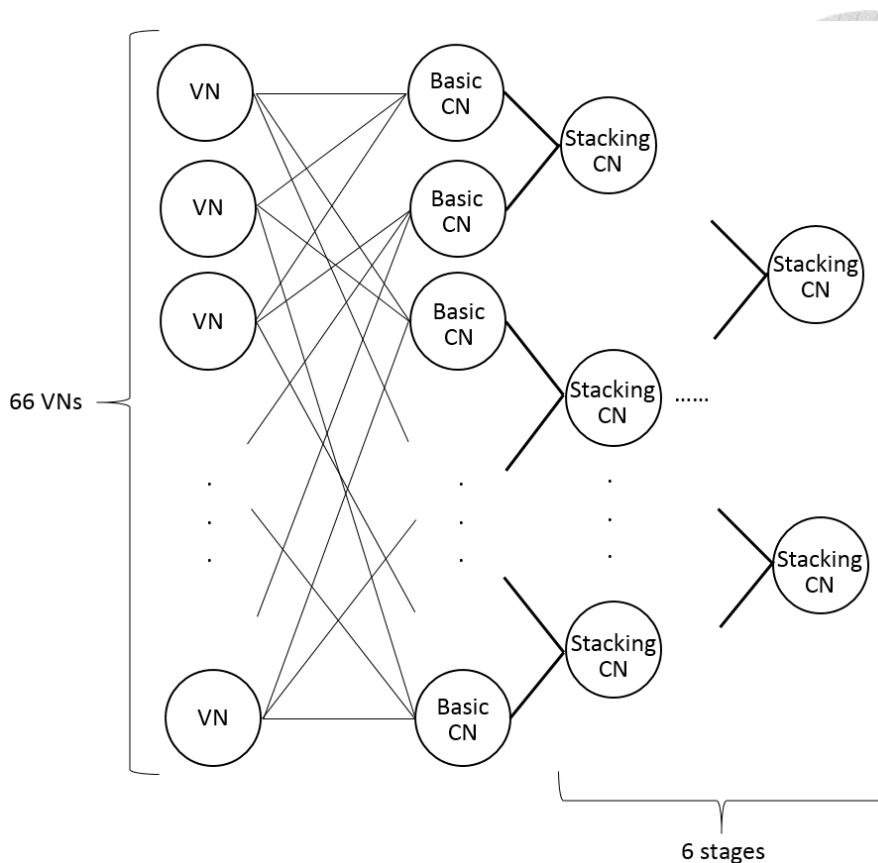


Fig. 3-8. Schematic Diagram of Wires Connected to VNs and BNs.

The units of the Basic CNs and Stacking CNs in the Comparing Tree are different. There are three input values and two outputs for the first and second minimum values in the Basic CNs. There are two pairs of inputs of sorted values and two outputs for the first and second minimum values in the Stacking CNs. Their function is to find the first two minimum values with minimum-value generators [19] as shown in Fig. 3-9 and Fig. 3-10.

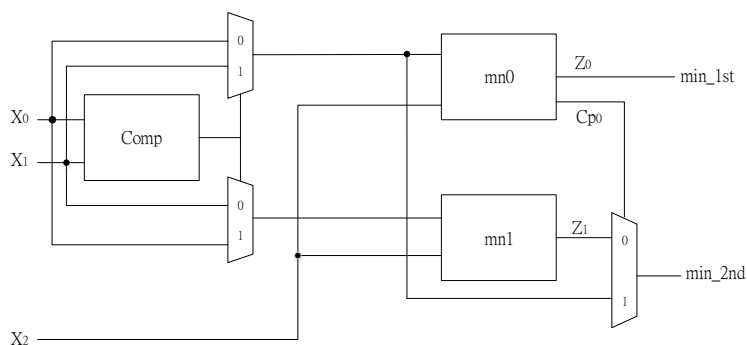


Fig. 3-9. A 3-Input Minimum-Value Generator.

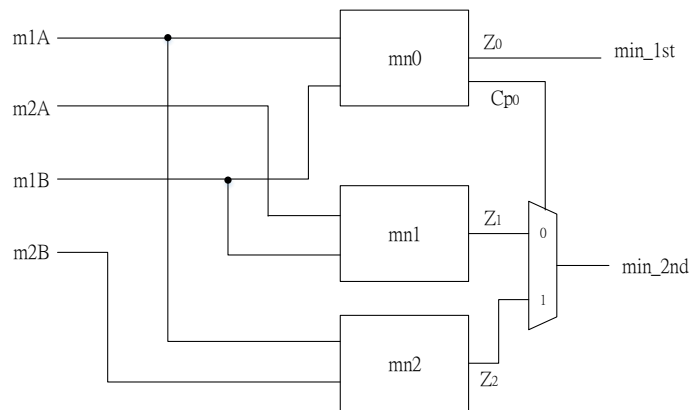


Fig. 3-10. A 4-Input Minimum-Value Generator.

In Fig. 3-9, X_0 , X_1 , and X_2 are inputs; Comp is a two-input comparator; mn0 and mn1 are two 2-input minimum-value generators outputting a minimum value and an index of a minimum value; Z_0 and Z_1 are minimum values; C_{p0} is the index of a minimum value; min_1st and min_2nd are the first minimum value and the second minimum value respectively. In Fig. 3-10, m1A, m2A, m1B, and m2B are two pairs of sorted minimum values; mn0, mn1, and mn2 are three 2-input minimum-value generators; Z_0 , Z_1 , and Z_2 are minimum values; min_1st and min_2nd are the first minimum value and the second minimum value respectively.

Figure 3-11 is an example of Comparing Tree consisting of Basic CNs and Stacking CNs. C_0 , C_1 , C_2 , C_3 , C_4 , and C_5 are Basic CNs. SC_0 , SC_1 , and SC_2 are Stacking CNs. The numbers on the left side of Basic CNs are the order of transmitted accumulated syndrome bits. Code rates, 6/6, 5/6, 4/6, and 3/6 present different steps of inserting Stacking CNs as described in the previous flow of creating Comparing Tree(p.32). According to proposed algorithm of creating Comparing Tree, Basic CNs are inserted first. From the code rate of 5/6 to the code rate of 3/6, every time the code rate is changed, a Stacking CN is inserted to the Comparing Tree. And the wires are connected to the merged Basic CN and the inserted Stacking CN.

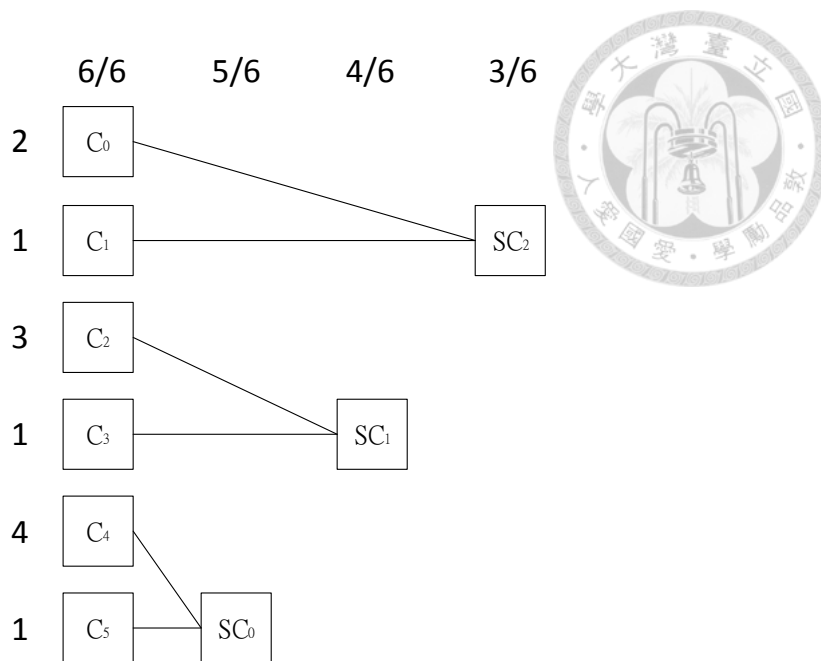


Fig. 3-11. An Example of Comparing Tree Consisting of Check Nodes.

Two examples of merging check nodes are shown in Fig. 3-12 and Fig. 3-13 where $V_0, V_1, V_2, V_3, V_4,$ and V_5 are variable nodes, $C_0, C_1, C_2, C_3, C_4,$ and C_5 are Basic CNs, and $SC_0, SC_1,$ and SC_2 are Stacking CNs. In Fig. 3-12, code rate is $4/6$. Basic CNs, C_2 and C_3 , are merged first; then Basic CNs, C_4 and C_5 , are merged. Variable nodes connected to Merged Check Nodes are updated by message passing from SC_0 and SC_1 while other variable nodes are updated by message passing from C_0 and C_1 . In Fig. 3-13, code rate is $5/6$ and there is only one Merged Check Node since there is only one accumulated syndrome bit not received by LDPCA decoder. Variable nodes connected to Merged Check Node are updated by message passing from SC_0 while other variable nodes are updated by message passing from $C_0, C_1, C_2,$ and C_3 . Other code rates can be achieved by the same way.

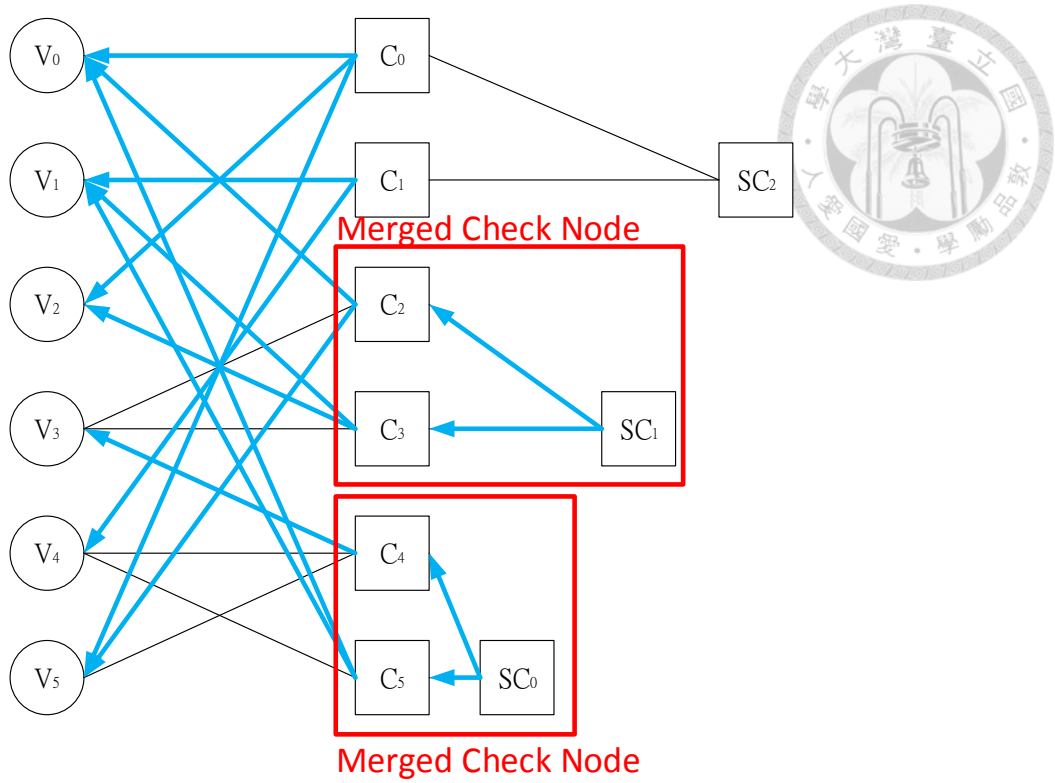


Fig. 3-12. An Example of Merging Check Nodes in Code Rate of 4/6.

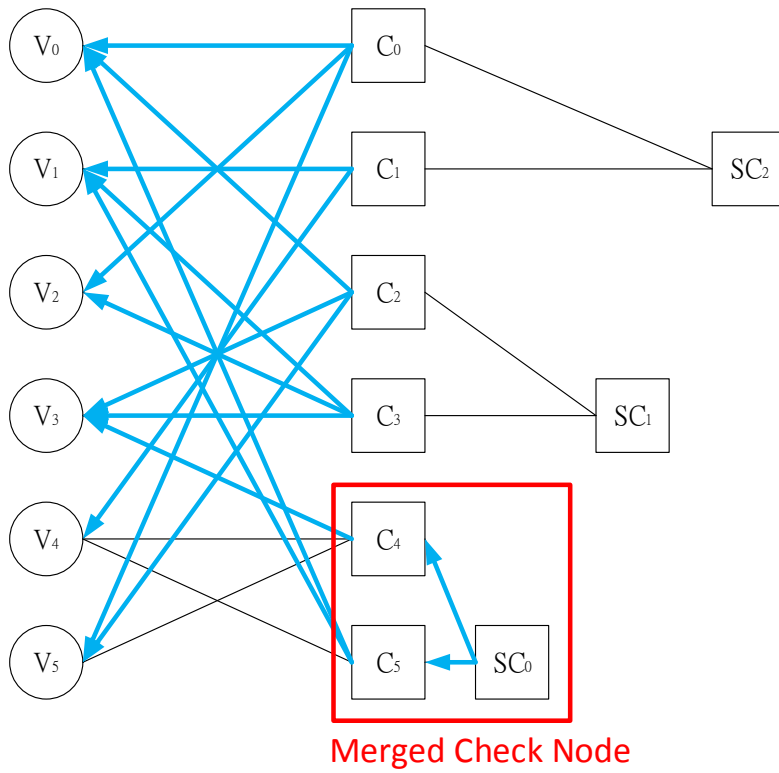


Fig. 3-13. An Example of Merging Check Nodes in Code Rate of 5/6.

In Fig. 3-8, there are 6 stages of Stacking CNs because 66 Basic CNs are included. Since the order of transmitting accumulated syndrome bits in each grouping of the 66 VNs is the same, the Comparing Tree of each group of 66 CNs is also the same. Thus, we proposed an architecture of LDPCA decoder as shown in Fig. 3-14.

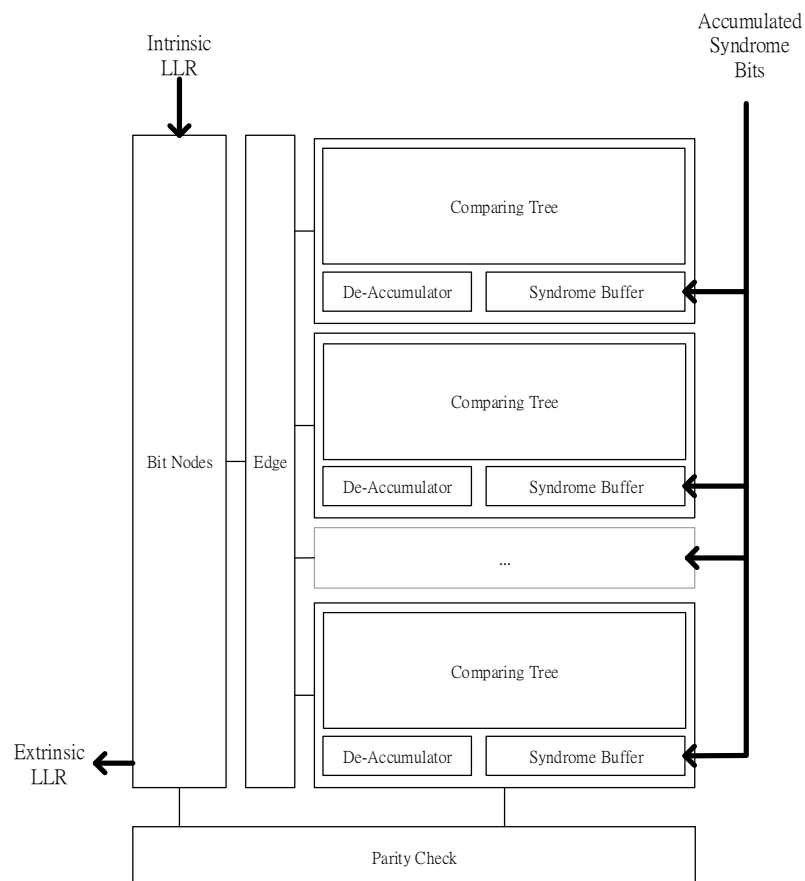


Fig. 3-14. Proposed Architecture of LDPCA Decoder.

Intrinsic LLR and Accumulated Syndrome Bits are input data, and extrinsic LLR will be outputted when the decoding process is finished. Each Accumulated Syndrome Bits stored in the Syndrome Buffer are required by the LDPCA decoder in different code rates according to Table 3-1. Basic CNs merged in the same CN will fetch the same syndrome bits. The syndrome bits fetched by basic check nodes are generated by a De-Accumulator. The De-Accumulator is a combinational circuit implementing

the function of Equation (3-3) and its input is the accumulated syndrome bits stored in the Syndrome Buffer. Exclusive-Or is used for binary addition in Equation (3-3). Edges of Bit Nodes in the Comparing Tree are generated according to the parity-check matrix proposed in [4]. And Parity Check is also designed according to the same parity-check matrix proposed in [4].

The state diagram of the proposed LDPCA decoder is shown in Fig. 3-15. First, LDPCA decoder reads the channel value, reads syndromes, and initializes message. Check nodes and variable nodes are updated. Then, we decide whether the syndromes are matched and the iteration number is equal to the maximum iteration number. If the checked syndrome is not matched and the iteration number is not equal to the minimum iteration number, the LDPCA decoder continues reading syndromes and performing decoding process. If the checking syndrome is matched, or if the source rate is 66/66 and the maximum iteration number is matched, the decoding process is finished. Finally, the decoded values are outputted.

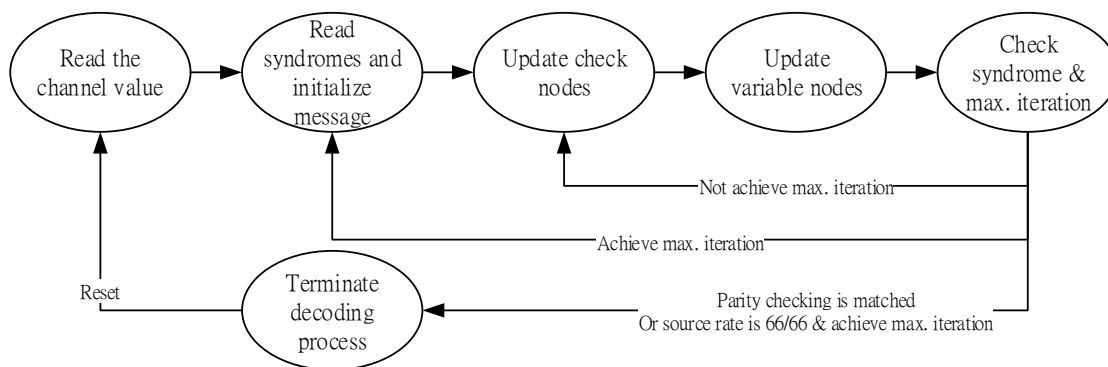


Fig. 3-15. State Diagram of Proposed LDPCA Decoder.

3.3 Side Information Creation

Side information creation is used to generate approximate Wyner-Ziv frames called side information. The proposed framework of side information creation is shown in Fig. 3-16. The proposed side information creation consists of three parts including Spatial Motion Smoothing, BiDirectional Motion Compensation, and Interpolation. First, Spatial Motion Smoothing adjusts the motion vectors on a frame

according to the input motion vectors of Previous Key Frame and Next Key Frame. Next, a motion-compensated key frame is generated by BiDirectional Motion Compensation using adjusted motion vectors. Then, the motion-compensated key frame and a Motion-Compensated Previous Key Frame are combined by Interpolation to create Side Information. Details of Spatial Motion Smoothing, BiDirectional Motion Compensation, and Interpolation will be described in the following subsections.

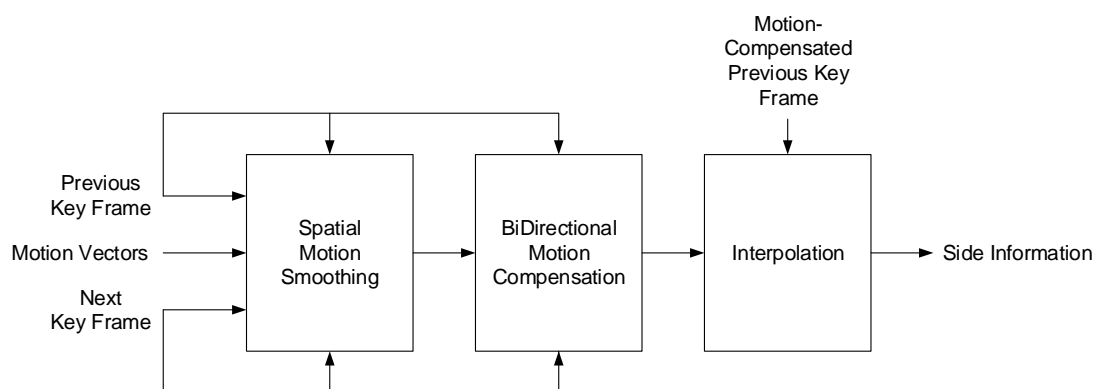


Fig. 3-16. Proposed Framework of Side Information Creation.

3.3.1 Spatial Motion Smoothing

The function of Spatial Motion Smoothing is to find the appropriate motion vectors within a specified window since there might be some errors in motion vectors predicted by block matched motion estimation. For example, Fig. 2-4(a) presents the motion vectors predicted by motion estimation and Fig. 2-4(b) presents the motion vectors after Spatial Motion Smoothing that refines the motion vectors according to the motion vectors of neighboring blocks within a specified window. In this example, the motion vector in the center of Fig. 2-4(a) is not correct because it just was predicted by the sum of absolute differences (SAD) which might cause block effect. After Spatial Motion Smoothing, the motion vector in the center was corrected such that block effect can be avoided.

In our work, we use a weighted vector median filter [20] to implement the

function of Spatial Motion Smoothing in a block. The idea of this filter is that not only distance of motion vectors within a specified window but also a weighted value are considered as predicted values. And the weighted value is the SAD of the previous forward motion-compensated block and next backward motion-compensated block. The criterion of the weighted vector median filter is defined in the following equation:

$$\sum_{j=1}^N SAD(x_{wvmf}, B)(|X_{wvmf} - X_j| + |Y_{wvmf} - Y_j|) \leq \sum_{j=1}^N SAD(x_c, B)(|X_c - X_j| + |Y_c - Y_j|), \quad (3-4)$$

The above equation is similar with Equation (2-1) except the SAD. x_{wvmf} is the motion vector obtained at the output of the weighted vector-median filters, N is the number of motion vectors within a specified region, x_c is the current appropriate motion vector, X and Y are x-dimension and y-dimension values of motion vectors respectively, and $SAD(x, B)$ means the sum of absolute differences of the motion vector x of current block compensated by previous and next reference frames.

As shown in Fig. 3-17, the proposed architecture of weighted vector median filter is modified from [21]. The architecture consists of Control Unit, Datapath of Distance, Weighing & Minimum Selector, and Datapath of SAD. Control Unit is used to control the dataflow of motion vectors, pixels of frames, and output vector. Datapath of Distance is used to calculate the distance of motion vectors. Datapath of SAD is used to calculate the SAD of compensated blocks. Weighing & Minimum Selector is used to calculate the weighting factor and select the candidate motion vector.

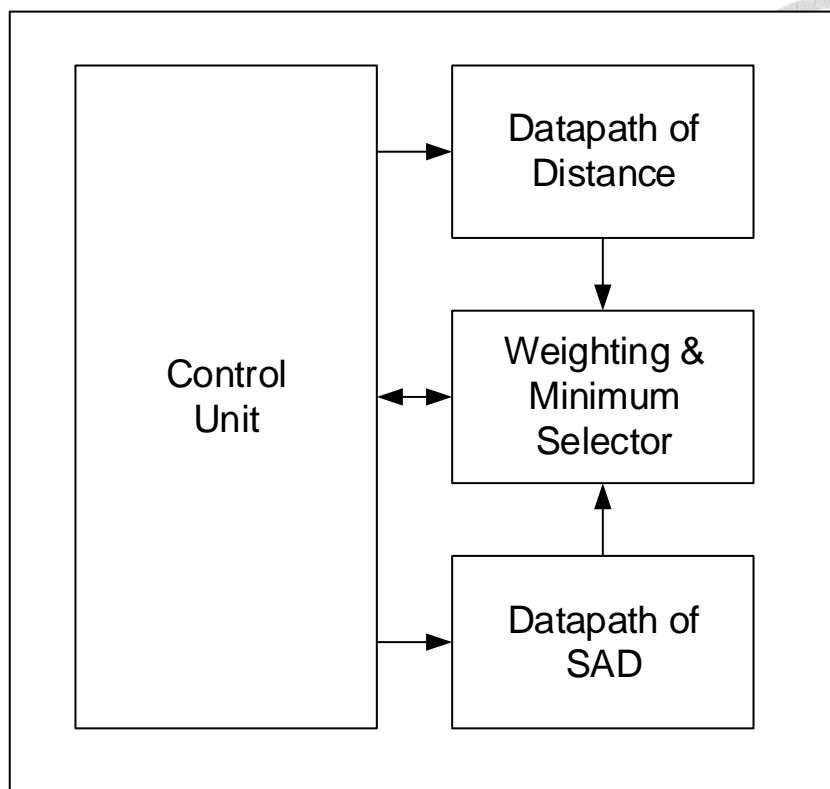


Fig. 3-17. Proposed Architecture of Weighted Vector Median Filter.

The architecture of Datapath of Distance is shown in Fig. 3-18. In this architecture, the window size is nine blocks. The motion vector is inputted through V_{in} to each register, V_1 Reg. to V_9 Reg. The mux is used to select the proper motion vectors, V_1 to V_9 , in different calculation phases. When the specified window moves, the number of input motion vectors is different. We have to input nine motion vectors into the nine registers, V_1 Reg. to V_9 Reg., when the specified window is at the leftmost of the row. We only input three motion vectors into the three registers, V_3 Reg., V_6 Reg., and V_9 Reg., when the specified window is not at the leftmost of the row. After calculation through absolute, adder, and storing in the register, D_i contains the distance of motion vectors.

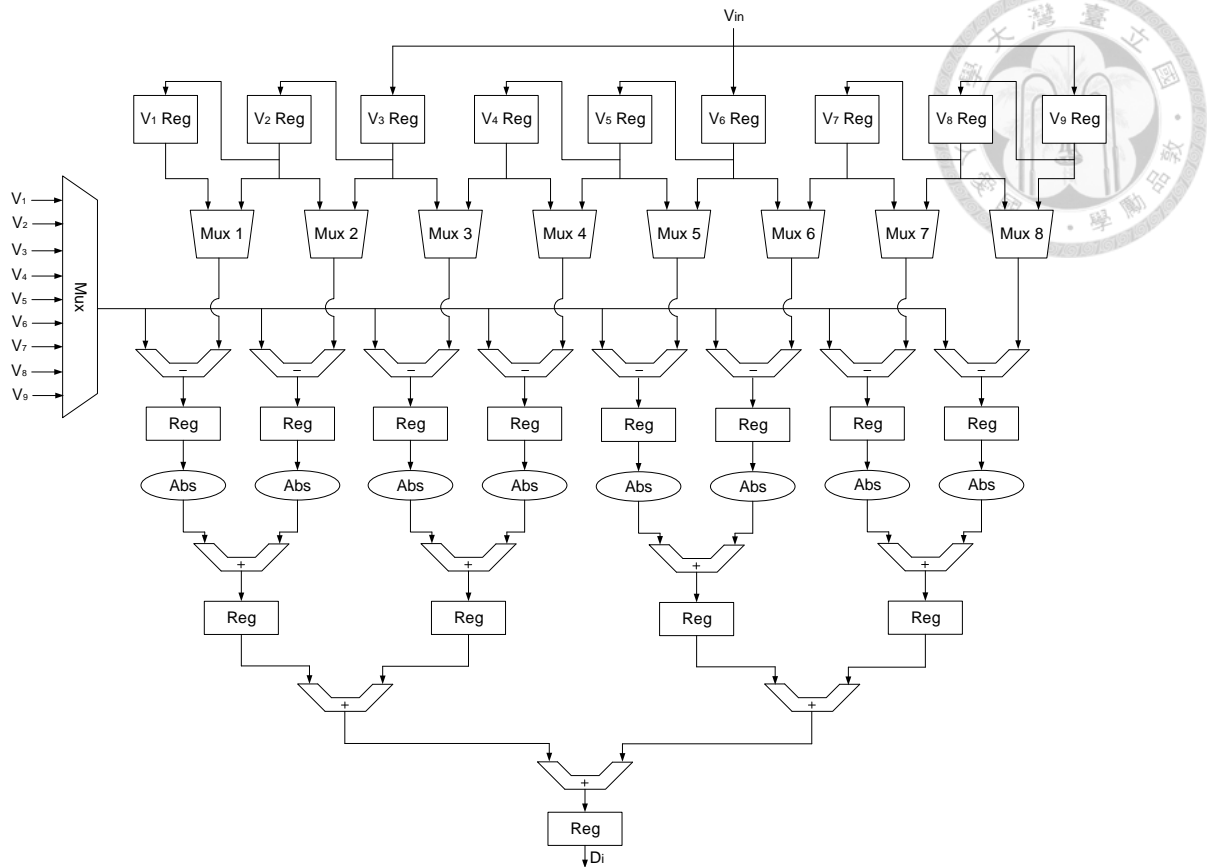


Fig. 3-18. Architecture of Datapath of Distance.

The architecture of Datapath of SAD is shown in Fig. 3-19. Pixels of a current block and compensated by the input previous and next reference frames, P_{prev} and P_{next} , controlled by the Control Unit. First, these two input values are calculated through a subtractor. Next, the absolute value of this difference value is obtained by Abs. Then, the absolute value is added with a temporary weighting factor, W_i , and the sum is stored in a register. Finally, calculation of weighting factor is completed after reading every pixels of the block.

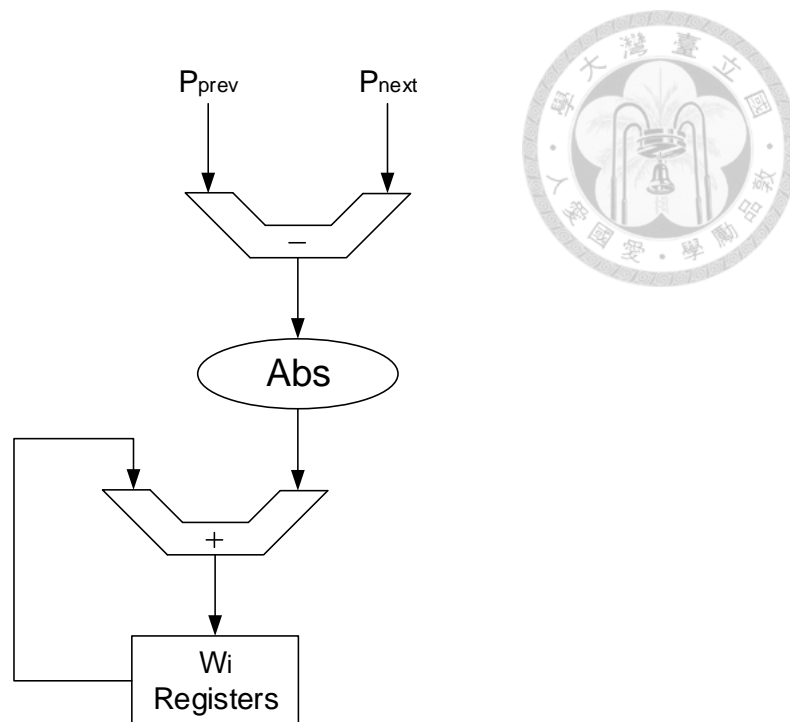


Fig. 3-19. Architecture of Datapath of SAD.

The architecture of Weighting & Minimum Selector is shown in Fig. 3-20. The functions of Weighting & Minimum Selector are multiplying the weighting factor and the distance, and selecting the motion vector with a minimum weighting value. The weighting values of components x and y are separately computed. The weighting value of x , WD_{ix} , is equal to weighting factor, W_{ix} , multiplied by the distance of component x , D_{ix} . The weighting value of y is calculated like the weighting value of x . The sum of weighting values will be compared with the value stored in the register to find the minimum weighting value by the Comparator. If the weighting value is smaller than the value stored in the register, the value stored in the register will be substituted by the weighting value and the previous motion vector will be substituted by the current motion vector.

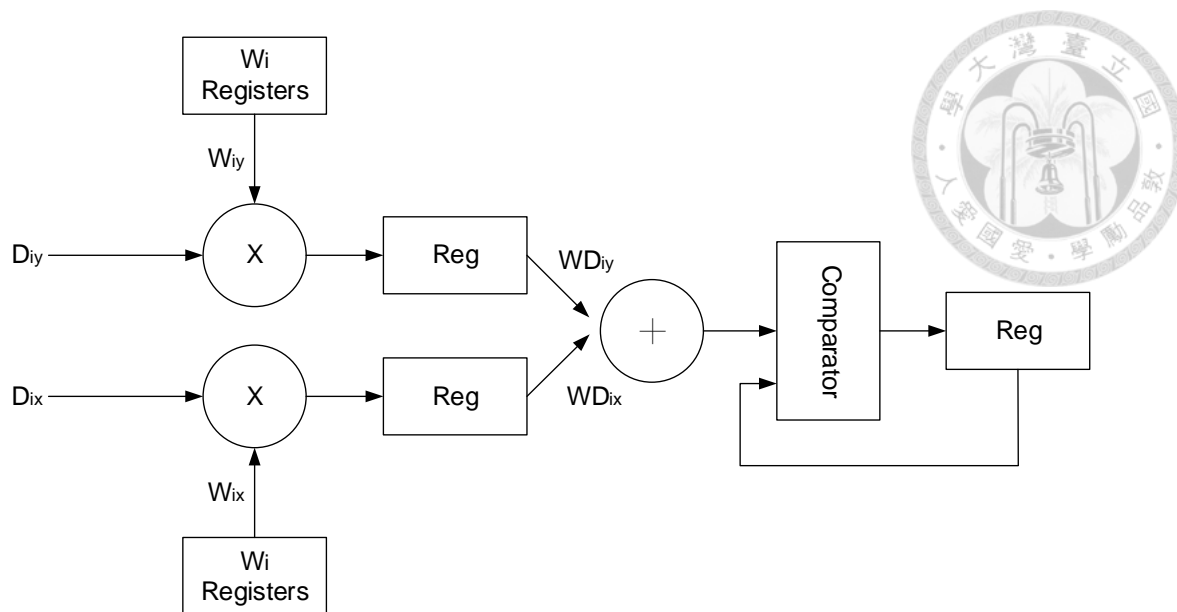


Fig. 3-20. Architecture of Weighing & Minimum Selector.

3.3.2 BiDirectional Motion Compensation

BiDirectional Motion Compensation means that using motion vectors filtered by a vector median filter to get forward motion-compensated frame and backward motion-compensated frame. The residue frames of forward motion-compensated frames and backward motion-compensated frames are used to be the inputs of Correlation Noise Modeling. The interpolation frames of forward motion-compensated frames and backward motion-compensated frames form exactly the side information.

The inputs of BiDirectional Motion compensation are motion vectors and key frames. The motion vectors of a forward motion-compensated frame and a backward motion-compensated frame are only different in sign bit. The direction of motion vectors of forward motion compensation and backward motion compensation is opposite as shown in Fig. 3-21.

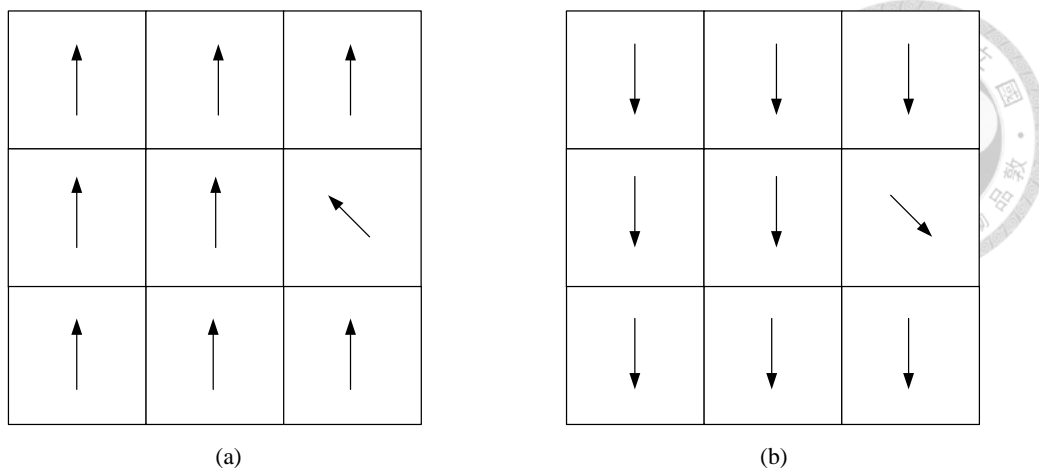


Fig. 3-21. Motion Vectors of (a) Forward and (b) Backward Compensations.

Key frames are read from external memory units, previous key frame buffer and next key frame buffer. The motion vectors stored by the vector median filter are read from SRAM. The motion vectors for backward motion compensation are obtained by letting the input motion vectors multiplied by negative 1, while the motion vectors for forward motion compensation are the input motion vectors.

The hardware of motion compensation is mainly for the control of reading address and writing address, and signal of writing. Besides, we have to consider the situation where motion vectors are pointing to the outside of the frame boundary. In this situation, we will use the nearest pixel of this frame instead.

The architecture of motion compensation is shown in Fig. 3-22. Motion Vectors are stored into registers first. Pixel Address of a reference frame is calculated by Control according to these stored motion vectors and current block. Pixel will stored into Buffer, and Control will send signal of Enable to present the Compensated Pixel is ready. The process between calculating Pixel Address and presenting the Compensated Pixel is ready will repeat until each Compensated Pixel of a frame is completed.

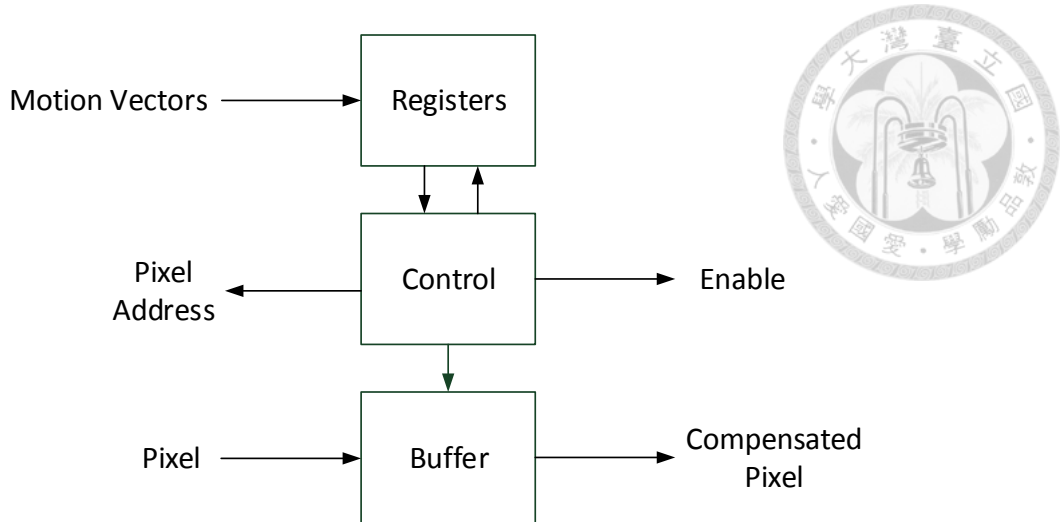


Fig. 3-22. Architecture of Motion Compensation.

3.3.3 Interpolation

The function of Interpolation in Side Information Creation is to calculate the mean values of pixels between bidirectional motion-compensated frames. The interpolated pixel can be presented as the following equation:

$$p_i = (p_f + p_b + 1) \gg 1, \quad (3-5)$$

where p_i is the interpolated pixel, p_f is the pixel in a forward motion-compensated frame, and p_b is the pixel in a backward motion-compensated frame. Adding one and right shifting one in Equation (3-5) means the content is divided by 2 and rounded to a single digit.

Implementation of Interpolation mainly consists of memory accessing and a combinational circuit of adder and shifter. Pixels are read from external memory and stored to external memory. The stored pixels will be read by the next stage. The combinational circuit is to implement the function of Equation (3-5).

The architecture of Interpolation is shown in Fig. 3-23. P_{forward} is the pixel in a forward motion-compensated frame, and P_{backward} is the pixel in a backward motion-compensated frame. We obtain the interpolation pixel, P_{forward} , by right shifting the sum of P_{forward} , P_{backward} , and 1 one bit.

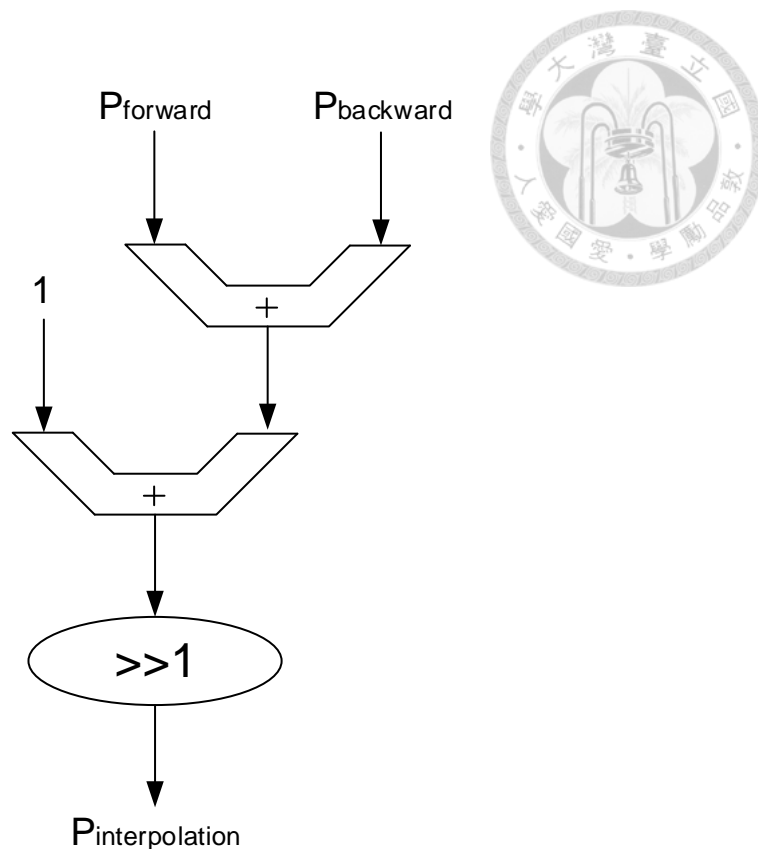


Fig. 3-23. Architecture of Interpolation.

3.4 CAVLC Decoder

Context-based Adaptive Variable Length Code (CAVLC) is proposed to code the coefficients after transform and quantization. CAVLC provides a good compression with some lossy compression techniques like quantization. The hardware of CAVLC used in our work is proposed in [22]. The decoding flow is shown in Fig. 3.24.

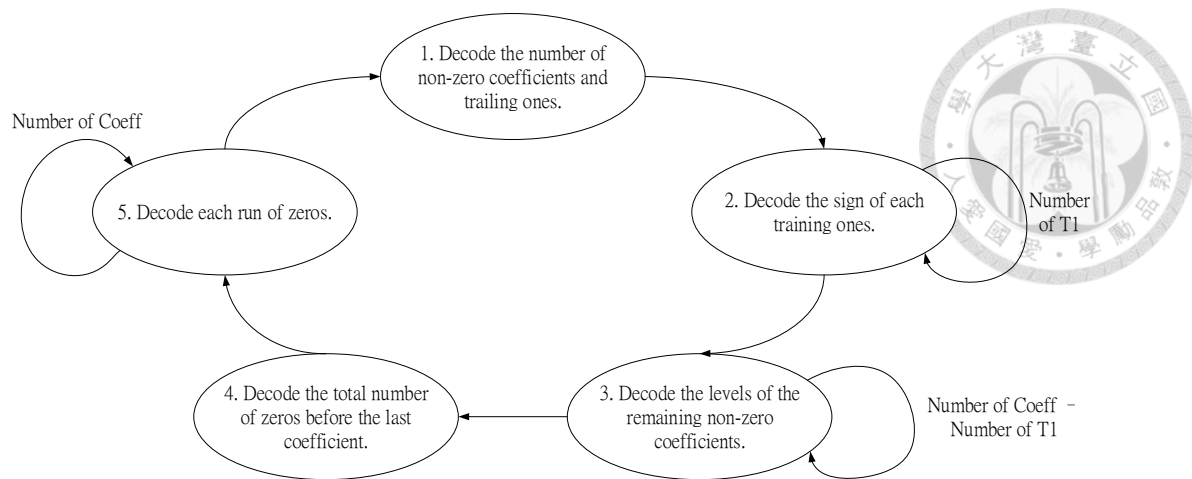


Fig. 3-24. Decoding Flow of CAVLC Decoder [22].

The flow of decoding process is as follows:

- (1) According to the number of non-zero coefficients in the left block and top block, and look-up table, we can decode the number of non-zero coefficients (Coeff) and number of training ones (T1). The range of the number of Coeff is from 0 to 16, and the range of the number T1 is from 0 to 3. There are five look-up tables in decoding the number of Coeff and T1, one of them is for chroma.
- (2) After knowing the number of T1, we can immediately decode T1 according to the bit stream. Only one bit is used to decode the sign of T1. Bit 1 means negative 1 and Bits 0 means positive 1.
- (3) In this step, the level of Coeff will be decoded according to 7 look-up tables, Level-VLC0 to Level-VLC6. The decision is dependent on the number of Coeff, the number of T1, and the threshold of decoded values. Initially, if the number of Coeff is greater than 10 and the number of T1 is less than 3, Level-VLC1, or Level-VLC0 will be chosen. The coefficient with the highest frequency will be decoded first. If the magnitude of the previous decoded coefficient is greater than the threshold of the current table, the next table will be chosen.
- (4) According to the number of Coeff, decode the total number of zero before the last coefficient.

- (5) Run-before zeroes is decoded according to the total number of zeroes, the previous number of zeroes of the coefficient, and the run-before table.

The block diagram of CAVLC Decoder is shown in Fig. 3-25. Bitstream will be fetched into R0 and R1. According to the signals of Controller and Accumulator, the correct 16-bit data will be outputted to Coeff-Token Decoder, TotalZero Decoder, Run_Before Decoder, and T1 Decoder through Barrel Shifter. The functions of these five decoders are depicted in previous decoding process. After finishing the decoding process, the 16 decoded level values will be stored in Registers and outputted to next stage.

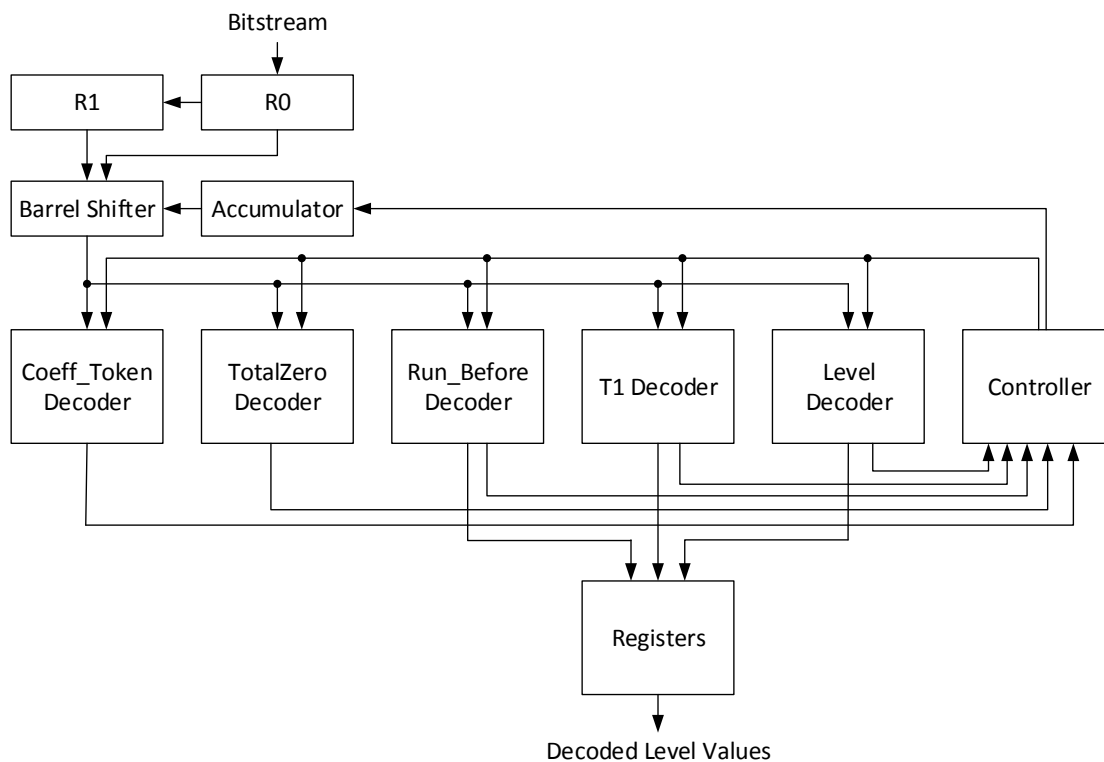


Fig. 3-25. Block Diagram of CAVLC Decoder.

3.5 DCT, IDCT, Quantization, and De-Quantization

Forward integer discrete cosine transform (DCT) and Quantization are the lossy compression techniques applied in distributed video coding. In the encoder, these two techniques are used to reduce the data rate. In the decoder, these two techniques are

used to create side information. Inverse integer discrete cosine transform (IDCT) and De-Quantization are used to transform a quantized and transformed domain residue to a pixel domain residue.

In our work, we implement a direct 2-D transform coding [23] hardware as a core for forward and inverse integer discrete cosine transforms. The 4x4 integer transform is shown as the following equation:

$$Y = (CXC^T) \otimes E$$

$$= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} [X] \begin{pmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{pmatrix} \otimes E, \quad (3-6)$$

where X is the input data, Y is the output data, E is the matrix of scaling factors, and CXC^T is the core of 2-D transform. After row-column decomposition and equation substitution, we can obtain the following equations:

$$M_{tu} = \sum_{i=0}^3 C_{ti} \cdot X_{iu}, \quad t, u = 0, \dots, 3 \quad (3-7)$$

$$Y_{st} = \sum_{u=0}^3 C_{su} \cdot (M_{ut}^T), \quad s, t = 0, \dots, 3 \quad (3-8)$$

where M is the intermediate data between the transform process. After substituting Equation 3-7 into Equation 3-8, the equations of forward transform can be expressed as following equations:

$$\begin{bmatrix} Y_{s0} \\ Y_{s2} \end{bmatrix} = \sum_{u=0}^3 C_{su} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{bmatrix} X_{0u} + X_{3u} \\ X_{1u} + X_{2u} \end{bmatrix}, \quad s = 0, \dots, 3 \quad (3-9)$$

$$\begin{bmatrix} Y_{s1} \\ Y_{s3} \end{bmatrix} = \sum_{u=0}^3 C_{su} \cdot \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix} \cdot \begin{bmatrix} X_{0u} - X_{3u} \\ X_{1u} - X_{2u} \end{bmatrix}, \quad s = 0, \dots, 3 \quad (3-10)$$

In the same way, we can obtain the equations of inverse transform shown in Equation 3-11 and Equation 3-12.

$$\begin{bmatrix} X_{s0} \\ X_{s3} \end{bmatrix} = \sum_{u=0}^3 C_{su}^{inv} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{bmatrix} Y_{0u} + Y_{2u} \\ Y_{1u} + \frac{1}{2}Y_{3u} \end{bmatrix}, \quad s = 0, \dots, 3 \quad (3-11)$$

$$\begin{bmatrix} X_{s1} \\ X_{s2} \end{bmatrix} = \sum_{u=0}^3 C_{su}^{inv} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_{0u} - X_{2u} \\ \frac{1}{2} X_{1u} - X_{3u} \end{bmatrix}, \quad s = 0, \dots, 3 \quad (3-12)$$

According to Equation 3-9, Equation 3-10, Equation 3-11, and Equation 3-12, the hardware of direct 2-D transform coding is proposed as shown in Fig. 3-26.

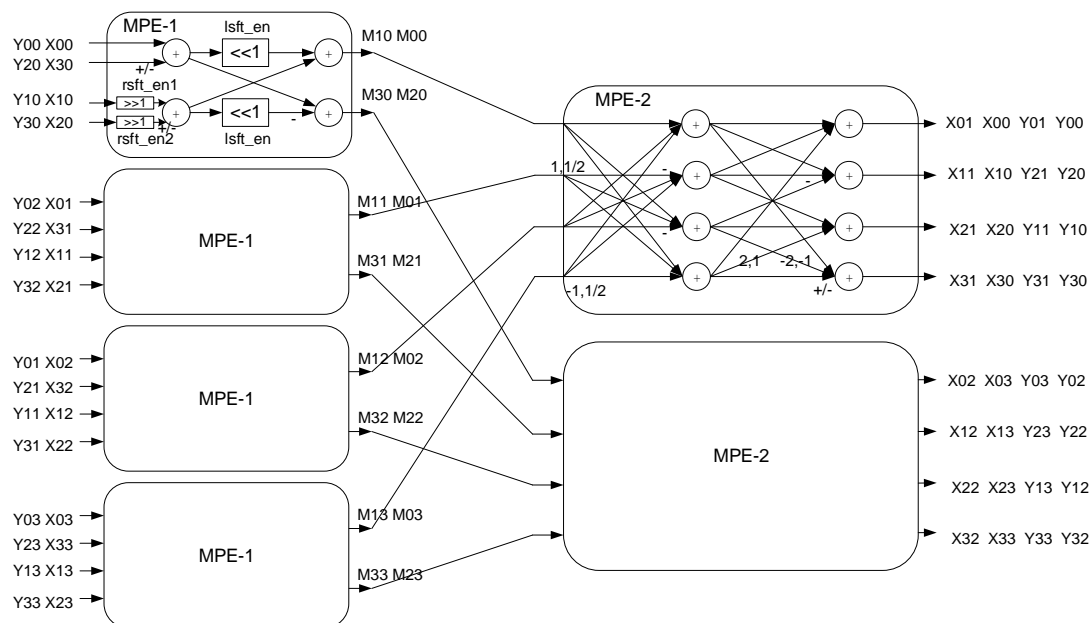


Fig. 3-26. Hardware of Direct 2-D Transform [23].

In MPE-1 (Multi-transform Processing Element), the enable signal of `lsft_en` is for M_{10} and M_{30} , the enable signals of `rsft_en1` and `rsft_en2` are for inverse transform, the “+” is for M_{00} and M_{20} , the “-” is for M_{10} and M_{30} . In MPE-2, $1/2$, 1 , -1 , and “-” are for inverse transform.

The architecture of DCT, IDCT, Quantization, and De-Quantization is shown in Fig. 3-27. When signal of `sel` is 1, forward transform, Coef. Multiplier & Shifter, and Quantization will be applied to input data. When signal of `sel` is 0, DeQuantization, Coef. Multiplier & Shifter, and inverse transform will be applied to input data.

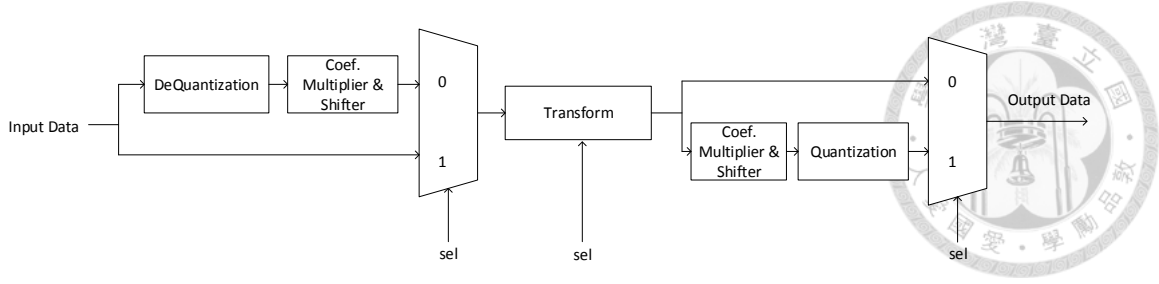


Fig. 3-27. Architectures of DCT, IDCT, Quantization, and De-Quantization.

3.6 Correlation Noise Modeling and Soft Input Computation

The Correlation Noise Modeling adopted in our work is a Laplacian distribution as proposed in [11, 12]. The Laplacian distribution is described by the following probability density function as depicted in Chapter 2.

$$p[WZ(x, y) - SI(x, y)] = \frac{\alpha}{2} \exp[-\alpha |WZ(x, y) - SI(x, y)|], \quad (3-13)$$

In hardware implementation, we will calculate the probability density function using the bits of a current bit plane. The probability of zero and one will be calculated using a lookup table for fixed alpha value and binary side information values. Then, the other lookup table is used to find the log-likelihood ratio shown as Equation 3-14 where $P(1)$ is the probability of 1 and $P(0)$ is the probability of 0.

$$L = \ln\left(\frac{P(1)}{P(0)}\right), \quad (3-14)$$

The hardware of Correlation Noise Modeling and Soft Input Computation is shown in Fig. 3-28. The Lookup Table 1 is used to get the probability of each bit in a current bit plane described in Equation 3-13. The Lookup Table 2 is used to get the log-likelihood ratio described in Equation 3-14. Each bit of a current bit plane, Bit, is inputted to Lookup Table 1. Lookup Table 1 generates the probability of zero and one, $P(0)$ and $P(1)$. Lookup Table 2 generates the log-likelihood ratio, LLR, according to the inputted probability of zero and one.

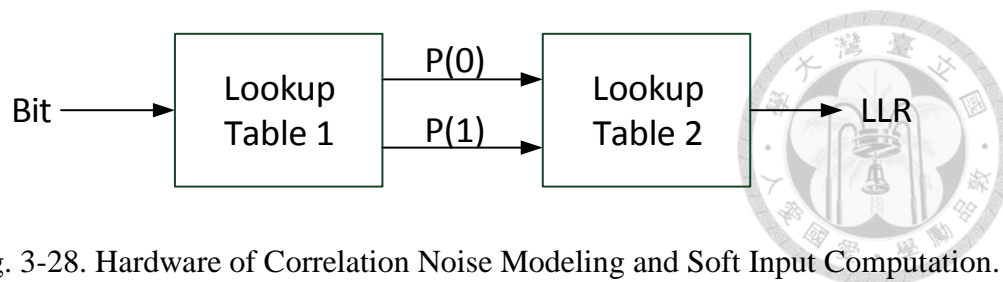


Fig. 3-28. Hardware of Correlation Noise Modeling and Soft Input Computation.

3.7 Reconstruction

The Reconstruction is used to reconstruct the Wyner-Ziv frame by adding each motion-compensated pixel and each de-quantized residue value. The hardware of the Reconstruction is shown in Fig. 3-29. The 16 residue values are stored into Registers. One Motion-Compensated Pixel generated by Motion Compensation is inputted at a time. The Control decides which Residue Value is outputted to the adder. Each Reconstructed Pixel is obtained from the output of the adder. An entire reconstructed Wyner-Ziv frame will be generated by processing one pixel at a time.

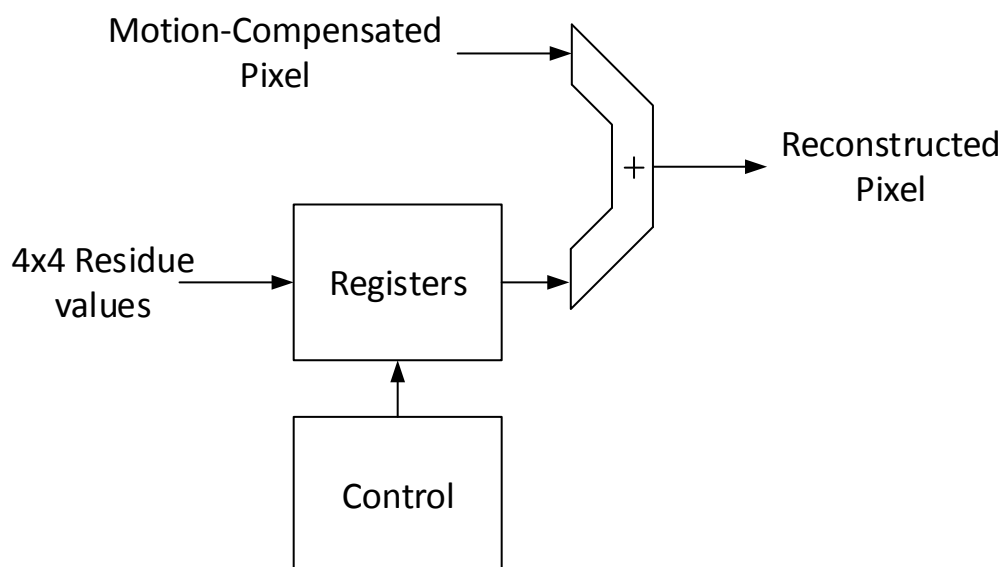


Fig. 3-29. Hardware of Reconstruction.





CHAPTER 4

EXPERIMENT RESULTS

In this chapter, we present the experiment results of our proposed distributed video decoder. First, cell-based design flow will be described. Second, we depict how to test our design. Third, we show chip implementation result. Finally, we describe the simulation results.

4.1 Design Flow

The Cell-based Design Flow is shown in Fig. 4-1. First, we use C++ to simulate the algorithm of our proposed distributed video coding. We implement the hardware of our design by coding RTL. Next, the results of C++ simulation and simulation of NC-Verilog will be compared to confirm function correctness. We use Synopsys Design Compiler to synthesize RTL code into gate-level netlist. Cadence Comformal is used to check logical equivalence. Then, we use Cadence SOC Encounter to floor plane, place, and route. Calibre is used to check DRC (Design Rule Check) and LVS (Layout Versus Schematic). PrimeTime is used to perform static timing analysis checking whether there are setup time and hold time violations. Finally, we will do post-layout simulation. If all the previous steps are passed, we tape out our design. The testing consideration will be described in next section.

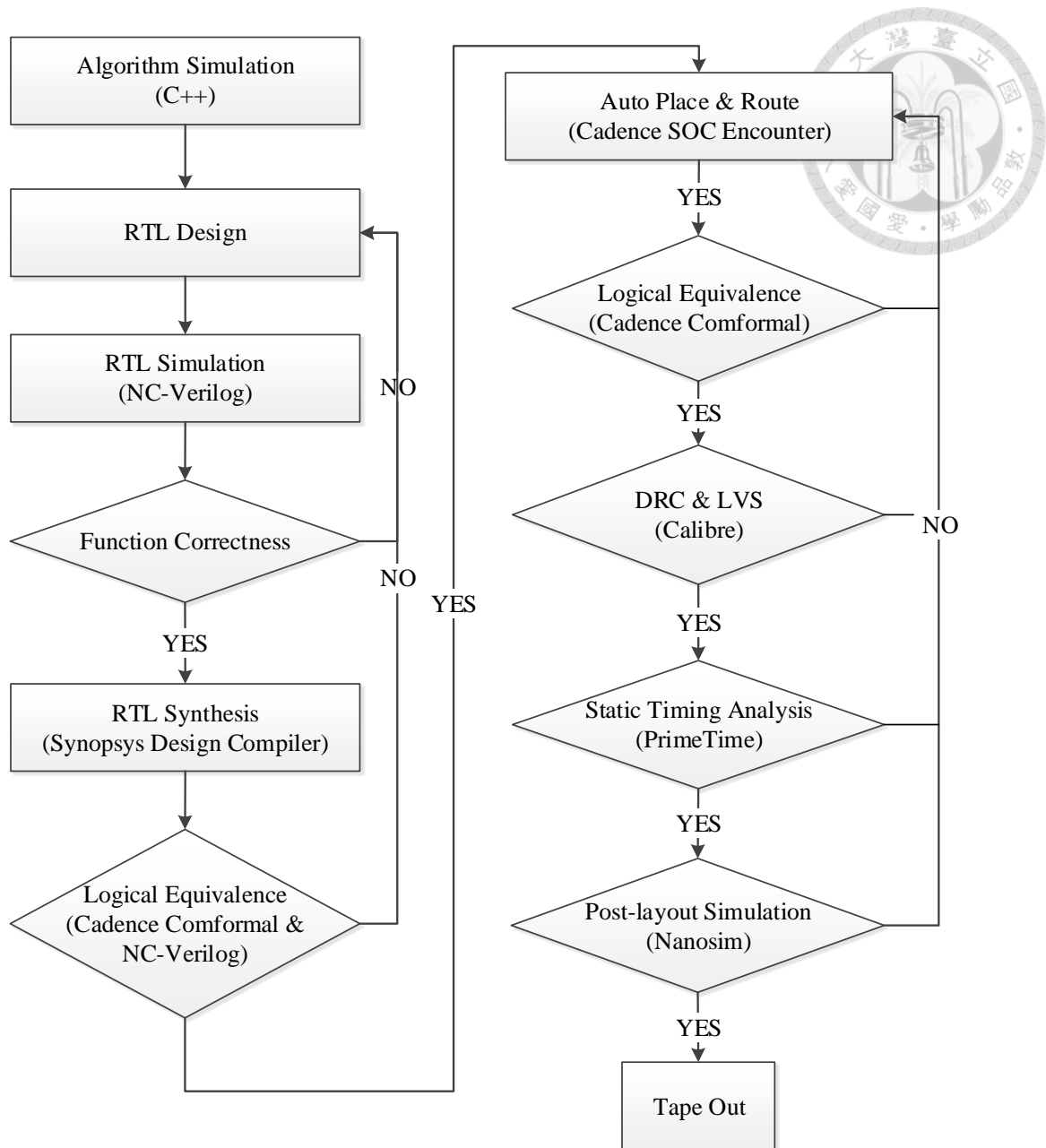



Fig. 4-1. Cell-based Design Flow.

4.2 Testing Consideration

In the stage of RTL synthesis, to check whether our chip is manufactured without any defect, we insert DFT (Design for Testability) circuit to the chip. In our work, we insert 6 scan chains to achieve a fault coverage of 98.93%. The DFT report is shown in Fig. 4-2.



```

Uncollapsed Stuck Fault Summary Report
-----
fault class                code    #faults
-----
Detected                   DT      2093462
Possibly detected          PT           0
Undetectable               UD      21437
ATPG untestable            AU           9
Not detected                ND      1166
-----
total faults                2116074
test coverage               99.94%
fault coverage              98.93%
-----
Pattern Summary Report
-----
#internal patterns          3590
  #basic_scan patterns      3590
-----

```

Fig. 4-2. Uncollapsed Stuck Fault Summary Report.

After chip was taped out, we use CIC93000 testing machine to measure our chip. The measurement considerations are shown in Fig. 4-3, and the measurement flow is shown in Fig. 4-4. First, we use the pattern of ATPG (Automatic Test Pattern Generation) to test whether there is any defect in manufacturing process. In the stage of ATPG, signal of test_se is 0, test pattern is inputted through test_si, and the result of test pattern is outputted through test_so. Then, we input a bitstream of test sequences. If testing of the decoding process is passed, the measurement process is finished. If testing is failed, we will switch test_mode to a different mode to test each part of the decoder including AVC intra decoder and LDPCA decoder.

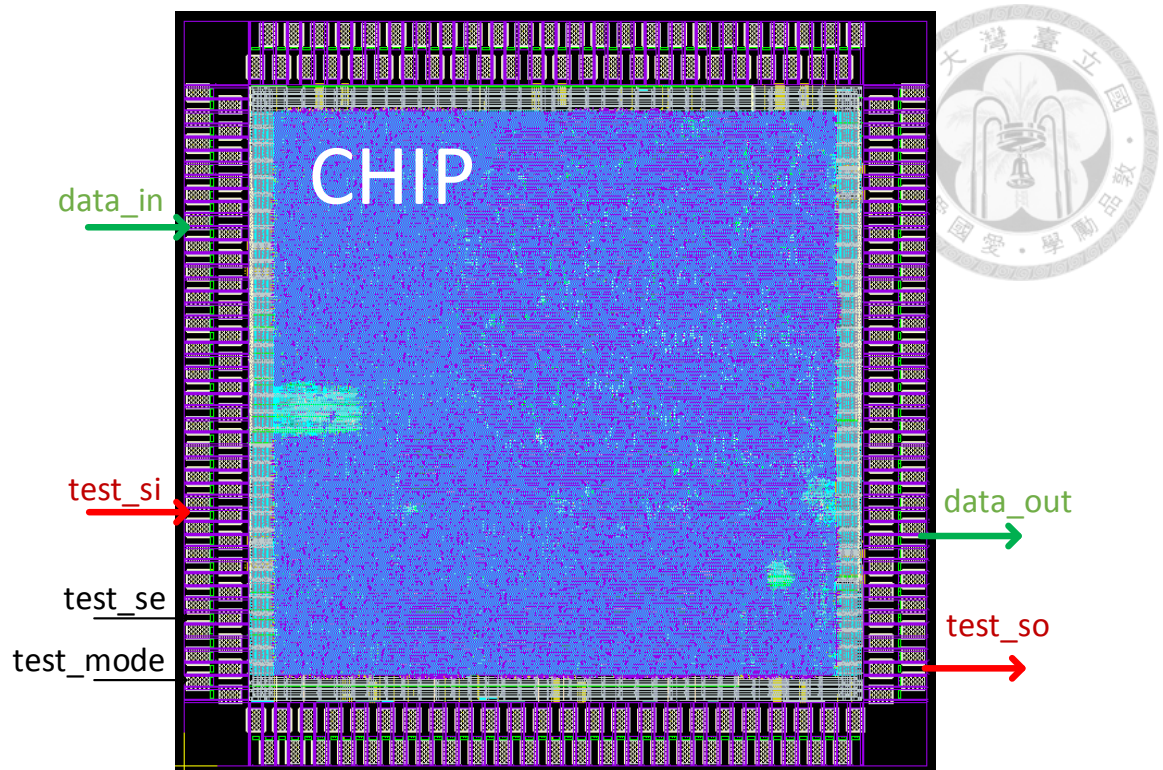


Fig. 4-3. Measurement Considerations.

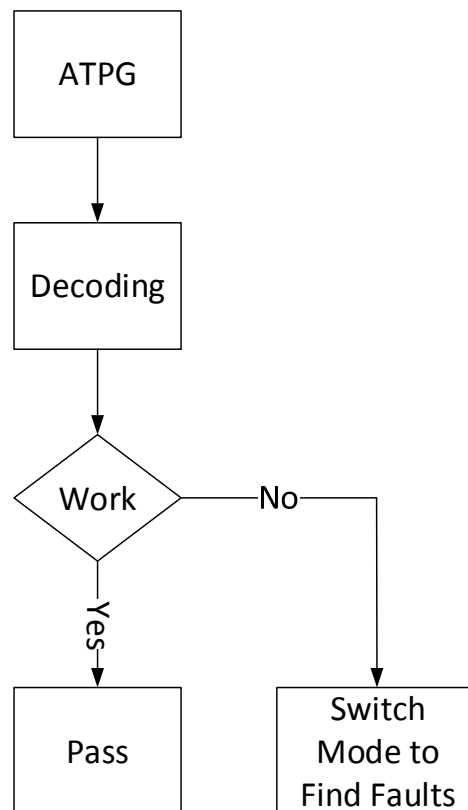


Fig. 4-4. Measurement Flow.

4.3 Chip Implementation Result

The process used in the design of our chip is TSMC 90nm GUTM CMOS. There are totally 191 I/O pins including 45 power/ground pins and 146 signal pins. The chip area is 4.67 mm², and gate count is 690K. The peak frequency of our design is 100MHz. The dynamic power dissipation of our design is 302 mW. The result of place and route is shown in Fig. 4-5. The specification table is listed in Table 4-1. And the synthesis result of LDPCA is listed in Table 4-2.

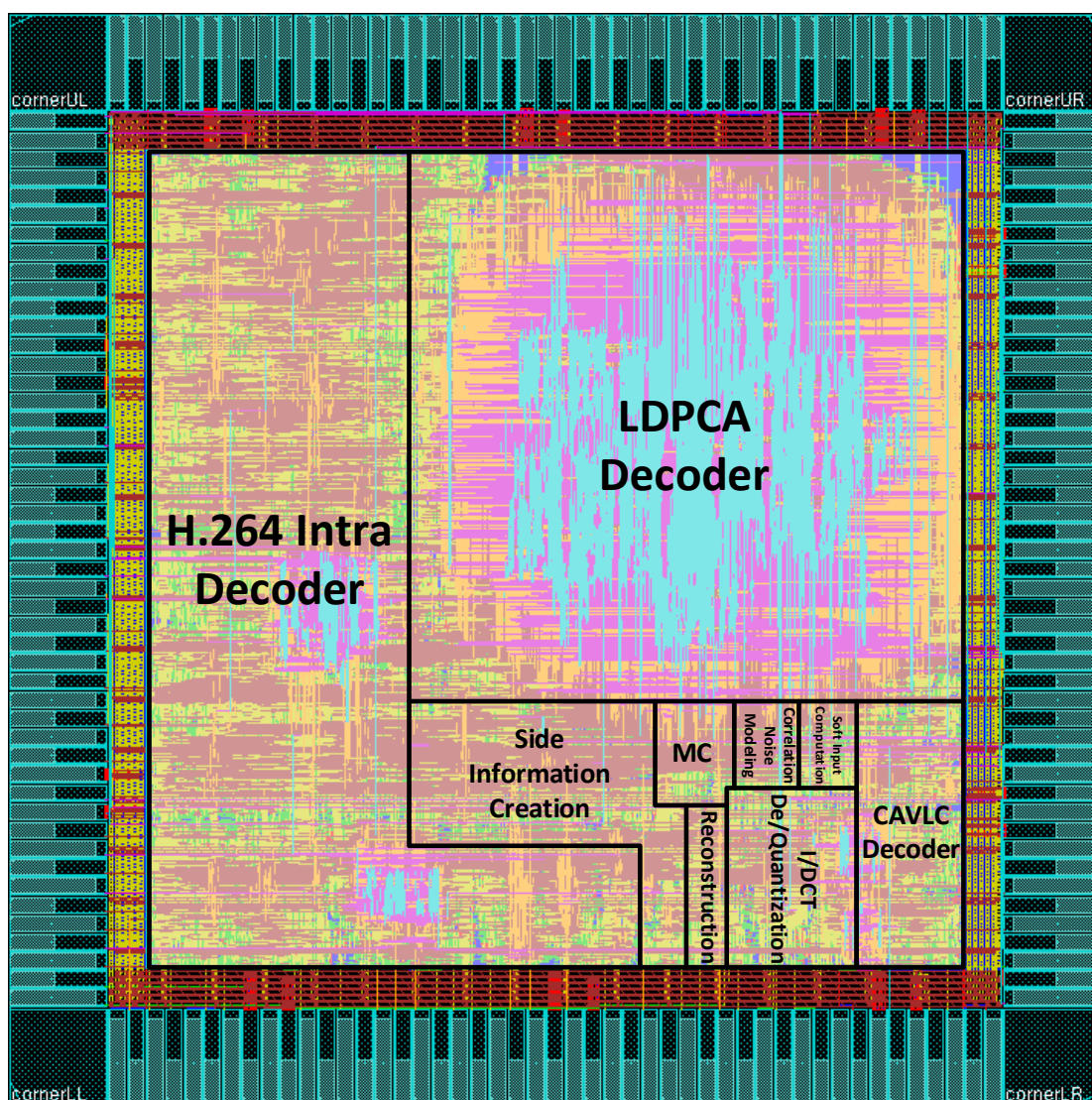


Fig. 4-5. Result of Place and Route.

Table 4-1. Specification Table.

Process	TSMC 90-nm GUTM
Power Supply	1.0 V
Gate Count	690K
Chip Area	4.67 mm ²
Frequency	100 MHz
Pins	191
Resolution	QCIF@30fps
Power	302 mW



Table 4-2. Synthesis Result of LDPCA.

Process	TSMC 90-nm GUTM
Code Length	396 bits
Code Rate	2/66 – 66/66 (support 65 code rates)
Gate Count	364K
Frequency	100 MHz
Power	17.96 mW

4.4 Simulation Results

The RD (Rate Distortion) performance of different test sequences is shown in following figures. In Fig. 4-6, Fig. 4-7, and Fig. 4-8, we compare the RD performance of our design and other designs including representative DVC, DISCOVER, H.264 Intra, and H.264 No Motion. The RD Performance of our hardware version is better than DISCOVER or almost the same with DISCOVER. The difference between software version and hardware version is on the block length of LDPCA and the

simplified soft input computation. In the software version, the block length of LDPCA is 1584 and the soft input computation is the same with the Hybrid DVC. In the hardware version, the block length of LDPCA is 396 and the soft input computation is simplified to using lookup tables.

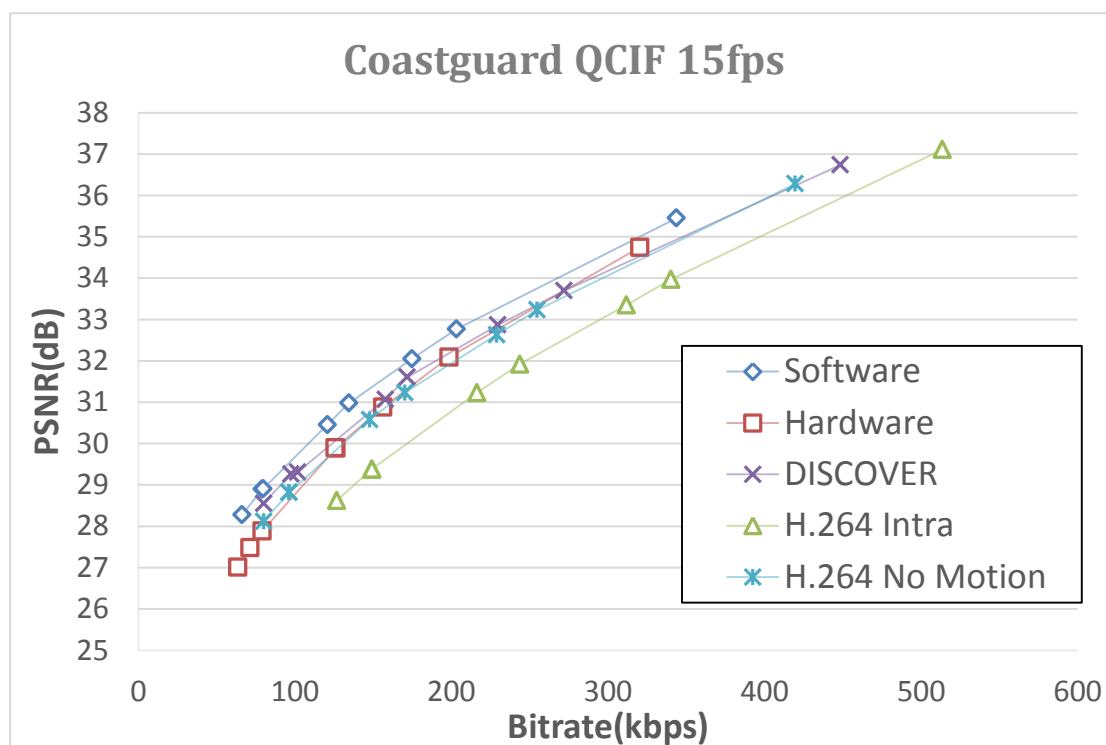


Fig. 4-6. RD Performance of Coastguard QCIF 15fps.

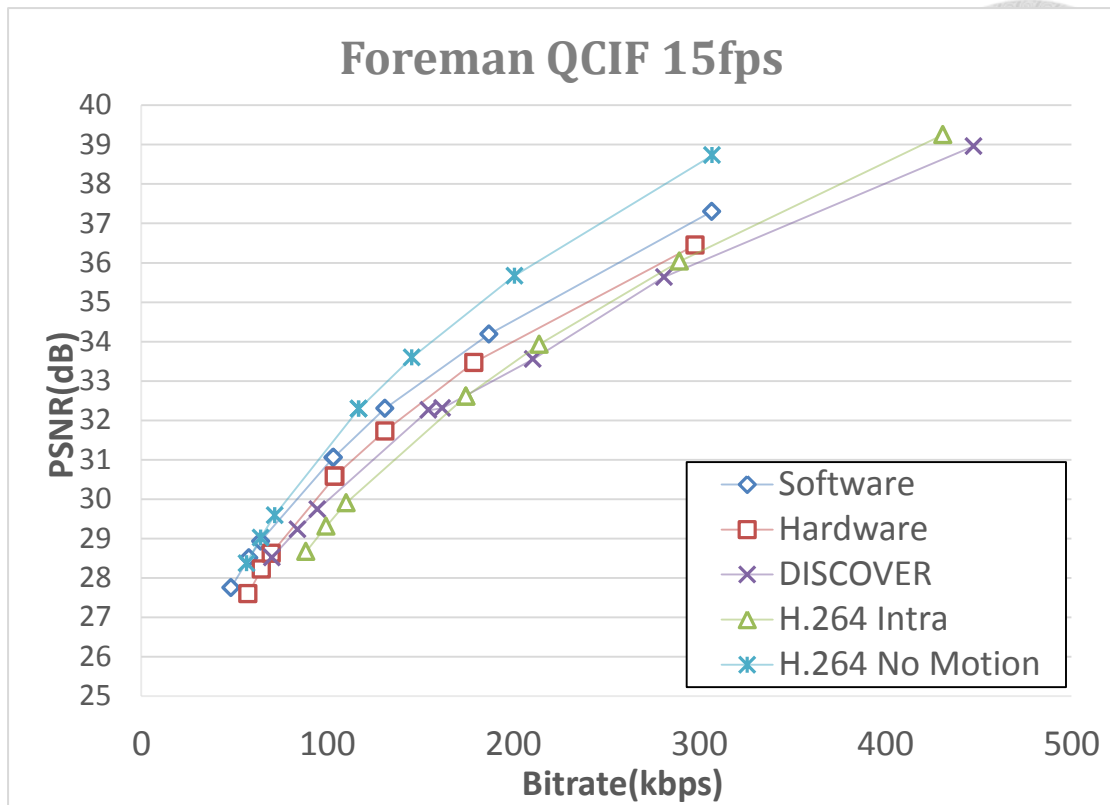


Fig. 4-7. RD Performance of Foreman QCIF 15fps.

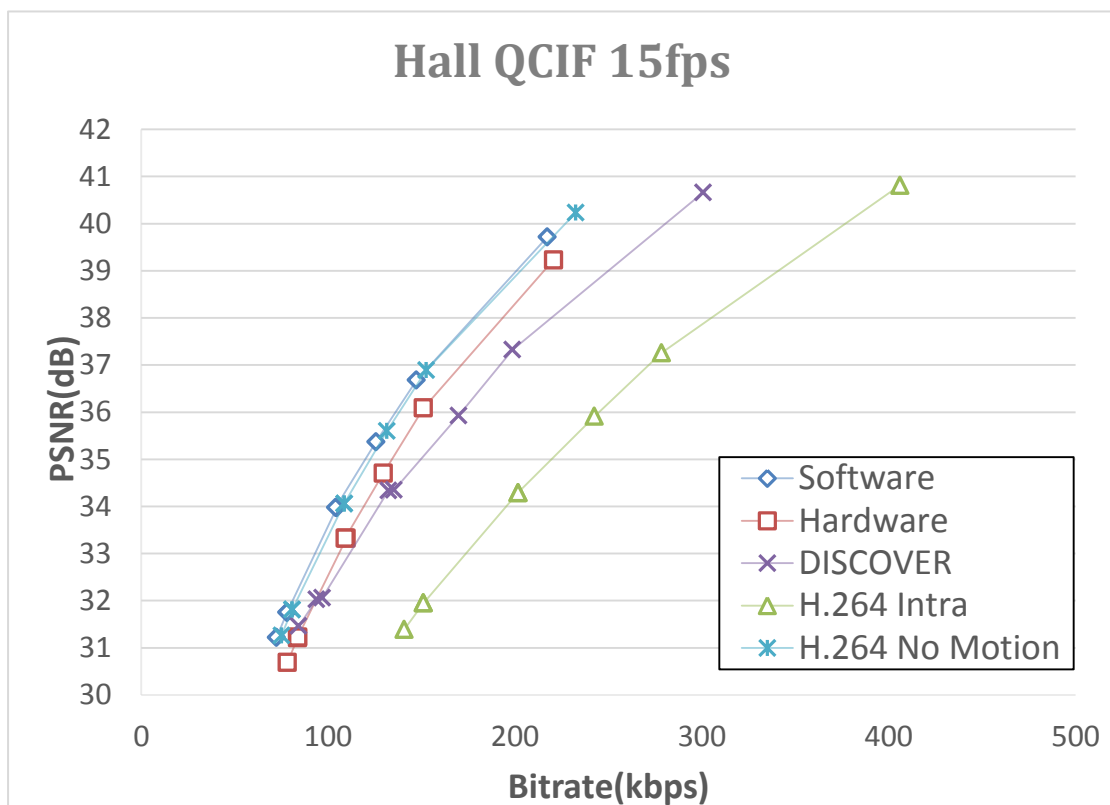


Fig. 4-8. RD Performance of Hall QCIF 15fps.

The reason that we chose LDPCA with a 396 block length is to save hardware cost. Though the throughput of an LDPCA of length 1584 is faster than that of an LDPCA of length 396, the hardware cost of the LDPCA of length 1584 is higher than the LDPCA of length 396 since the hardware costs of their respective comparing trees are proportional to the block length of LDPCA. In our work, we chose an LDPCA of length 396 since it meets the required throughput to decode the test sequences of QCIF videos in real time. The rate performance of a regular degree-3 LDPCA of length 396 bits over BSC (Binary Symmetric Channel) is shown in Fig. 4-9.

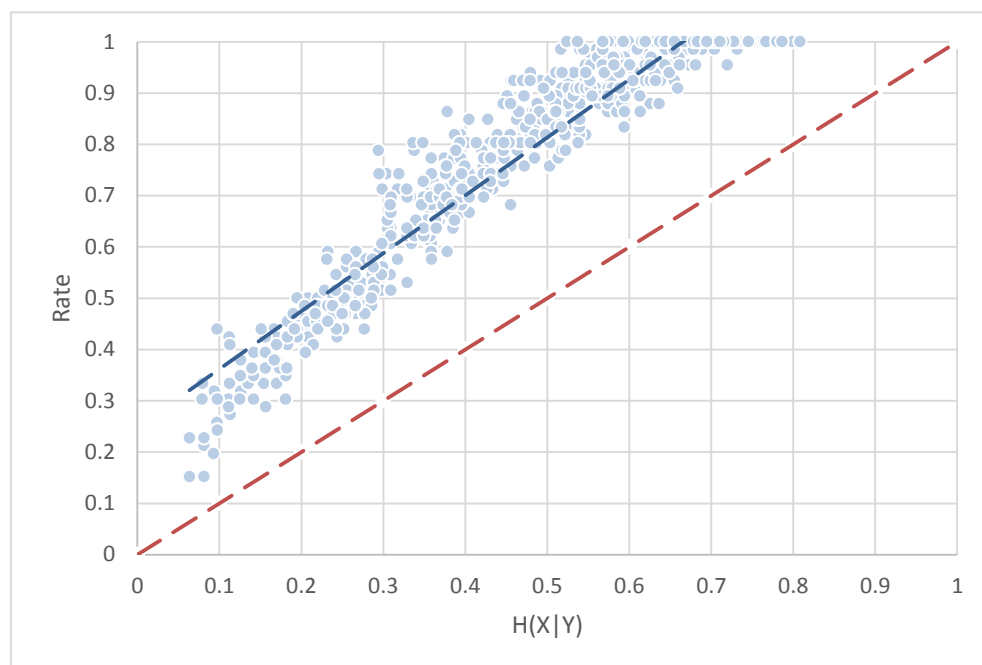


Fig. 4-9. Rate Performance of Regular Degree-3 LDPCA of Length 396.





CHAPTER 5

CONCLUSION

In this Thesis, we implemented a distributed video decoder which can decode video sequences in real time. We presented the architecture of a distributed video codec which RD performance is better than the presentative distributed video codec, DISCOVER. The crucial challenge for a real-time distributed video decoder is that the computing complexities of side information creation and LDPCA are high. Our approach is to design a hardware of distributed video decoder, which is not only focused on improving the architectures of LDPCA and side information creation but also considers the trade-off between hardware cost and performance.

A prototype of our proposed DVC decoder was implemented in TSMC 90nm GUTM process technology and fabricated through CIC. Our decoder can meet the requirement of decoding a QCIF video with a speed of 30fps. The maximum operation frequency of our design in the post-layout stage is 100MHz. The chip area is 4.67 mm², and gate count is 690K.

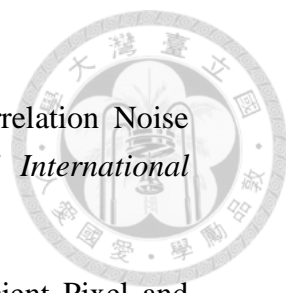


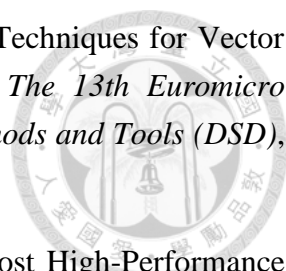


REFERENCE

- [1] *Information Technology—Coding of Audio-Visual Objects—Part 10: Advanced Video Coding (H.264)*, ISO/IEC JTC 1/SC 29 14496-10, 2004.
- [2] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed Video Coding," *Proc. IEEE*, vol.93, no.1, pp. 71-83, Jan. 2005.
- [3] D. Slepian and J. K. Wolf, "Noiseless Coding of Correlated Information Sources," *IEEE Trans. on Information Theory*, vol. 19, no. 4, pp. 471-480, Jul. 1973.
- [4] D. Varodayan, A. Aaron, and B. Girod, "Rate-Adaptive Codes for Distributed Source Coding," *EURASIP Signal Processing Journal, Special Section on Distributed Source Coding*, vol. 86, no. 11, pp. 3123-3130, Nov. 2006.
- [5] DISCOVER DVC Final Results: <http://www.img.lx.it.pt/~discover/home.html>.
- [6] A. Aaron, R. Zhang, and B. Girod, "Wyner-Ziv Coding of Motion Video," *Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, vol.1, pp. 240-244, Nov. 2002.
- [7] A. Aaron, S. Rane, E. Setton, and B. Girod, "Transform-Domain Wyner-Ziv Codec for Video," *Proc. of Society of Photo-Optical Instrumentation Engineers – Visual Communications and Image Processing*, San Jose, CA, USA, Jan. 2004.
- [8] C.-C. Chiu, S.-Y. Chien, C.-H. Lee, V.S. Somayazulu, and Y.-K. Chen, "Hybrid Distributed Video Coding with Frame Level Coding Mode Selection," *IEEE International Conference on Image Processing (ICIP)*, pp. 1561-1564, Oct. 2012.
- [9] J. Ascenso, C. Brites, F. Pereira, "Content Adaptive Wyner-ZIV Video Coding Driven by Motion Activity," *IEEE International Conference on Image Processing*, pp. 605-608, Oct. 2006.
- [10] L. Alparone, M. Barni, F. Bartolini, and V. Cappellini, "Adaptively Weighted Vector-Median Filters for Motion-Fields Smoothing," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.4, pp. 2267-2270,

May 1996.

- 
- [11] C. Brites, J. Ascenso, and F. Pereira, "Studying Temporal Correlation Noise Modeling for Pixel Based Wyner-Ziv Video Coding," *IEEE International Conference on Image Processing*, pp.273-276, Oct. 2006.
- [12] C. Brites and F. Pereira, "Correlation Noise Modeling for Efficient Pixel and Transform Domain Wyner-Ziv Video Coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.18, no.9, pp.1177-1190, Sep. 2008.
- [13] R. Martins, C. Brites, J. Ascenso, and F. Pereira, "Refining Side Information for Improved Transform Domain Wyner-Ziv Video Coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.19, no.9, pp.1327-1341, Sep. 2009.
- [14] R. G. Gallager, "Low Density Parity Check Codes," *IRE Trans. Information Theory*, vol. IT-8, no. 1, pp. 21-28, Jan. 1962.
- [15] R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Information Theory*, vol. IT-27, no.5, pp. 533-547, Sep. 1981.
- [16] S. Y. Chung, G. D. Forney, T. J. Richardson, and R. L. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Communication Letters*, vol. 5, no. 2, pp. 58-60, Feb. 2001.
- [17] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of Binary Sources with Side Information at the Decoder using LDPC Codes," *IEEE Communications Letters*, vol. 6, no.10, pp.440-442, Oct. 2002.
- [18] M. P. C. Fossorier, M. Mihaljević, and H. Imai, "Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation," *IEEE Trans. Communications*, vol. 47, no.5, pp.673-680, May 1999.
- [19] C. L. Wey, M. D. Shieh, and S. Y. Lin, "Algorithms of Finding the First Two Minimum Values and their Hardware Implementation," *IEEE Trans. on Circuits and Systems-I: Regular Papers*, val. 55, no. 11, pp. 3430-3437, Dec. 2008.
- [20] L. Alparone, M. Barni, F. Bartolini, and V. Cappellini, "Adaptively Weighted Vector-Median Filters for Motion-Fields Smoothing," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.4, pp. 2267-2270, May 1996.

- 
- [21] O. Tasdizen and I. Hamzaoglu, "Computation Reduction Techniques for Vector Median Filtering and their Hardware Implementation," *The 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*, pp.731-736, Sep. 2010.
- [22] H.-C. Chang, C.-C. Lin, and J.-I. Guo, "A Novel Low-Cost High-Performance VLSI Architecture for MPEG-4 AVC/H.264 CAVLC Decoding," *IEEE International Symposium on Circuits and Systems*, vol.6, pp. 6110-6113, May 2005.
- [23] K.-H. Chen, J.-I. Guo, and J.-S. Wang, "A High-Performance Direct 2-D Transform Coding IP Design for MPEG-4AVC/H.264," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.16, no.4, pp. 472-483, April 2006.