國立交通大學

電子工程學系　電子研究所碩士班

碩 士 論 文

低複雜度無失真視訊壓縮

Low Complexity Lossless Video Compression

研 究 生：方耀諄

指導教授：蔣迪豪　博士

中 華 民 國 九 十 三 年 七 月

低複雜度無失真視訊壓縮

Low Complexity Lossless Video Compression

研究生: 方耀諄　　Student: Yao-Chun Fang
指導教授: 蔣迪豪　　Advisor: Tihao Chiang

國 立 交 通 大 學
電子工程學系 電子研究所碩士班
碩 士 論 文

A Thesis
Submitted to Department of Electronics Engineering & Institute of Electronics
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Electronics Engineering

July 2004

HsinChu, Taiwan, Republic of China

中華民國九十三年七月

低複雜度無失真視訊壓縮

研究生: 方耀諄　　指導教授: 蔣迪豪 博士

國立交通大學

電子工程學系 電子研究所碩士班

摘要

本篇論文提出一個適應性無失真視訊壓縮演算法，以”行(line)”為處理單元，擁有低複雜度、延遲時間短的特性，可應用於互動式多媒體。對於高解析度視訊的傳輸及儲存裝置間的資料交換，為了減少傳輸所需的頻寬及節省儲存空間，同時又不造成資料的誤差，無失真的視訊壓縮演算法是必要的。考慮多媒體應用之低複雜度及低延遲的需求，我們提出低複雜度無失真視訊壓縮演算法(Line-based Lossless Video Compression, LALVC)。低複雜度無失真視訊壓縮演算法 LALVC 具有以下特性—單次資料分析(one-pass)處理，以掃瞄的順序(raster-scan order)處理，不採用轉換(transformation)而只用簡單的本文預測法(Context based predictors)，同時兼顧時間及二維圖形中的關係來做有效的編碼。在時間軸上的圖像關係，採用零動量(zero-motion)的預測法來除去多餘資料以達到高效率的視訊壓縮。此外針對自然影像及電腦圖形的特性不同，利用不同模式的處理，透過一個簡單而有效的模式判定(mode decision)，來提升編碼的效率。另外 entropy 編碼方法採用 Golomb code—同樣具有低複雜度，又有很高的編碼效率，且易於用硬體實現。LALVC 演算法經由摸擬的結果，在時間軸上的預測法，搭配”行”為處理單元的視訊編碼方法，可以有效地提升壓縮效率。計算量上，相對於影像無失真演算法 JPEG-LS，雖僅增加了些加法運算，即可有效地提升整體的編碼效率，且達到即時壓縮的效果。同時為了展示 LALVC 演算法的硬體實現與即時視訊壓縮，本論文也利用 ASIC 晶片設計方式實現了 LALVC 演算法的硬體架構。

Low-Complexity Lossless Video Compression

Student: Yao-Chun Fang Advisor: Dr. Tihao Chiang

Department of Electronic Engineering &
Institute of Electronics
National Chiao Tung University

Abstract

We present a line-based adaptive lossless video compression (LALVC) algorithm for interactive multimedia applications that demand low complexity and low latency. Communications between high-resolution display and storage devices require high bandwidth for exchanging raw data. To reduce the cost of video transmission without losing data accuracy, a lossless video compression is necessary. Considering low complexity and low delay, the proposed LALVC adopts a simple and yet efficient architecture that consists of one-pass, raster-scan, transform-free coding process with a simple predictor. For low latency, zero-motion prediction and single frame buffer are used to reduce temporal redundancy. In addition, to maximize the coding efficiency for both natural and computer-generated video sequences, LALVC adaptively selects the best coding mode for each line in a frame. The entropy coding of each line is based on Golomb code to enhance coding efficiency with less computation load and is easy for hardware realization. The simulation results show that temporal preprocessing and line-based mode decision of LALVC can increase compression ratio with properly increased complexity as compared to that of JPEG-LS. In addition, to demonstrate easy hardware realization and real-time video compression, we realize hardware architecture of LALVC for ASIC design.

誌　　謝

這兩年來的碩士生活，首先感謝指導教授 蔣迪豪 博士，提供一個不錯的研究制度和環境。實驗室的大學長 俊能 花了許多時間和我討論，給予許多建議; 俊毅(大李)，博班學長，帶了我兩年多，從研究音訊開始，到轉換跑道，改作視訊壓縮，不管在研究方向和生活經驗，熱心協助，是個好學長，也是個好朋友; 之後也加入了一位學弟，盈閔，也協助了許多，讓我在作研究上，隨時有討論的對象。有了這幾位的參與，讓這篇論文更為完整。

在 commlab 的五樓實驗室，待了兩年，有一齊奮鬥的夥伴，小強、大范、子良、俊安、名彥，大家一起為學業打拚，為生活尋找樂趣，有同伴的感覺真的很好。還有兩位可愛學妹，汝芩和瑛姿，相處了一年，打球、聊天，很開心的回憶。另外實驗室其他的學長姊，郁男、肥肥等等，或多或少提供了些幫助，在此感謝。

另外，國維，在生活上幫了我不少忙，幫我找了家教和助教，經濟上無後顧之憂。

最後對於家人的背後支持，獻上我衷心的祝福與感謝。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data compression is ubiquitous in various multimedia applications. In a network system, bandwidth is limited. Data compression can increase transmission efficiency and decrease bandwidth usage. In a storage system, compression provides higher storage density by removing the redundancy within delivering contents.

For multimedia data, lossy compression is widely adopted in consumer applications. There are still some special applications that essentially require lossless video compression. For example, lossless compression is being used to preserve high quality media contents including images and videos for surveillance systems or medical diagnosis. Compression reduces the data volume for storage or transmission. In addition, lossless compression can support the content exchange between the different systems with minimal amount of bandwidth and without reducing quality of media contents.

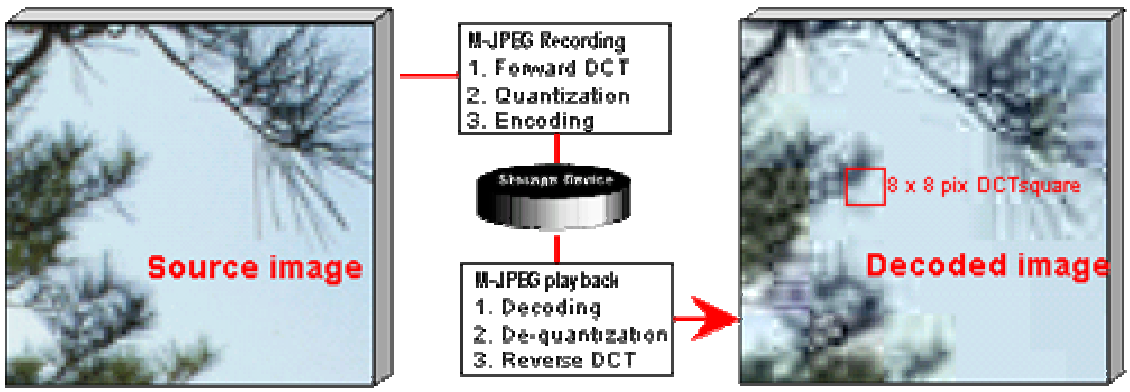


Figure 1. Block diagram of Motion-JPEG [10]

From the existing standards, few standards are defined for the lossless video contents due to small commercial demand. For high quality applications, such as MPEG-2 video for DVD format, video contents have near perceptually lossless quality, which still does not fit the application with the lossless requirement. For lossless video content compression, Motion JPEG (M-JPEG) is a simple extension of JPEG. In Figure 1, M-JPEG compresses the successive images by a way similar to concatenating of the compressed image bitstream by

JPEG. In addition, M-JPEG does not adopt temporal prediction that is employed in existing lossy video coding standards. JPEG contains DCT transform, quantization, and entropy coder. The lossless mode of JPEG does not include the functions of DCT and quantization for retaining the content precision.

For the applications demanding the loss video compression, our idea is to define a closed system to enhance the performance of lossless video compression. Our main goal focuses on the low complexity for the FPGA implementation or ASIC design. In the research process, the overall algorithm can be classified into three parts. The first part is temporal correlation issue. For video with low or zero motion, the neighboring frames are highly correlated, which indicates there exists temporal redundancy within the video sequence. The temporal redundancy can be removed or reduced with a temporal prediction method. The second part is spatial domain issue, for which various lossless image compression algorithms are surveyed for references.

Among the image compression methods, the well-known candidates are CALIC [1], JPEG-LS (LOCO-I) [2]. The algorithms are designed for image compression only using intra-prediction to de-correlate the redundancy within an image. Some extended versions of image coders are SILIC [3] or inter-band algorithm [4]. The compression algorithms that extend from the gray-level images to the color images have added the simple predictor to de-correlate the information existing between different color planes, i.e. R, G, B planes.

For the video contents, there exist both temporal and spatial relationship. Temporal correlation is not visible in the image coding. Considering the temporal factor, inter-frame redundancy should be removed before being fed into the entropy coder, which can increase the overall coding efficiency.

In addition to the algorithms of an image coder, inter-frame prediction between neighboring frames (pictures) is used for efficient video coding. The video coding efficiency is increased based on the existing inter-frame correlation.

In the thesis, we propose a low-complexity algorithm for lossless video compression. Both temporal and spatial domain redundancy could be efficiently removed by a suitable predictor. Combining the three removal approaches of redundancy including temporal, spatial, and coding redundancy, we can achieve the efficient coding efficiency at very low-complexity.

In chapter 2, we review the prior work about the lossless video compression. For spatial domain prediction, we survey related documents of lossless image coders. Chapter 3 explains

the line-based adaptive lossless video compression (LALVC) algorithm in detail. In addition, we discuss the hardware implementation architecture of LALVC for an ASIC design. The simulation results are given in chapter 4. Finally, Chapter 5 draws the contribution and future work.

Chapter 2

Lossless Data Compression Methods

For lossless video compression, existing approaches are introduced. The major functions included in lossless video compression can be divided into two parts covering prediction methods and entropy coder. The prediction methods can help remove the temporal and spatial redundancy in the raw data. The prediction will generate residuals after subtracting the original values with predicted values, which maps the PDF (Probability Density Function) of raw data into another PDF of residuals. With a particular PDF of residuals, the prediction process can improve the coding efficiency of entropy coders. For example, JPEG-LS [2] uses the prediction residuals for representing the original data by removing the similarity of raw data. The PDF of prediction residuals with (near) zero mean is close to a Laplacian distribution. With the entropy coder like Golomb code used in JPEG-LS, we encode the residuals into the bit-stream to achieve high compression.

2.1 Prediction Methods

The key module in lossless coders is a predictor. A good predictor can promote the efficiency of entropy coder by removing the redundancy and minimizing the residual energy. The following section will introduce the related work about predictions from three aspects including inter-frame, inter-component and intra-frame.

2.1.1 Inter-frame Prediction

The special characteristic of video is that the temporally neighboring images have high similarity, which can be removed by temporal prediction. In addition to the spatial predictors in the image coders, inter frame prediction is further investigated in the video compression for enhancing the overall coding gain. In addition, inter-pixel statistical correlation can be found in [5]. The sequence discussed is Mobile as shown in Figure 2. The statistical correlation in the paper is listed in Table 1 and Table 2.

Figure 2. The Mobile sequence

Table 1. Correlation coefficients between 3*3 pixel neighborhoods in the same positions in frame[*i*] and frame[*i-1*]

Red			Green			Blue		
0.88	0.96	0.92	0.89	0.97	0.92	0.91	0.97	0.93
0.87	0.93	0.89	0.88	0.94	0.90	0.90	0.95	0.92
0.83	0.88	0.85	0.85	0.89	0.86	0.86	0.90	0.88

Table 2. Correlation coefficients between 3*3 pixel neighborhoods in the same positions in frame[*i*] and frame[*i-2*]

Red			Green			Blue		
0.84	0.90	0.87	0.85	0.91	0.88	0.87	0.92	0.90
0.82	0.87	0.84	0.83	0.88	0.86	0.86	0.90	0.88
0.79	0.84	0.81	0.81	0.85	0.83	0.83	0.87	0.85

Form Table 1 and Table 2, high correlation exists between neighboring frames. In the three-dimensional (3-D) prediction model, LOCO 3-D [6] improved coding gain by choosing the best prediction value from 4 predictors. LOPT 3-D [6] used the adaptive matrix to achieve the prediction accuracy. In addition, both LOCO 3-D and LOPT 3-D apply motion search to promote the performance of the predictor. The motion search needs heavy computation, which is a bottleneck for hardware implementation of video encoders.

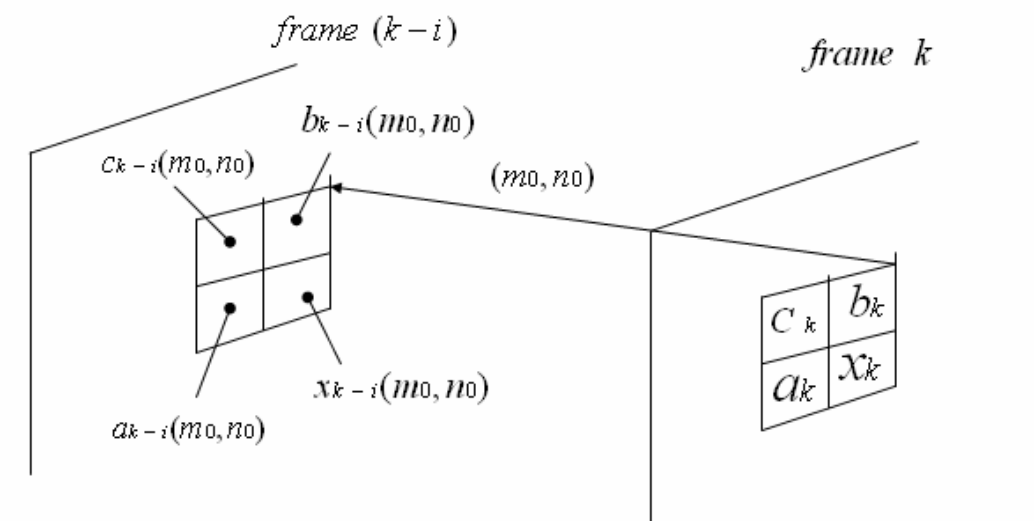


Figure 3. Pixels from frame k and k-I used to calculate the prediction in LOCO 3-D [6]

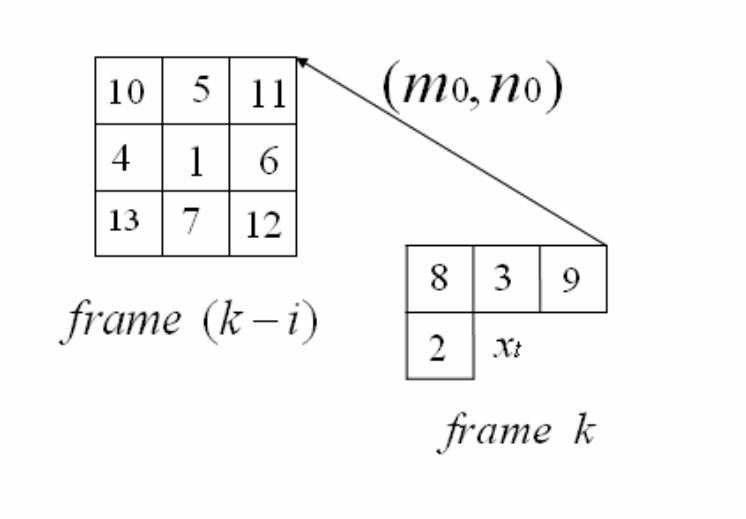


Figure 4. 3-D enumerations of pixels according to their distance from the current pixel X_t. [6]

2.1.2 Inter-Component Prediction

With some modifications to the gray-level image coders, the same coding techniques can be applied for color images containing three color planes (R, G, B or Y, U, V). The color image is applied to each of three color planes individually.

Existing color image coding schemes are introduced as follows. SICLIC [3] bases on LOCO-I [2] to develop an algorithm to process the color images of RGB format. The G plane uses the context models identical to the LOCO-I. The remaining R and B planes are predicted with the modified inter-band context models as shown in Figure 5.

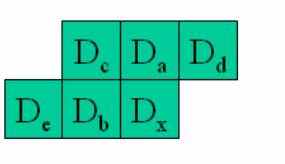


Figure 5. SILIC causal templates

In Figure 5, the symbol “D” represents the difference between the pixel in current plane and the one in another color plane. In Eq.1, $X_{i,j}$ denotes pixels in the G plane that is encoded first at the SICLIC scheme. With $X_{i,j}$, we can predict $Y_{i,j}$ of the plane B or G. by Eq.1. The group of value D forms the template for prediction.

$$D_{i,j} = X_{i,j} - Y_{i,j} \qquad \text{Eq. 1}$$

For the color images, temporal and spectral relationships between the color components covering RGB, or YCbCr, or YUV can be used for improving overall coding efficiency. In addition, coding with spectral component can perform better than coding with temporal component [7]. Therefore, motion compensation does not show absolute superiority for lossless video compression. To address the coding efficiency of loss video coding, the best solution is the hybrid of spectral and temporal prediction.

2.1.3 Intra-Frame Prediction

Intra-prediction uses casual templates to predict the value within the same image. According to the low complexity structure, JPEG-LS is an efficient coder for lossless image compression. Therefore, we take JPEG-LS as the reference model for advanced video processing.

The structure of JPEG-LS is highly related with Sunset algorithm [8], which has pioneered the following properties of lossless image compression scheme.

1. $\hat{x}_{t+1}$ is guessed for the next sample x_{t+1} based on a causal template of the available past data.
2. The context corresponding to the occurrence of $\hat{x}_{t+1}$ is a function of the causal template.
3. A probability model of the prediction residual $\varepsilon_{t+1} = x_{t+1} - \hat{x}_{t+1}$ is based on the context of $\hat{x}_{t+1}$

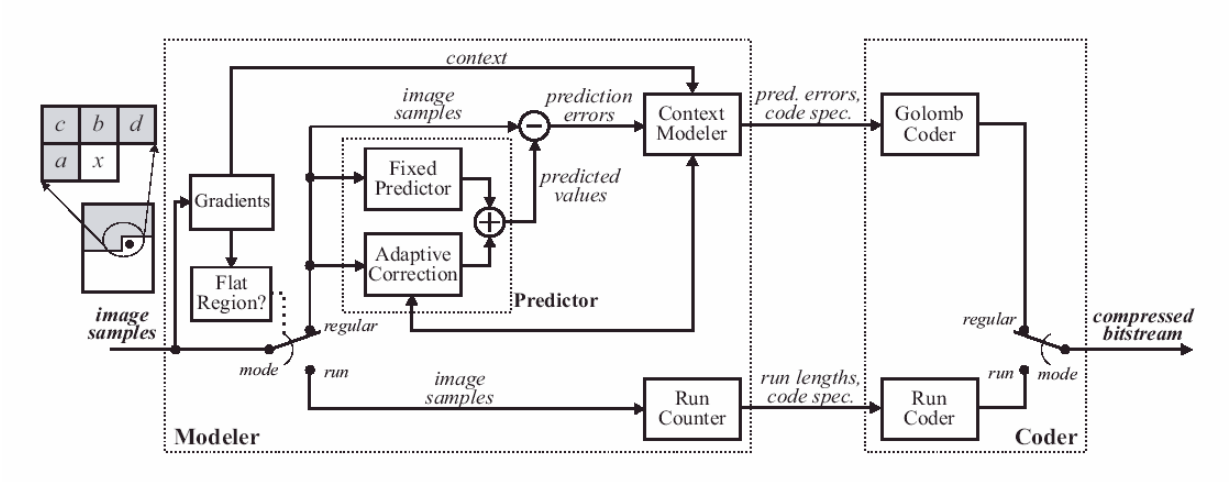


Figure 6. JPEG-LS block diagram [2]

In Figure 6, the left-hand part of diagram is a context-model predictor and the right-hand part is an entropy coder. At the context model, pixel *x* is predicted by the template consisting of *a*, *b*, *c*, and d around *x*. The template helps produce gradients used for the context model, which operates in intra-frame prediction.

G1=*d*-*b*, G2=*b*-*c*, and G3=*c*-*a* Eq. 2

The context model is decided by gradients G1, G2, and G3 that are computed from the template in Eq. 2. According to the template, G1, G2, and G3 in Eq.2 are used for detecting the image property of local region. If G1=G2=G3=0, the region is categorized as a flat region. The flat region can use the run-length coder to encode the pixel x. If the condition leads to the non-flat region, the fixed predictor in Eq. 3 is used.

$$\hat{x} = \begin{cases} \min(a,b) & if\ c >= \max(a,b) \\ \max(a,b) & if\ c <= \min(a,b) \\ a+b-c & otherwise \end{cases} \quad \text{Eq. 3}$$

The fixed predictor lacks flexibility to process the various data. In JPEG-LS, the fixed predictor provides the context modeling to gather the statistics of the prediction error. By avoiding context dilution, quantization of G1, G2, and G3 (wide range, -255 to +255) into Q1, Q2, and Q3 (narrow range, -4 to +4) reduces the number of states for statistics.

$$
\begin{aligned}
D_i \le -T_3 &\Rightarrow Q_i = -4 \\
-T_3 < D_i \le -T_2 &\Rightarrow Q_i = -3 \\
-T_2 < D_i \le -T_1 &\Rightarrow Q_i = -2 \\
-T_1 < D_i \le 0 &\Rightarrow Q_i = -1 \\
D_i = 0 &\Rightarrow Q_i = 0 \\
0 < D_i \le T_1 &\Rightarrow Q_i = 1 \\
T_1 < D_i \le T_2 &\Rightarrow Q_i = 2 \\
T_2 < D_i \le T_3 &\Rightarrow Q_i = 3 \\
T_3 < D_i &\Rightarrow Q_i = 4
\end{aligned}
$$

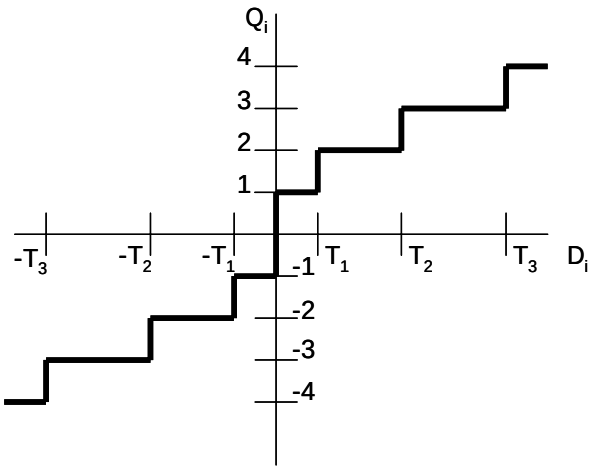


Figure 7. Non-uniform quantizer used in JPEG-LS for context modeling [2].

Based on Q1, Q2, and Q3, each prediction error corresponding to certain state (Q1, Q2, Q3) is recorded. In addition, adaptive error feedback increases the prediction accuracy. The following paragraphs give the explanation of adaptive correction.

The gradients calculated from the template are grouped into Q1, Q2, and Q3 after quantization. Q1, Q2, and Q3 are used to decide the entry of the context model states. For coding each pixel, Q1, Q2, and Q3 are derived based on the template pixels. The initial prediction from the fixed predictor will add the average error from the context model. After prediction, a residual prior to coding is produced. The context modeler is implemented by four sets of array, named N[Q], A[Q], B[Q], and C[Q] (Q is the entry number derived form Q1,Q2, and Q3) respectively. N[Q] indicates the counts of the occurrence corresponding to the state Q. A[Q] is the accumulated prediction error of the state Q and B[Q] is prediction error remainder of the state Q. C[Q] means the bias cancellation value of the state Q.

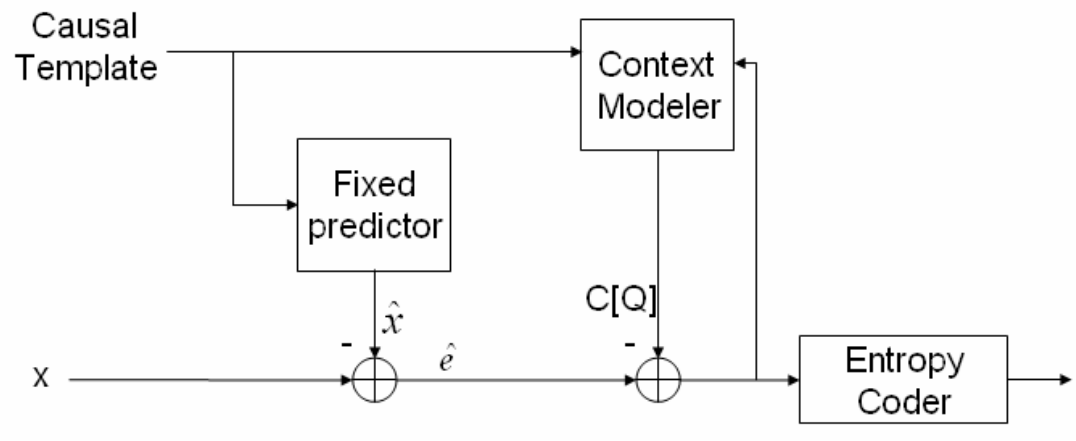


Figure 8. The cooperation between fixed predictor and context modeler.

The major bias correction factor is C[Q]. The bias correction is based on N[Q], A[Q] and B[Q].

2.1.4 Mode Decision

There are several kinds of predictions suitable for prediction in video compression. By the reference of temporal, spatial or color domain, single predictor will not always give the best prediction efficiency. To improve the efficiency of using single predictors, we need a optimized decision approach to combine the predictors, which can be called as a hybrid scheme.

If the algorithm includes many predictors consisting temporal, spatial and others, the simple method sends side information to remind the decoder to use the best predictor to reconstruct the value, which will decrease the performance due to the side information.

Another method is to use the causal template to conduct implicitly the prediction, which can remove the overhead for high coding gain. Both encoder and decoder follow the same rule to decide the proper prediction mode. The coding efficiency may be decreased when the prediction template used does not capture the best predictor behavior. Thus there is a trade-off between performance and amount of side information.

The other method is a hybrid method. In [12], both spatial and temporal predictions are calculated first, and then the weight of each prediction is decided by referring the previous prediction results. The hybrid predictor is formulated as in Eq. 4. The prediction error distance is calculated by Eq. 5 and Eq. 6. The weight is decided by the Eq. 7 and Eq. 8. The Eq.4 to Eq. 8 need some heavy computations including dividers.

$$\hat{P}[i,j] = a(i,j)\cdot \hat{P}_S[i,j] + b(i,j)\cdot \hat{P}_T[i,j] \quad \text{Eq. 4}$$

$$ERR_S(i,j) = \left|P_S(i-1,j) - \hat{P}_S(i-1,j)\right| + \left|P_S(i,j-1) - \hat{P}_S(i,j-1)\right| \quad \text{Eq. 5}$$

$$ERR_T(i,j) = \left|P_T(i-1,j) - \hat{P}_T(i-1,j)\right| + \left|P_T(i,j-1) - \hat{P}_T(i,j-1)\right| \quad \text{Eq. 6}$$

$$a(i,j) = \frac{ERR_T(i,j)}{ERR_T(i,j) + ERR_S(i,j)} \quad \text{Eq. 7}$$

$$b(i,j) = \frac{ERR_S(i,j)}{ERR_S(i,j) + ERR_T(i,j)} \quad \text{Eq. 8}$$

2.1.5 Summary

Table 3. Remarks of inter-prediction schemes

Temporal Predictor Methods	Performance	Complexity	Overhead
Motion Search	High	High	High
Zero Motion	Medium	Low	None
Non-motion based	Low	None	None

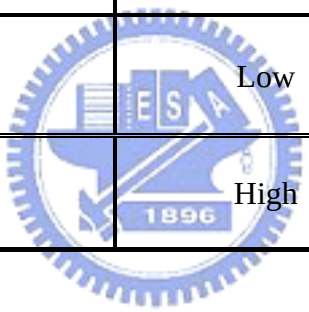
In Table 3, we compare the inter-prediction methods. Considering the trade-off between complexity and performance, zero-motion can capture the slow variation in the video. For fast motion case, motion search will need a great amount of computational load for real-time coding. Side information including motion vector will degrade overall coding efficiency. Thus, based on the observations, we choose the zero-motion for removing the temporal redundancy in our lossless video coding scheme.

In addition to the inter-frame prediction, we review the intra-prediction methods. In Table 4, we illustrate three methods including DPCM, fixed predictor plus context model, and minimized MSE. DPCM uses the simple predictor to remove the redundancy. DPCM is a simple structure without good adaptations for variable media. The fixed predictor plus context

model is the structure defined in JPEG-LS. The fixed predictor is adapted by context model to keep the better prediction accuracy. The last method is “minimized MSE”, which takes MSE to estimate the best predictor. The minimized MSE needs extra overhead to indicate the mode decision. In addition, we need to decide the real range of each mode. Using one mode to represent the whole frame would get the poor local variation. If one frame is split into small blocks, the total amount of overhead increases drastically. Thus to minimize the overhead, we select the fixed predictor plus context model for developing our lossless video coding scheme.

Table 4. Remarks of intra-prediction approaches

Spatial Predictor Methods	Performance	Complexity	Overhead
DPCM	Low	Low	None
Fixed Predictor + Context Model	High	Low	None
Minimize MSE	High	High	High



2.2 Entropy Coder in Lossless Video Coding

2.2.1 Entropy Coders

Entropy is the upper bound estimation of coding efficiency. With the entropy, any event that has more probability provides less info. For lossless compression, entropy coders including arithmetic code and Huffman code have been used in existing coding schemes. In JPEG-LS, Golomb code is a low complexity entropy coder and performs good coding efficiency for residuals of two-sided geometric distribution (TSGD). LOCO-I [2] is the previous prototype of JPEG-LS. LOCO-I and JPEG-LS have almost identical compression performance. LOCO-A [2] is an arithmetic-extension version of LOCO-I. In addition, with the arithmetic coder instead of using Golomb coder, LOCO-A has better performance than LOCO-I.

For a binary representation method, the probability of event A is *P(A)*, then the entropy is $-\log_2 P(A)$ bits. Golomb code can be explained in the run-length coding procedure [9]. For only two possible outcomes in the set, the unfavorable event's probability is q (= 1- p). In general, the probability of run-length n is $p^n q$. If $p^m=1/2$, the entropy will increase 1-bit per run of length n.

$$p^{m+n}q = \frac{1}{2} p^n q \qquad \text{Eq. 9}$$

The method is good for the infinite outcomes in the set. When the statistics of the alphabet set matches the distribution law in Eq. 10. As for Huffman coding, the distribution of the whole events is prior to coding procedure for achieving the upper bound of coding performance.

The accumulated probability for a whole set of events is equal to the unity. With the probability distribution, we can find a suitable set of codewords to fit the optimal coding efficiency. Figure 9 provides an example of probability distribution with m = 2, m = 3 and m = 4 respectively.

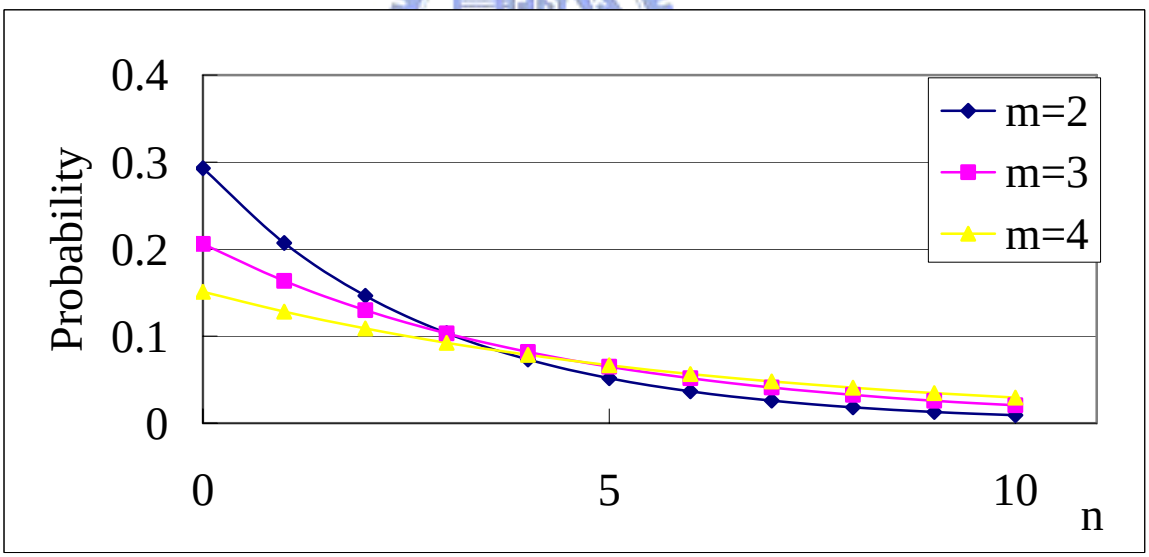


Figure 9. Probability distribution with m = 2, m = 3 and m = 4.

Table 5 gives codewords corresponding to m = 2, 3 and 4 respectively. The parameter m decides the probability distribution decay rate. The special case of Golomb-Rice used in the JPEG-LS is the limitation of parameter m. Due to the low complexity issue, simplification on division-free operation is to balance between complexity and performance. The parameter, m,

in a form of power of 2 is taken in JPEG-LS coding procedure. For example, if m = 2^k = 4 with k = 2, the codeword of n = 9 is divided by 4. The quotient is 2 and the remainder is 1. The quotient represents the number of leading one and the remainder is expressed by k (=2) bits. In the middle of the leading one and remainder a single zero is inserted. According to the principle mentioned above, the codeword of n = 9 is 11001(11+0+01). Therefore, m can be easy for hardware implementation with the restriction of power of 2. With only shifting operations, we can produce the quotient and remainder and generate the codeword.

Table 5. Codeword list of Golomb Code with m = 2, m=3 and m = 4.

	m=2		m=3		m=4	
n	P(n)	Codeword	P(n)	Codeword	P(n)	Codeword
0	0.293	00	0.206	00	0.151	000
1	0.207151	01	0.163564	010	0.128199	001
2	0.146456	100	0.12987	011	0.108841	010
3	0.103544	101	0.103117	100	0.092406	011
4	0.073206	1100	0.081875	1010	0.078453	1000
5	0.051756	1101	0.065008	1011	0.066606	1001
6	0.036592	11100	0.051617	1100	0.056549	1010
7	0.02587	11101	0.040984	11010	0.04801	1011
8	0.01829	111100	0.032541	11011	0.04076	11000
9	0.012931	111101	0.025838	11100	0.034606	11001
10	0.009142	1111100	0.020515	111010	0.02938	11010

2.2.2 Summary

For high compression ratio, an arithmetic coder is a better choice than a Golomb coder. For hardware implementation issue, the Golomb coder that has simple structure is a good candidate. Only shifters and adders are needed to implement the Golomb coder. If the prediction error distribution matches with TSGD, the Golomb coder can have higher

performance.

Table 6. Remarks of various entropy coders.

	Performance	Complexity	Adaptation
VLC (Huffman)	Medium	Low	Low
Golomb Code	Medium	Low	High
Arithmetic Code	High	High	High

2.3 Lossless Data Compression Tools

Lossless data compression algorithms consisting of WinRAR and WinZip are the common and popular. The compression algorithms are designed for general data compression under the consideration of flexibility. The algorithms mix various algorithms and adopt the best algorithm adaptively to achieve the better performance over various sources. The adaptation of data compression is based on analysis prior to coding. For analysis, the buffer requirement is larger when the source data is huge. Thus, for low complexity issue, WinRAR and WinZip are not suitable. In addition, the technical details of the commercial software WinRAR and WinZip are not available. In our simulations, the two tools are used for the performance comparisons with proposed line-based adaptive lossless video compression (LALVC) algorithm.

Chapter 3

Line-based Adaptive Lossless Video Compression

For interactive multimedia applications that demand low complexity and low latency, we present a novel line-based adaptive lossless video compression (LALVC) algorithm. In LALVC, the architecture, syntax and coding methods are explained in detail. In addition, we reveal the hardware architecture of LALVC on an ARM based platform. In addition, the hardware implementation issues are addressed and resolved.

3.1 Motivation

We propose an algorithm with low complexity for implementation on FPGA platform or ASIC design. The features of the proposed algorithm include one-pass, transform-free, motion-free, and simple structure of entropy coder. Since video interaction on high-resolution screens or display devices needs huge bandwidth for exchanging raw data. To reduce the transmission cost and provide video without losing data accuracy, lossless video compression is required.

3.2 Main Architecture

Figure 10 shows the application example of LALVC. The screen signal is output from the VGA card and transformed into LVDS-type signal for wire-transmission. Our algorithm is implemented into the PCB with FPGA Chip. Digital signal is fed into our design for compression. Compressed signal is transformed by the LVDS-type signal for transmission. After the receiver gets the signal from the wire-line, de-LVDS transform is applied on the signal. The Decoder PCB decodes the signal on the digital-type signal to reconstruct the screen.

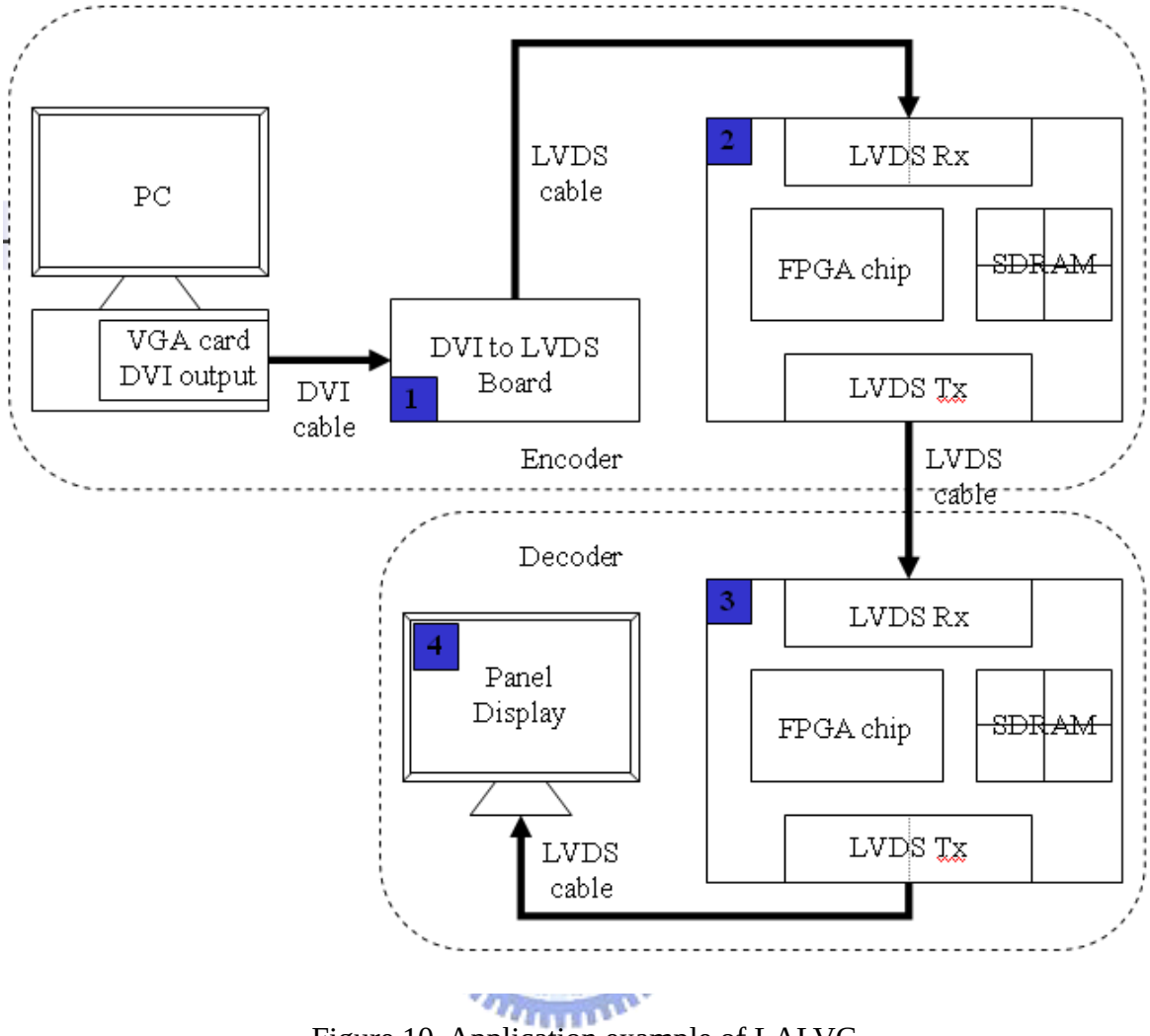


Figure 10. Application example of LALVC

3.3 Algorithm Development

3.3.1 Overview of Algorithm

We propose a line-based adaptive lossless video compression (LALVC) algorithm for interactive multimedia applications that demand low complexity and low latency. Considering low complexity and low delay, LALVC adopts a simple and efficient architecture that adopts one-pass, raster-scan, transform-free coding process and a simple predictor.

(a). One-pass:

Multiple-pass encoding can enhance the coding performance by analyzing the source first and use side information to set the coding flow. However, the multiple-pass encoding needs more buffer for pre-analysis and will increase the latency of the bitstream output.

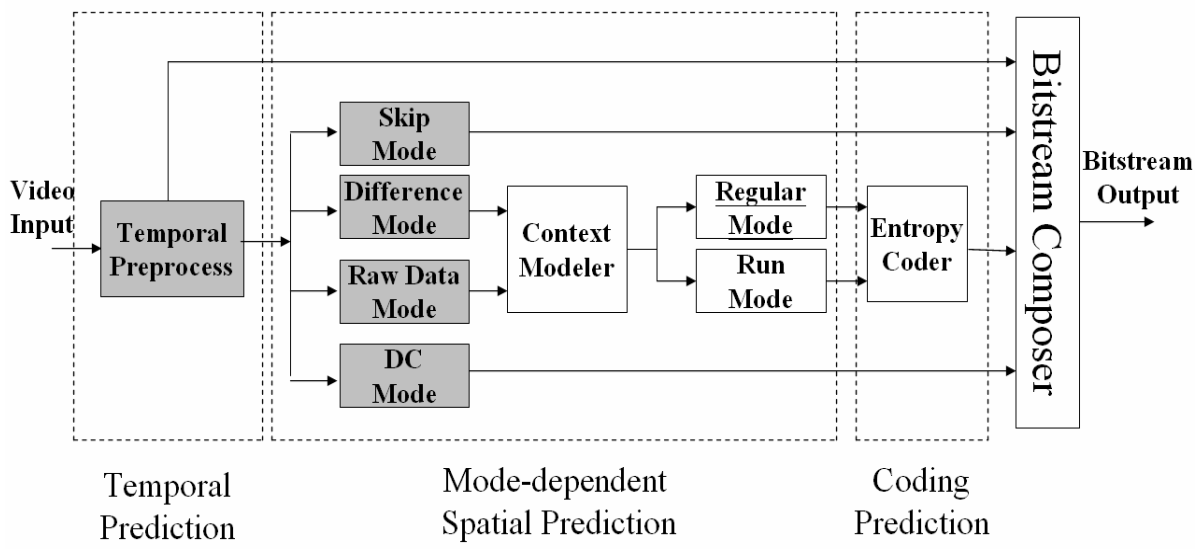


Figure 11. The block diagram of the LALVC encoder.

(b). Transform-free:

In lossless image coding, data accuracy is preserved. If a transformation method including DCT or Wavelet is taken, the coefficients are represented in floating point when non-integer transformation is applied. With transform coefficients, we can keep best subjective quality based on the removal of high frequency components, which causes the data loss. For the lossless video compression, the transformation is not used.

(c). Simple predictor:

Complex statistics is an obstacle for low-complexity requirement. The main goal of our algorithm is to get the high coding efficiency with a simple predictor, i.e. multiplier and divider are avoided.

For low latency, zero-motion prediction and one-frame buffer are used to reduce temporal redundancy. In addition, to maximize the coding efficiency for both natural and computer-generated video sequences, LALVC adaptively selects the best coding mode for each handling line of every frame. For each line, Golomb code in entropy coding can enhance coding efficiency with less computation load and is easy for hardware implementation. The experimental results show that temporal preprocessing and line-based mode decision can increase compression ratio with properly increased complexity as compared to that of JPEG-LS. LALVC contains three major parts. In Figure 11, the source video is first fed into a

preprocessing unit for temporal redundancy removal. The output of the preprocessing process provides the best mode for coding each line. The mode information is then passed to the second part, called as a mode-dependent spatial predictor. Based on a simple spatial predictor, inter-pixel redundancy is removed and the residuals are encoded with an entropy coder using Golomb Coder to remove coding redundancy. The decoder uses inverse operations at the encoder to reconstruct sequence line by line and frame by frame. The blocks with gray color in Figure 11 are novel. The remaining blocks including fixed predictor, context model, and Golomb codes are selected from the existing JPEG-LS standard to fit our requirements.

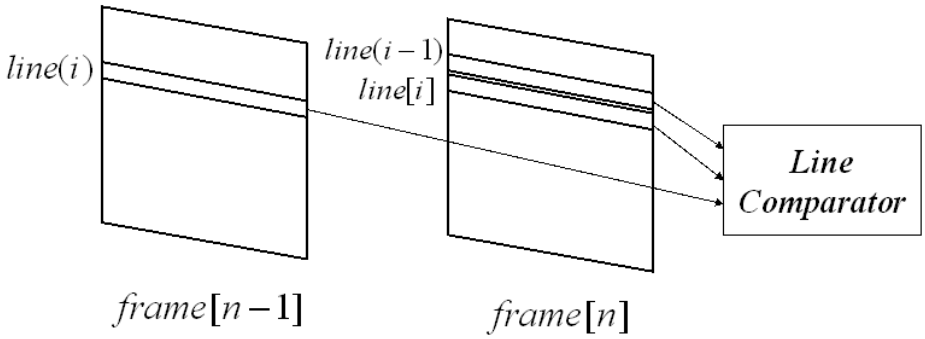


Figure 12. Line-based zero-motion comparator

3.3.2 Prediction Method

Three predictors used in temporal, spatial and coding relations are introduced individually. For low-complexity requirement, we develop the algorithm with simple operations to alleviate computation cost by multiplication or division.

3.3.2.1 Temporal Prediction (Inter-Frame Process)

To avoid high computation load, motion search is not considered in our development of LALVC. Zero-motion DPCM is the simplest method. To emulate the performance of DPCM, we compare the performance of the video coding with or without zero-motion prediction. Table 7 shows the results of DPCM without zero-motion prediction. Both Table 8 and Table 9 show the enforced zero-motion prediction and the difference is the subsequent coding process

Table 7. Original sequence test without DPCM process

	WinZip	WinRAR	JBIG1	JPEG2000	JPEG-LS	CALIC
Akiyo_cif	1.670916	2.649852	1.909968	2.82182592	2.995522	3.170546
Silent_cif	1.399279	1.859162	1.555788	2.12750716	2.198372	2.194892
Foreman_cif	1.53747	2.085842	1.753247	2.28110599	2.414634	2.506403
Bus_cif	1.368598	1.602092	1.423095	1.99786088	1.988431	2.022003

Table 8. DPCM process (Negative value plane + positive value plane)

V1	WinZip	WinRAR	JBIG1	JPEG2000	JPEG-LS	CALIC
Akiyo_cif	4.184175	4.807215	4.935194	4.68809193	5.791167	5.146251
Silent_cif	2.120254	2.337154	2.649184	2.22588623	2.716298	2.648214
Foreman_cif	1.559567	1.554926	1.833107	1.55026621	1.854164	1.860442
Bus_cif	1.124387	1.213907	1.23257	1.08378339	1.335618	1.313306

of the prediction residuals. The source images are gray-level pictures. Residual is derived by differentiating each pair of pixels. The level range becomes double (-255~+255), which needs 9-bit depth to represent each residual. In Table 8, we classify the residuals into two different frames. One is positive plane and the other is negative part. Table 9 divides the residuals into sign plane (1-bit plane) and the other is the plane consisting of absolute values.

The gains based on zero-motion prediction can be evaluated by comparing Table 7 with Table 8. The sequences Akiyo_cif and Silent_cif could get the better compression when zero-motion is applied. The coding performance of the sequences Foreman_cif and Bus_cif are not favorable. Thus, we need a hybrid method of context-aware scheme to automatically apply zero-motion to the suitable sequences. Considering sequence properties, Akiyo_cif and Silent_cif contain slow motion objects and Forman_cif and Bus_cif have high motion objects. Thus, to achieve the highest performance, a good mode decision is demanded to make a hybrid predictor algorithm for various video sequences.

In our algorithm, LAVLC encodes video sequences in a manner of pixel by pixel and then line by line in raster-scan order. The line-based preprocessing algorithm employs some predictions to remove redundancy between temporally neighboring frames. As shown in Figure 12, the temporal prediction with zero motion vector, which has the lowest complexity,

Table 9. DPCM process (Absolute-value plane + Sign plane)

V2	WinZip	WinRAR	JBIG1	JPEG2000	JPEG-LS	CALIC
Akiyo_cif	4.106447	4.760964	5.201401	4.72215597	5.201765	5.379369
Silent_cif	2.136152	2.478691	2.667505	2.30054222	2.619741	2.782672
Foreman_cif	1.667592	1.921768	2.001752	1.81020296	2.013832	2.155733
Bus_cif	1.270787	1.374832	1.468044	1.48336829	1.538868	1.639251

is applied to co-located lines of successive frames. In Figure 12, the *i*-th line and the (*i-1)*-th line of the frame I_n and *i*-th line of the frame I_{n-1} are imported into the line comparator to calculate the difference of the two lines for advanced mode decision. The mode decision is based on two metrics including SAD (sum of absolute difference) and SAGD (sum of absolute gradient of difference).

$$SAD_{inter} = \sum_{j=1}^{N} \left| I_n(i,j) - I_{n-1}(i,j) \right| \quad \text{Eq. 11}$$

$$SAGD_{intra} = \sum_{j=2}^{N} \left| I_{intra}^{D}(i,j) - I_{intra}^{D}(i,j-1) \right|, \quad \text{Eq. 12}$$

$$SAGD_{inter} = \sum_{j=2}^{N} \left| I_{inter}^{D}(i,j) - I_{inter}^{D}(i,j-1) \right|, \quad \text{Eq. 13}$$

Where the difference signals $I_{intra}^{D}(i,j)$ for the *i*-th row and *j*-th pixels within the *n*-th frame are derived by

$$I_{intra}^{D}(i,j) = I_n(i,j) - I_n(i-1,j);\ i,j = 1 \ldots N \quad \text{Eq. 14}$$

The other difference signals $I_{inter}^{D}(i,j)$ is calculated by

$$I_{inter}^{D}(i,j) = I_n(i,j) - I_{n-1}(i,j);\ i,j = 1 \ldots N \quad \text{Eq. 15}$$

With SAD_{inter}, $SAGD_{intra}$ and $SAGD_{inter}$, Figure 17 shows the proposed mode decision algorithm. With zero-motion prediction, line-based preprocessing is shown in Figure 13.

We use a near exhaustive approach to find the upper bound of coding performance under the specified structure and to decide the mode tags. According to the upper bound, we analyze the absolute gradient of difference (AGD) in inter and intra domains to show the principle of optimal mode decision.

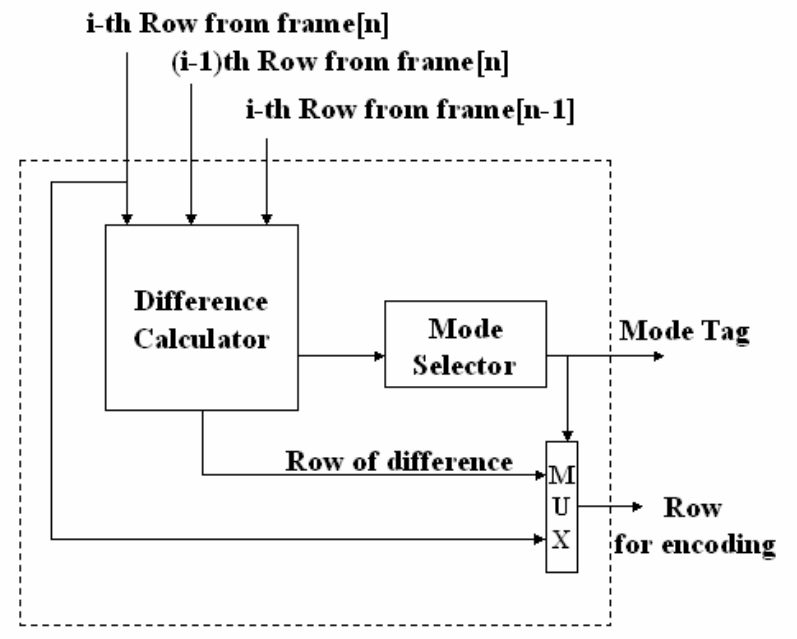
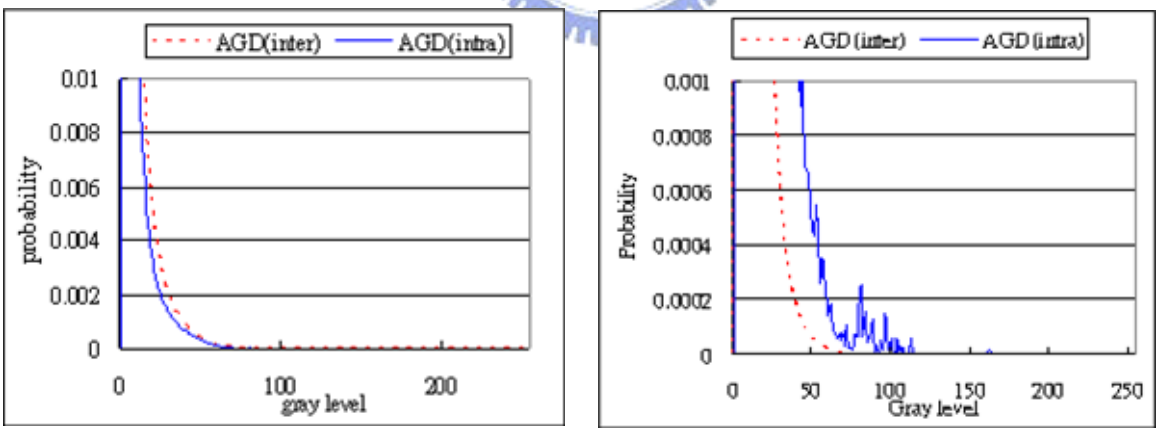


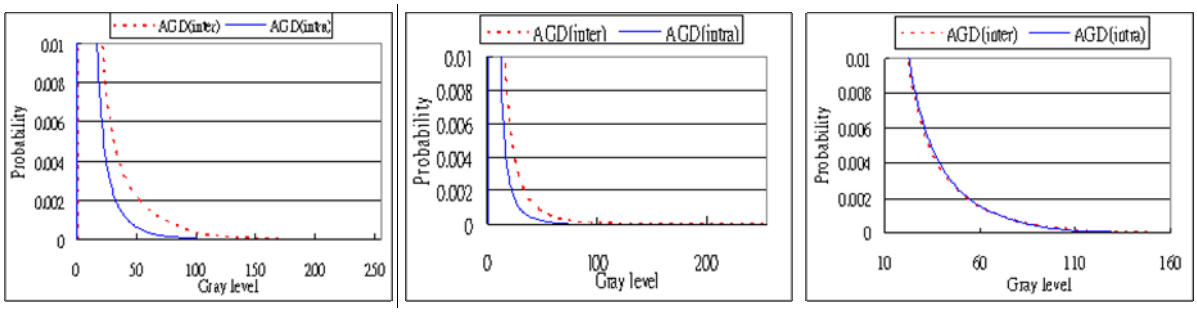
Figure 13. Line-based preprocessing.

Figure 14(a) and Figure 14(b) show the AGD(inter) and AGD(intra) histogram probability distribution. When raw data mode is preferred, the AGD(intra) will be centralized at lower level area. The sequence used for analysis is Foreman, under the near-upper bound case, raw data mode probability is 72.7% and the difference mode is 27.2%.



(a) Raw data mode (72.7%) (b) Difference data mode (27.2%)

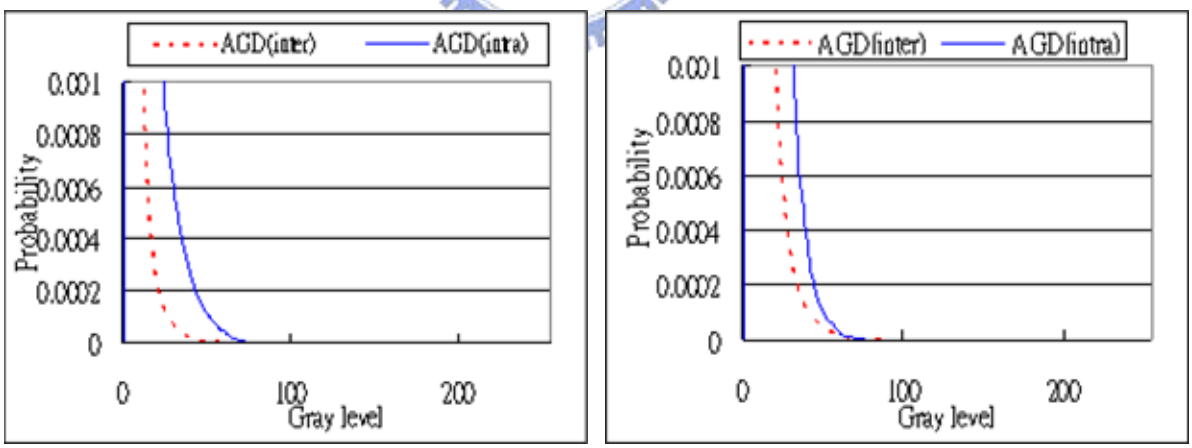
Figure 14. Probability distribution of AGD(inter) and AGD(intra) for Foreman sequence in CIF resolution.



(a). Bus_Raw_Mode(98.1%) (b). Football_Raw_Mode(86.9%) (c). Mobile_Raw_Mode(85.3%)

Figure 15. Probability distribution of sequences with major mode of Raw Data Mode

In our simulation, we take 6 sequences. Except the Foreman, we group the other 5 sequences into 2 types. The first group is Raw_Mode dominated sequences. In Figure 15, Bus, Football, and Mobile belong to this type. AGD(inter) is distributed at higher gray level. Mobile is not obvious. The second type of sequences is Diff_Mode dominated. In Figure 16, AGD(intra) is over AGD(inter) for Akiyo and Silent in probability distribution. According to the statistics, we can conclude a simple rule for mode decision to take the suitable mode for coding the line.



(a). Akiyo_Raw_Mode(99.6%) (b). Silent_Diff_Mode(88.7%)

Figure 16. Probability distribution of sequences with major mode of Diff_Mode

With the preprocessing, we choose the best coding mode for each pixel every line. With the best coding mode, each line of the current frame can be encoded most efficiently. To further improve the coding efficiency based on close relationship of spatially successive lines,

a line-based intra-frame prediction method that consists of adaptive reference line and simplified context modeling is introduced.

3.3.2.2 Four Line-based Coding Modes

Four modes including skip mode, difference mode, raw data mode and DC mode are used to categorize the outputs of the line comparator at the encoder. For decoding each line, 2-bit overhead represents the coding mode.

(a). Skip Mode (codeword '00' in the syntax)

The line in current frame is the same as the one in previous frame. The line is skipped by using the side information for decoder to reconstruct the line by directly duplicating it from the previous frame. In Table 10, the previous two columns are compression ratio with and without skip mode. Coding efficiency has great improvement on Akiyo, OutR, OutG, and OutB due to high probability of skip mode cases. If the screen is still for a long time, skip mode would contributes to the best degree.

Table 10. Statistics of skip mode for 9 sequences.

	Intra coding	Intra coding + Skip Mode	Skip Counts	Skip Probability
Akiyo.Y	2.70	3.05	11683	0.135
bus.Y	1.64	1.64	0	0
Football.Y	1.99	1.99	249	0.003
Foreman.Y	1.97	1.97	0	0
Mobile.Y	1.43	1.43	0	0
Silent.Y	1.86	1.86	0	0
Out.R	4.96	16.81	21051	0.731
Out.G	5.10	17.97	21048	0.731
Out.B	5.30	19.45	21109	0.733

(b). Difference Mode (codeword '01' in the syntax)

Table 7, Table 8, and Table 9 show the benefits of the zero-motion prediction for the slow-motion sequence just as Akiyo or Silent. We assign difference mode to turn on the zero motion prediction. The difference mode is applied when the magnitude of $SAGD_{inter}$ is smaller than $SAGD_{intra}$. In this case, we take the residuals of zero motion prediction for coding. With the prediction residual, the sign information costs one extra bit. To save the sign bits, the same pixel value re-mapping approach adopted in JPEG-LS is used. If the image alphabet size is $\alpha (=2^8$ for a gray-level image), remapping the large prediction residual ($-\alpha \le \varepsilon \le \alpha, \varepsilon$ is the current residual) into the small and unsigned one ($0 \le \varepsilon \le \alpha$) can decrease the value range without violating the data correction since the prediction value is known in the decoder

(c). Raw Data Mode (codeword '11' in the syntax)

Table 7, Table 8, and Table 9 show the fast motion sequence would be processed well without turn on the zero motion predictor. As $SAGD_{intra}$ is smaller than $SAGD_{inter}$, spatial coding algorithm instead of zero-motion prediction shall be used. In this mode, only intra-frame prediction is taken on the original pixel.

(d). DC Mode (codeword '10' in the syntax)

DC mode is designed for when brightness or luminance on the screen is changed. The mode will contribute to high performance. When the difference value $SAGD_{inter}$ is zero, $I_{inter}^{D}(i, j)$ is the same for all pixels in the processing line, we can use one value to represent DC offset. With the DC offset, the decoder can perfectly reconstruct the line.

A. Inter-Component Process

For color sequence, color plane can be fed into 3 encoding modules individually. R plane is coding by reference of the R, and G and B planes are processed by the same way. If the context models of various planes are independent, memory requirement is large for realizing the inter-component process. Another choice is coding by the order of R-G-B, which means R plane is referenced by G plane and G plane is referenced by B.

B. Mode-Dependent Spatial Prediction (Intra-Frame Process)

In LALVC, we use a simple template in Eq.16 to predict the pixels of the same frame by a fixed predictor by

$$\hat{x} = \begin{cases} \min(a,b) & if\ c >= \max(a,b) \\ \max(a,b) & if\ c <= \min(a,b) \\ a+b-c & otherwise \end{cases} \qquad \text{Eq.16}$$

With the template, only two lines are buffered for advanced processing including error bias cancellation in context modeling and run mode extension. To simplify the context modeling, we use two gradients (*b*-*c*) and (*c*-*a*) for the error bias cancellation. To avoid performance degradation of inter-line prediction when the coding mode transition occurs, we set the reference as the same property as the current coding line, which means the raw data mode with raw value reference line and the difference mode with difference value reference line. In addition, the distinct content model for different modes can retain the continuity of the template with the coding pixels. The mode adaptation is applied priori to the gradient computation in Eq.16.

C. Mode Decision

Mode decision mainly focuses on choosing the better mode between the raw data mode and the difference mode. In LALVC, mode decision is made in the temporal preprocess, which can be seen as one-pass of the line first to analyze the statistics of the line. The complexity is analyzed in the section of 4.1.3.3 . Each line has 2-bit overhead to conduct the decoder to follow the chosen mode for reconstruction of the current line.

We take the opt_16_line for the near-optimum performance. Under the simulation, the line mode tag is acquired. Do the statistics about image property related to the tags. Mode decision is made between temporal and only intra predictions. The following shows the distance of difference between reference lines from inter frame and intra frame.

$if\ SAD_{inter} = 0$

$Enter\ Skip\ Mode\ and\ Skip\ the\ i\text{-}th\ line\ without\ coding$

$else\ if\ SAGD_{inter} = 0$

$Enter\ DC\ Mode\ and\ process\ the\ i\text{-}th\ line\ by\ one\ pixel\ value$

$else\ if\ SAGD_{inter} < SAGD_{intra}$

$Enter\ Difference\ Mode\ and\ take\ the\ difference\ row\ into\ coding\ process$

$else$

$Enter\ Raw\ Mode\ and\ take\ the\ current\ row\ into\ coding\ process$

Figure 17. Mode decision in the preprocessing.

D. Summary

LALVC follows the basic ideas mentioned in the beginning of the chapter. By means of mode decision, we use the four-mode adaptive algorithm to get a high performance based on temporal and spatial predictors. The functions involved in LALVC maintain the low complexity requirements. To show the hardware realization of LALVC, we give the hardware design architecture in the next chapter.

3.3.3 Error Mapping and Entropy Coder

3.3.3.1 Error Mapping

In the previous sections, the difference mode is explained. The residuals is produced from the temporal preprocess. Residuals are derived with subtraction of two pixel values. For gray-level precision (8-bit), the range of residuals with no remapping is increased to as the range of -255~+255, which means that 9 bits are required to represent each value. To reduce the range for saving bits, an error remapping approach is adopted in LALVC.

For 9-bit value range mapping into 8-bit value range, the information of pre-mapped value can be perfectly preserved. The remapping can be explained by decoding processing. In the decoder, the reconstruction value is decided by both residual and predicted value. The remapping methods can be categorized into 2 types. For value range distribution, one is unsigned 8-bit representation (0~255) and the other is singed for 8-bit representation

(-128~127).

A. Signed mapping

In Figure 18, the two triangle blocks are remapped. For the upper one, Ex (Residual) ranges from 128 to 255, and Px (Predicted value) ranges from 0 to 127. The coding value $Rx(=Ex+Px)$ can be perfectly reconstructed by Ex and Px based on 1 to 1 mapping without aliasing.

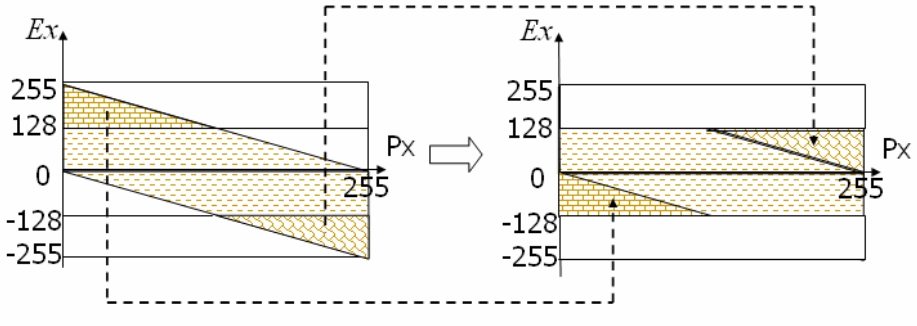


Figure 18. Residual remapping method (signed mapping).

B. Unsigned mapping

In Figure 19, the remapping is done by moving the lower triangle to the upper side. After remapping, Ex (Residual) ranges from 0 to 255 that can be represented by 8-bit. For the two remapping methods, there is minor difference of coding performance in our simulation. In the hardware implementation issue, the unsigned mapping is simpler for architecture design. When 9-bit Ex is available, remapping is finished with the truncation of MSB. More comparators are necessary for implementing the signed remapping. Due to complexity consideration, the unsigned mapping method is preferred.

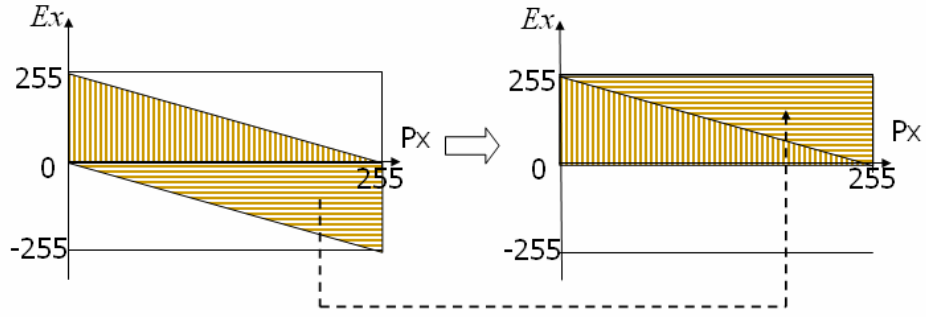


Figure 19. Residual remapping method (unsigned mapping)

3.3.3.2 Entropy Coder

To fit low-complexity requirements, Golomb code is the best choice on a tradeoff between coding efficiency and complexity. Arithmetic code owns higher coding efficiency and higher complexity than Huffman code.

3.3.4 Syntax Organization

We define the syntax for LALVC decoding that supports the four line-based coding modes. In Figure 20, the bits for the frames are separated by the ‘Sync’ field. The field ‘Frame Skip Tag’ can imply the use of the reference as the current one without display refreshment, which can reduce the data volume when the video content is not changed for a long time. For line-level syntax, the ‘Line Mode Tag’ is set in the preprocessing. In addition, the field ‘DC

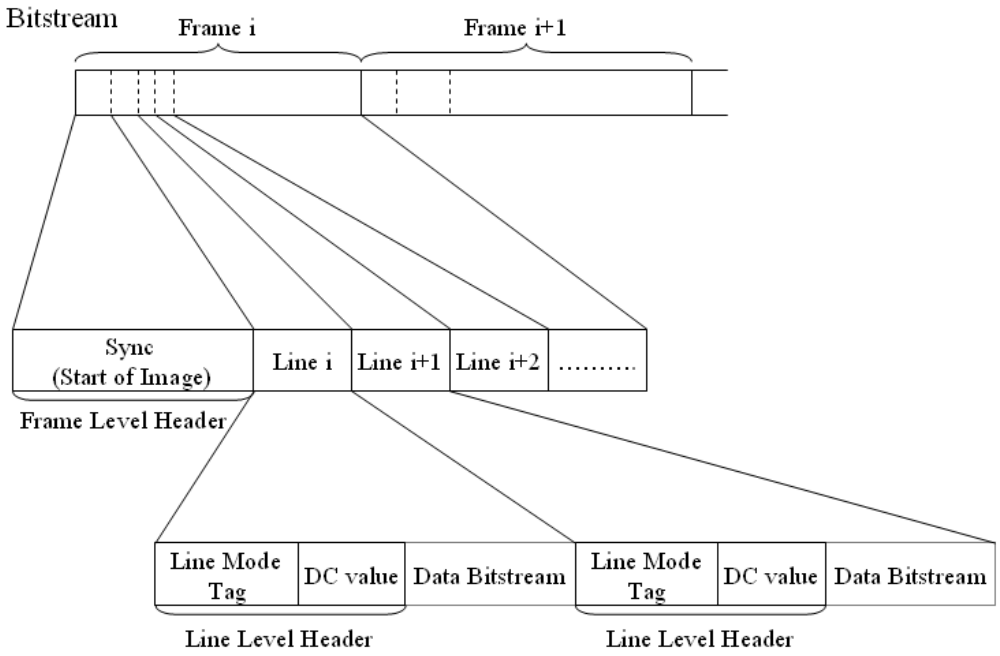


Figure 20. Syntax of CAVLC file format.

value’ can reduce coding overhead when the brightness or color saturation variation of whole image pixels is adjusted. With the selected coding mode, the prediction residuals are generated and then passed through the entropy coder to form the data bitstream. With the syntax, the LALVC decoding process reconstructs the sequence.

3.3.5 Algorithm Simplification

Context Model used in JPEG-LS takes advantage of 3 gradients for deriving the entry. The gradient is computed by *a*-*c*, *b*-*c*, *d*-*b*, (in Figure 21, *a*, *b*, *c*, and *d* are the neighboring pixels around current pixel *x*). Each gradient ranges from -255 ~ 255. If no quantization of gradient is applied, the memory requirement is 511*511*511 states. After quantization (Figure 7), the threshold is set at 3, 7 and 21 (corresponding to T1, T2, and T3). Each gradient is mapping to 9 level states (-4 ~ +4). Total amount of states can be reduced into only 9*9*9 = 729. Merging the states based on the context symmetric property reduces the total number of states into 50%.

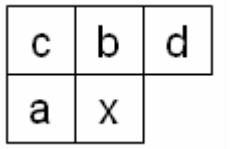


Figure 21. The template for context model

Table 11 lists the number of states for the context models based on 2 or 3 gradients. “2 gradients” means that only (*c*-*a*) and (*b*-*c*) decide the entry point of the states.

Table 11. The number of states to support the context model

	No grouping		Grouping(Quantization)	
Num of gradients	No merging	Merging with symmetric	No merging	Merging with symmetric
3	133432831	66716416	729	365
2	261121	130561	81	41

Four arrays of *N*[], *A*[], *B*[] and *C*[] are the elements of the Context Modeler. For hardware implementation, the state size must be decided. The relation between the four factors can be defined by

$$A/N = C + B/N \qquad \text{Eq. 17}$$

Table 12. The memory requirement for elements in the context model

Array Name	Value Range	Storage Requirement	Actual Memory Size
N[]	0 ~ 64	7-bits	8-bits
A[]	0 ~ 64*255	15-bits	16-bits
B[]	-63 ~ 0	7-bits	8-bits
C[]	0 ~ 255	8-bits	8-bits

In JPEG-LS, reset scheme is enabled when *N*[] is 64, which sets the range of *A*[] in a reasonable region. If arrays of the context model are stored in the SRAM, resetting of each frame is not good for implementation. The simple method is to turn on the reset scheme when initialization process and the context model is used for the successive frames without any reset schemes.

There is a trade-off between coding efficiency and memory requirement.

Table 12 shows the memory size for each element of array. Each state needs 8+16+8+8 = 40 bits, which equals to 8 bytes. For the case of 3 gradients, 365 states are included with grouping and merging. Total memory size is 2920bytes (365*8). For the case of 2 gradients, 41 states are included and total memory size is 328 bytes (41*8). The case of “2 gradients” would need less memory requirement. Thus, with less memory usage, prediction accuracy is decreased for the nature sequences, which is consistent to the observations in Chapter 4.

3.4 Case Study: Hardware Architecture

3.4.1 Temporal preprocess implementation

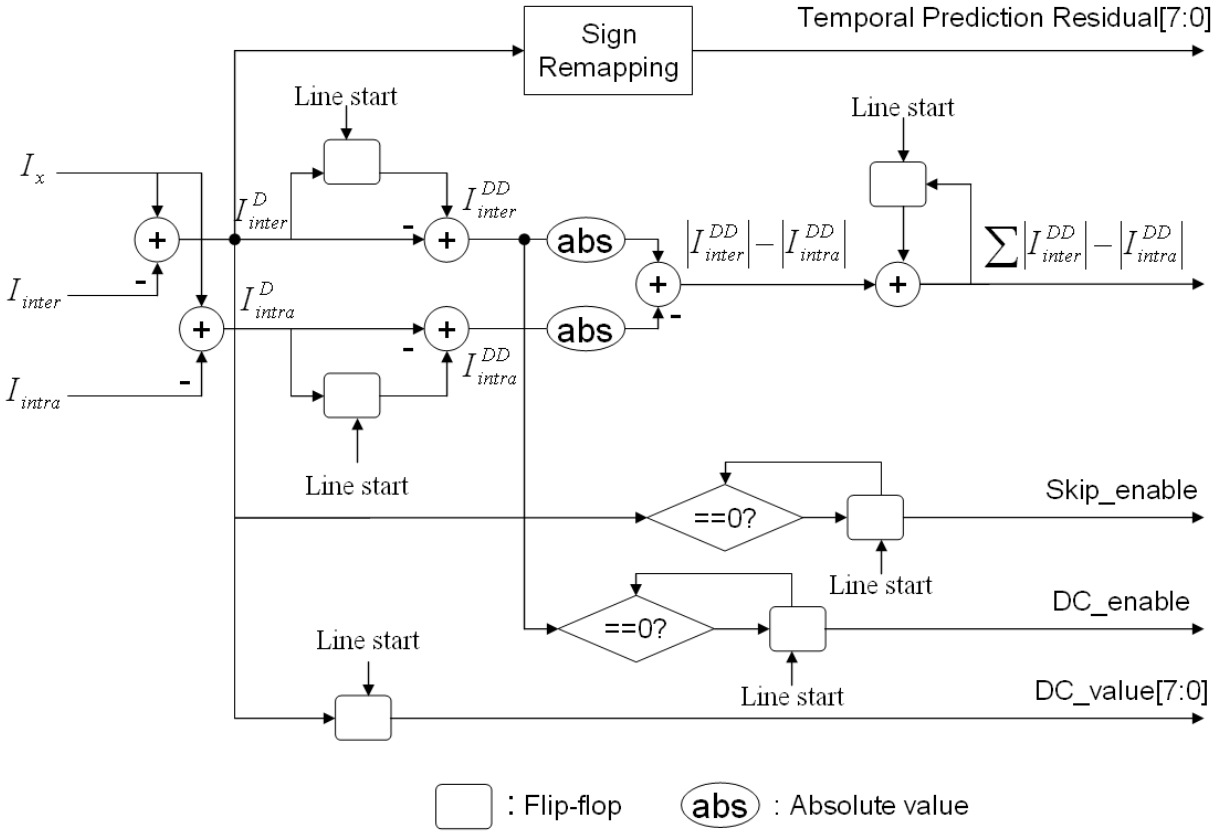


Figure 22. Temporal preprocess architecture

For the discussion in mode decision method (Figure 17), the equations required are listed in Table 13.

Table 13. Complexity comparisons of various metrics used in mode decision

Equation	Num of Adders each line
SAD(inter)	N+N-1= 2N-1
SAGD(intra)	N+(N-1)+(N-2) = 3N-3
SAGD(inter)	N+(N-1)+(N-2) = 3N-3

(SAD_{inter} and $SAGD_{inter}$ share *N* adders each line)

In Table 13, $SAD_{inter} = \sum_{j=1}^{N} I_{n,i}(j) - I_{n-1,i}(j)$ and $SAGD_{inter} = \sum_{j=2}^{N} \left| I_{inter}^{D}(i,j) - I_{inter}^{D}(i,j-1) \right|$ can share *N* adders. To sum up the overall adders, (8N-7) adders are required to execute the mode decision. Furthermore, we design hardware architecture for the same function in Figure 22. It reveals that only 6 adders are enough for one pixel consumption.

3.4.2 Parameter K generator

As to the implementation issue of real-time encoding, throughput of the encoder should achieve 1pixel/cycle. Under the consideration, the loop implemented in the software simulation should be unrolled for hardware requirement. For the parameter, k, is decided by the codes:

```
for (k=0; N[Q] < A[Q]; N[Q]<<=1, k++)
```

If the hardware implements the loop by feedback scheme, there will variable delay time for pixel processing, which will violate the real-time requirement. Thus, the worse case scenario shall be considered. In addition, we use parallel structure to implement the loop, which indicates to increase area cost to get the high execution speed. The implementation method is illustrated in Figure 23.

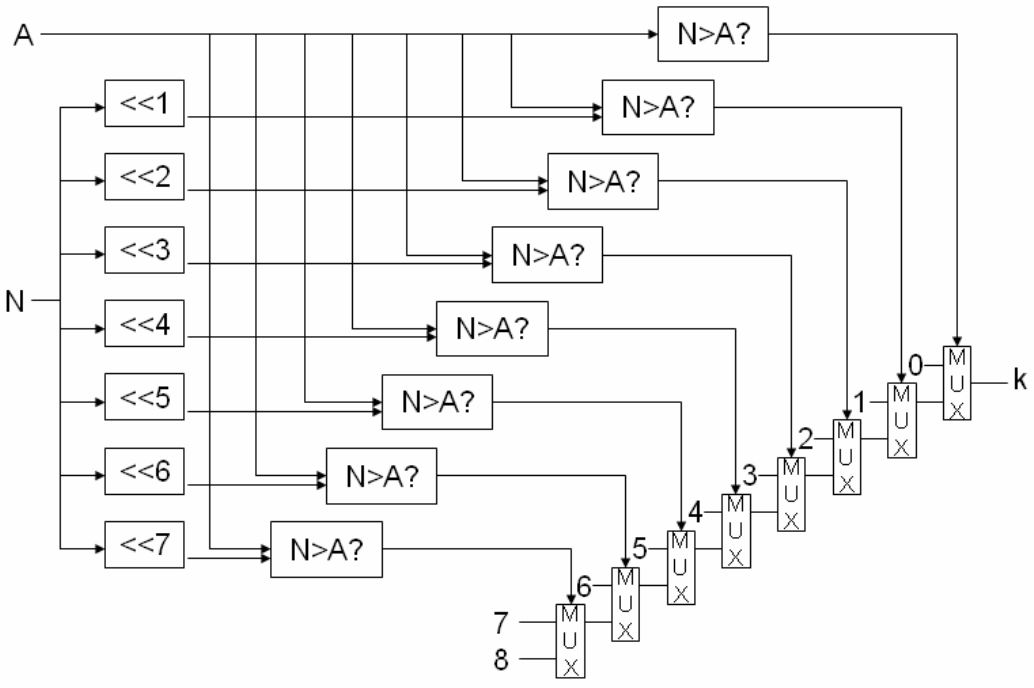


Figure 23. Block diagram of parallel structure for parameter K generator

3.4.3 Context model simplification

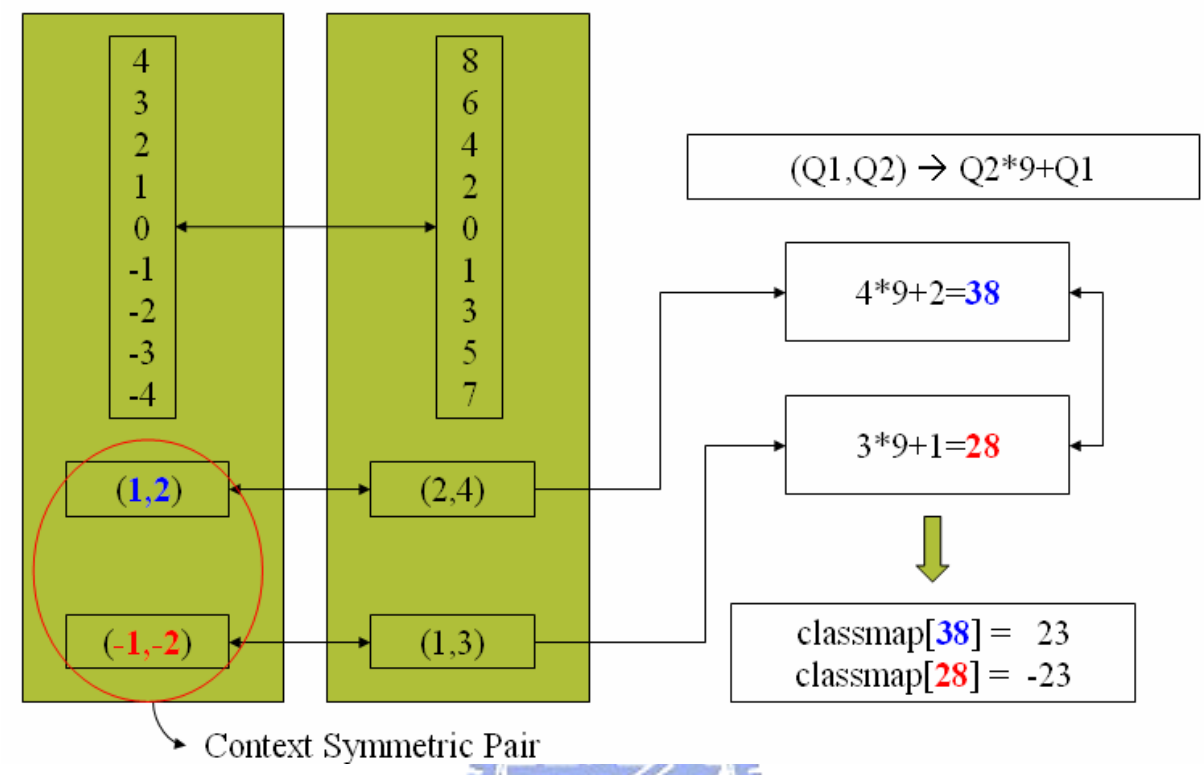


Figure 24. Context merging by symmetric property

In JPEG-LS, the states of context model can be reduced into half based on the property of symmetric. The symmetric implementation in software is illustrated in Figure 24. The method needs two stage of table look up (LUT). In Figure 25, the first LUT is quantization mapping. G1 and G2 are quantized into 9 levels (4 ~ -4). The second LUT is for symmetric merging mapping. The first LUT is mapping 512 to 81 and the second LUT is mapping 81 to 41, which is good for software implementation due to fast execution in LUT. For hardware implementation, we can use a simple combinational circuit to calculate the final entry address. The two LUTs can match with the memory requirements.

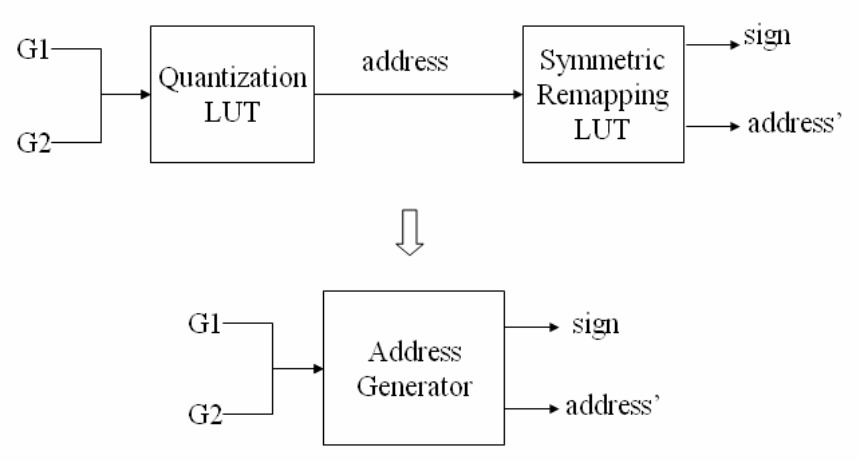


Figure 25. Context model entry address

3.4.4 Summary

In hardware design considerations, we give the architecture of temporal prediction module for mode decision and temporal residuals production. Memory requirement is reduced by the modified module implementation. Overall system could get the simple structure and efficient circuit area, which is beneficial for real-time system implementation.

Chapter 4

Experimental Results

4.1 Performance & Results

4.1.1 Optimized Performance Evaluation

The mode decision of LALVC plays the important role in choosing better case between the 2 modes, the difference mode and the raw data mode. When the Diff mode is applied, the zero-motion residuals are fed into the line-based encoder.

To evaluate the best performance of the zero-motion architecture, a near optimum case is observed. We encode each line in the difference mode and the raw data mode first, and then apply the better case for real encoding of the current line. Based on the context model statistics property, the method called as Opt_1_line could not derive the near optimum performance. The prior mode decision will alter the statistics property, which changes the coding states of the next lines. For some sequences, even forced zero-motion method would get the better compression ratio.

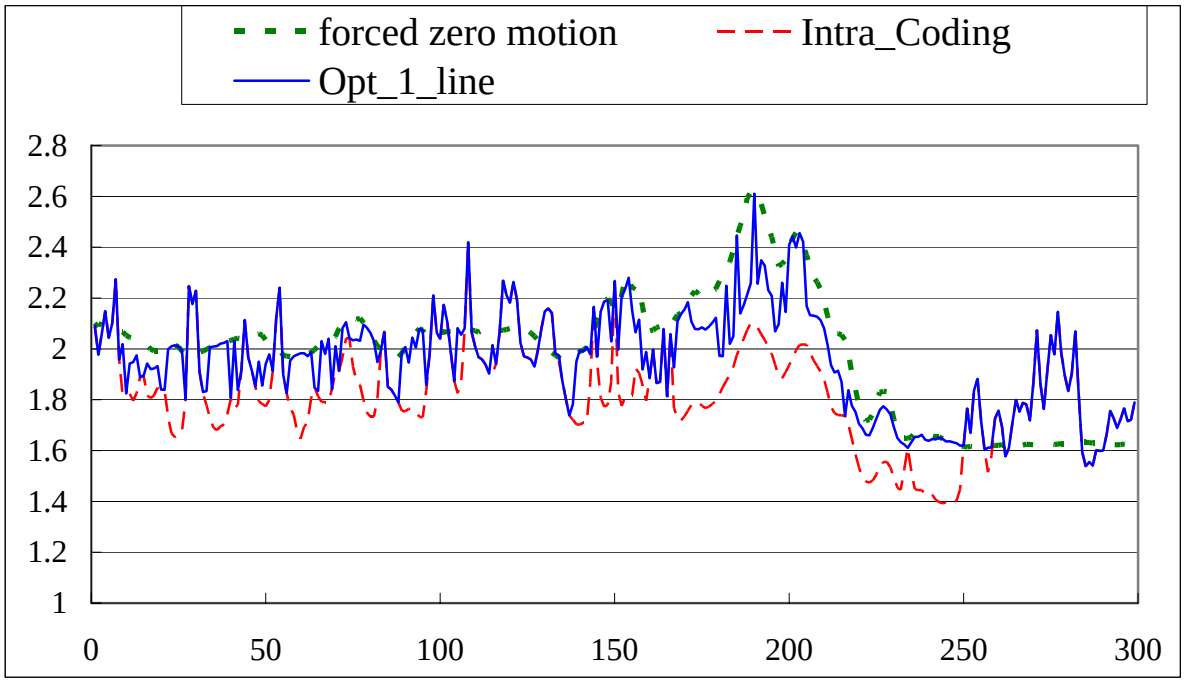


Figure 26. Compression ratios for each Frame of the sequence “Foreman”

In Figure 26, the method “Opt_1_line” represents that Foreman sequence is encoded by three times each line. Tow iterations of pre-encoding choose the better one between the difference and raw data modes. Figure 26 shows the compression ratio for each frame. Opt_1_line can not keep the highest performance in comparison with intra-frame coding and enforced zero-motion prediction coding. Figure 27 show the compression of the 86th frame to observe the performance of line-based compression.

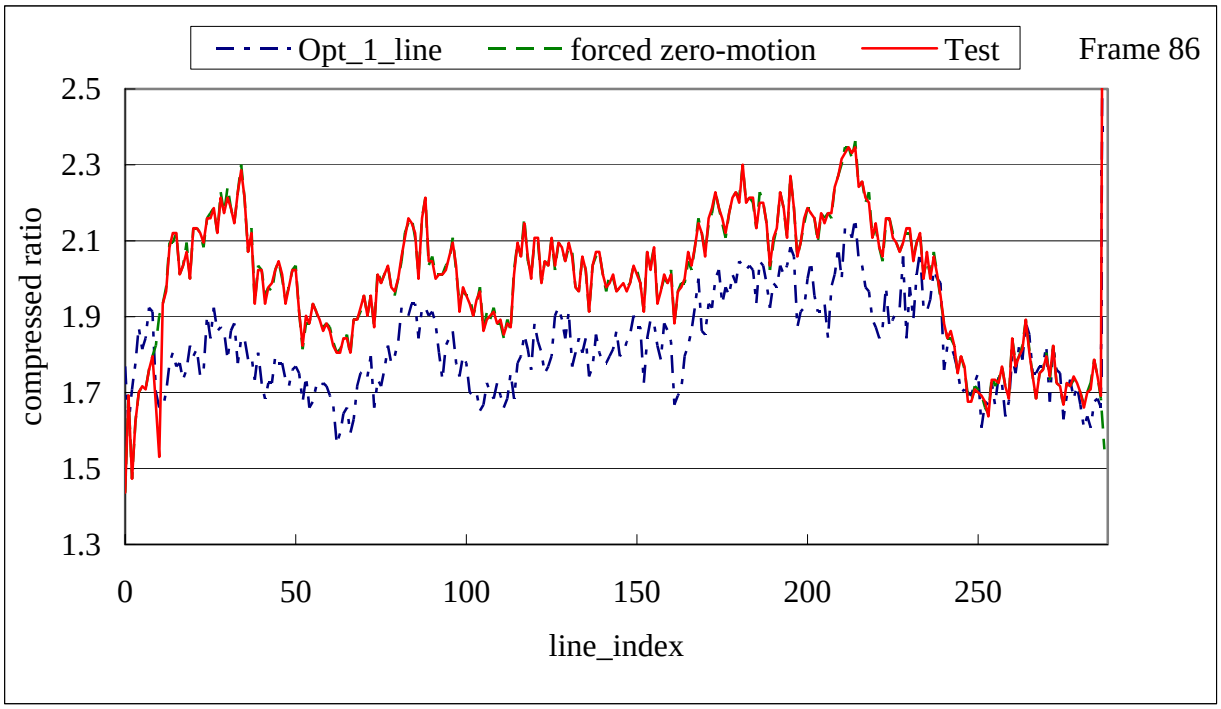


Figure 27. Line compression condition detail in 86th frame of “Foreman”

In Figure 27, Opt_1_line performs better than forced-zero-motion in the beginning. The curve shows a decreasing ratio for Opt_1_line. To test the Opt_1_line algorithm, we set the previous mode selection pretended as the enforce-zero-motion, latter part are operated by Opt_1_line. The curve is represented by the “Test” in Figure 27. Figure 27 shows the Opt_1_line could not find the global best mode selection. The following shows the extension version of Opt_1_line to find the near-upper bound performance under the architecture for low-complexity requirement

Observe extending windows of selection in line mode. Opt_1_line introduced in the previous section could not be aware of the optimum case of the image. Context model

statistics are increased during coding. Under the structure of LALVC, context is empty at the beginning. Each decision would change the overall coding performance. Opt_1_line could be viewed as one-line range window for pre-coding test. Now, we extend the window to more than one line. When the window size is extended to the whole frame, the performance is the best. For encoding the CIF sequence (352*288), the number of lines is 288. For practical computation, the worse case is that all lines are used for the mode selection. Each line will be encoded by 2^{288} times for the worst case scenario in the frame windows.

To simply the derivation of optimal case, we set the window size to cover a range of 16 lines. Each line is at most encoded by 2^{16} times. In the desktop platform, Pentium 4 2.0 GHz, windows XP, each frame encoding takes 20 minutes, which does not fit the real requirement and some complexity reduction is needed.

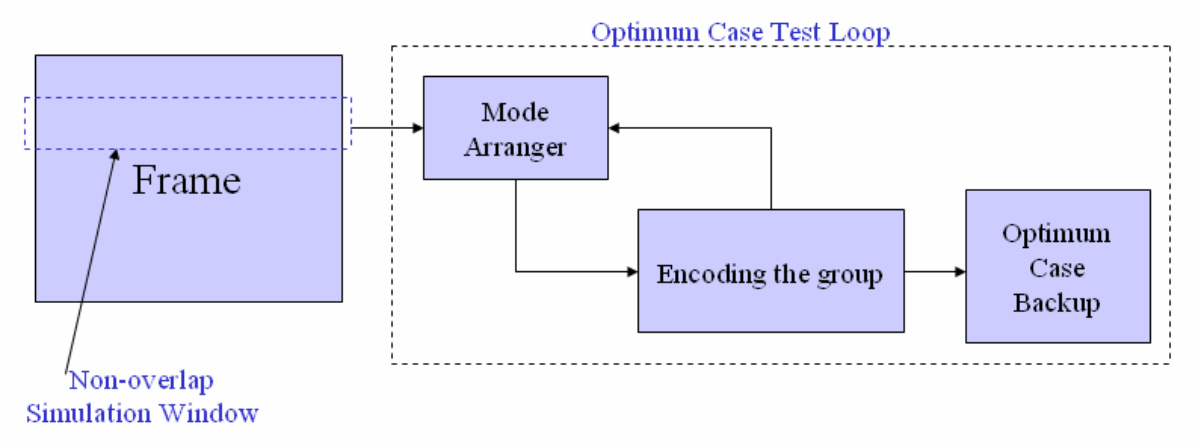


Figure 28. The diagram of optimum performance evaluation.

The following results are derived from the simulations with a window of 16 lines. For slow motion sequence, Akiyo, compression ratio is shown in Table 14. The mothod of Opt_16_line is the same as forced zero-motion method.

Table 14 tells the compression results of three methods. For the overall performance, Opt_16_line could get the best performance on the average. In addition to Table 14, we analyze the frame level conditions to evaluate the overall performance.

(1). Akiyo is a typical slow motion sequence. Figure 29. Compression ratio for each frame of the sequence “Akiyo”. OPT_16 is identical to the enforced zero-motion method. Zero-motion residuals are better for removing coding redundancy.

Table 14. The list of compression ratios

	Intra_coding	Zero_motion	Opt_16_line
Akiyo.Y(300-frames)	2.70	6.76	6.76
Bus.Y(150-frames)	1.63	1.44	1.64
Football.Y(250-frames)	1.99	1.75	2.03
Foreman.Y(300-frames)	1.97	1.84	2.03
Mobile.Y(300-frames)	1.43	1.36	1.43
Silent.Y(300-frames)	1.86	2.61	2.64

(2). Foreman is a sequence with more motions than Akiyo and Silent. In Figure 30, Opt_16_line locates at the top of the three curves, which proves a windows size of 16 lines

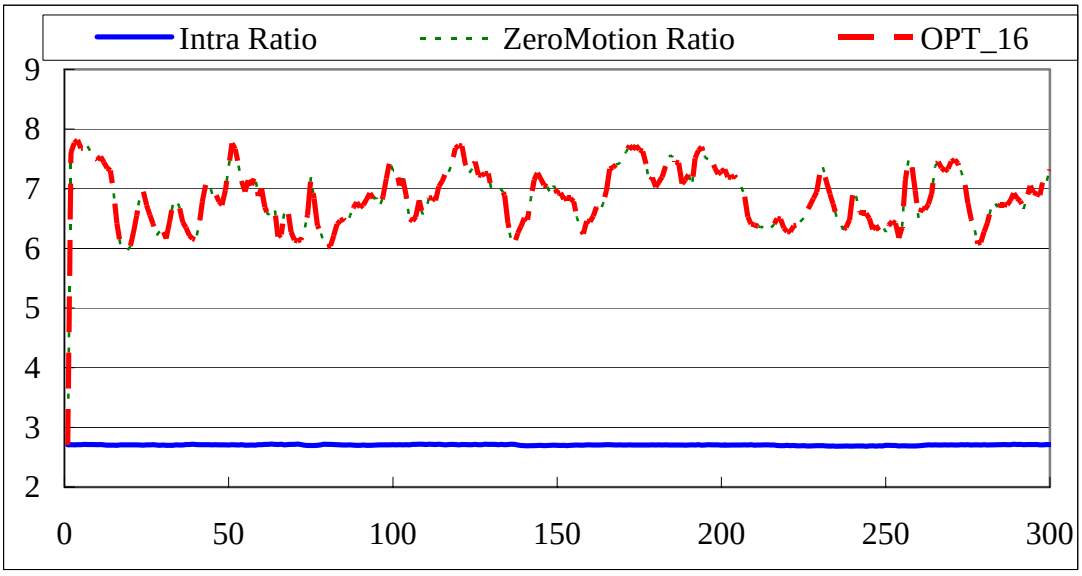


Figure 29. Compression ratio for each frame of the sequence “Akiyo”

can capture the near-optimum case for the Foreman sequence.

(3). Mobile is a sequence with a moving train and zooming in/out. In Figure 31, Opt_16_line does not occupy the top positions of several frames, which means a larger window size is used to capture the top case.

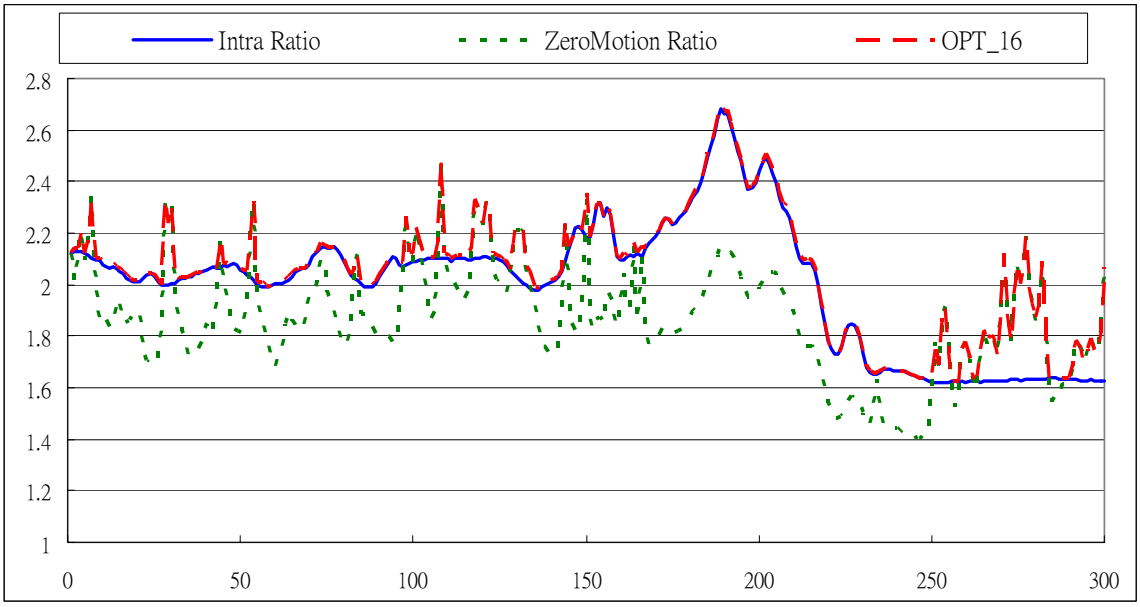


Figure 30. Compression ratio for each frame of the sequence “Foreman”

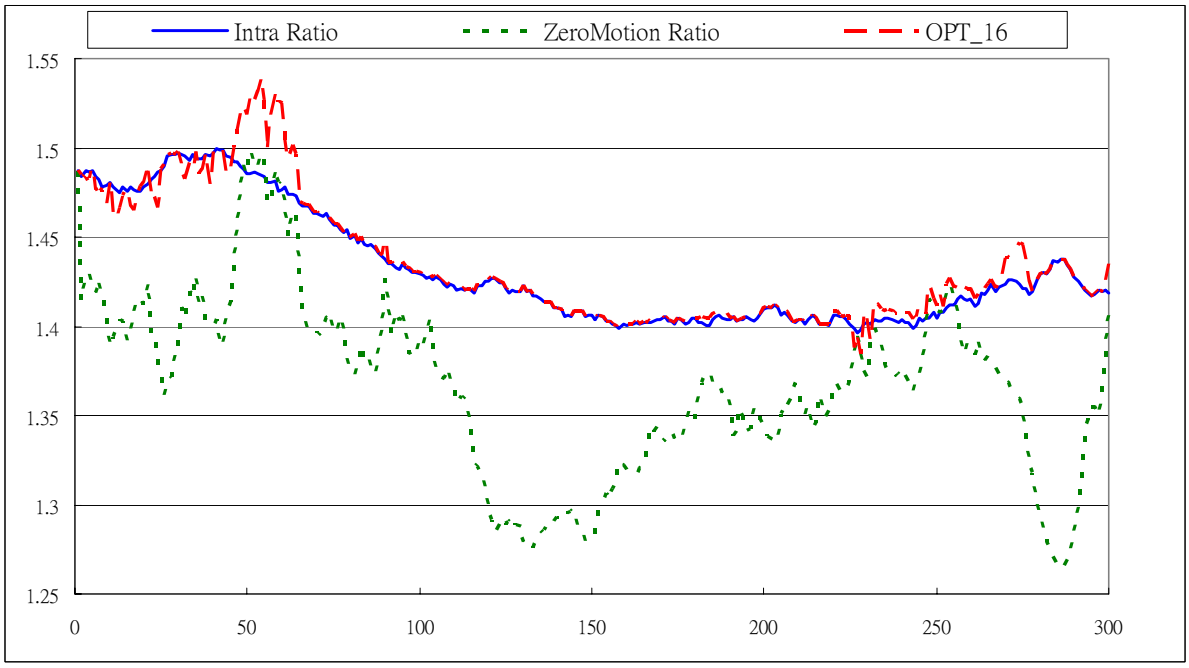


Figure 31. Compression ratio for each frame of the sequence “Mobile”

In frame-level statistics (Figure 26), intra-frame coding is the best one for coding of the 86th frame. In Figure 27, we find that Opt_1_line shows the best gain in the beginning. After 10th line, intra-frame becomes the better one. Ultimately, intra-frame coding gets the better performance. Each pixel coding would alter the context model that affects the coding of the subsequent pixels. With the issue, we set two factors to observe the performance variation.

1. Context Model Interference:

The energy of residuals generated form the difference mode is smaller. The property of the context model for the difference mode should be distinguished from the raw data mode. Hence, the following simulation will show the performance comparison between 1 model and 2-separate models.

2. Context Model Continuity:

Context models are derived based on statistical property of the processing contents. If the model prediction performs very well, the probability of zero residuals will be increased, which indicates the OGDS (one side geometric distribution) would be steeper. Originally, the reset interval is set as the range of one frame. If the successive frames have some stationary property, enlarging the reset interval may be beneficial for the prediction performance. The remaining issue is the decision of the reset moment.

In the first issue of context model interference, we investigate the control factor of 1 or 2 models for context models.

Table 15. 2-Model and 1-Model performance comparison

	Size (Kilo-Bytes)		Compression ratios	
Sequences	Opt_16_line (2-Model)	Opt_16_line (1-Model)	Opt_16_line (2-Model)	Opt_16_line (1-Model)
Akiyo	4393	4393	6.76	6.76
bus	9077	9078	1.64	1.64
Football	12187	12205	2.03	2.03
Foreman	14662	14674	2.03	2.02
Mobile	20729	20753	1.43	1.43
Silent	11251	11257	2.64	2.64

In Table 15, separate context model does not have significant improvement on performance. When the mode decision can choose as the same mode as Opt_16_line, 1-Model approach has identical compression ratios as 2-Model approach. For low complexity, 1-Model instead of 2-Model is used.

Table 16. Performance compression for the factor of state number of Context Model

	Size(Kilo-Bytes)				Compression ratios			
Sequence	2M3G	2M2G	1M3G	1M2G	2M3G	2M2G	1M3G	1M2G
Akiyo	4456	4671	4457	4675	**6.665**	6.358	6.664	6.353
bus	9093	9085	9091	9085	1.633	**1.635**	1.633	**1.635**
Football	12222	12423	12249	12438	**2.025**	1.992	2.021	1.990
Foreman	14918	15098	14939	15152	**1.991**	1.967	1.988	1.960
Mobile	20900	21070	20958	21168	**1.421**	1.410	1.417	1.403
Silent	11330	11740	11330	11748	**2.621**	2.530	**2.621**	2.528
OutR	495	477	492	476	20.000	20.755	20.122	**20.798**
OutG	460	447	460	447	21.522	**22.148**	21.522	**22.148**
OutB	417	406	416	407	23.741	**24.384**	23.798	24.324
Total	74291	75417	74392	75596	**2.532**	2.494	2.528	2.488

*Bold number represents the best in the row of table

In Table 16, “2M3G” means 2-Models plus 3 gradients applied on the algorithm. Other items mean the same. In Table 16, the best performance is 2M3G. Table 15 shows that 2-Model is proven to have near performance to 1-Model, which has slightly less coding efficiency due to mismatch between the mode decision and Opt_16_line.

The Y component of D1 resolution sequences has 720*480 pixels. The four testing sequences are full of motion. We divide the original sequence into four small sequences by cutting the spatial area. In the high resolution sequence, the performance improvement is not significant. When we divide the sequence into sub-resolution sequences width reduced width to evaluate the effect of the line length. In Figure 15, performance difference for the different sub-resolution process is minor. Cutting D1 into 360*480 resolution sequences could get minor improvement by 0.006 ratios. Side information of tag increases with more segments used. In the resolution of 240*480, D1 sequence is cut into 3 small ones. The average performance may not be better than the resolution of 360*480.

Table 17. Simulation of D1 sequence

(720*480)	Ratio				
Sequence	WinRAR	WinZip	LALVC(1M3G)	Intra	ZeroMotion
crew(300)	2.047	1.456	2.347	2.348	1.980
harbour(300)	1.494	1.162	1.991	1.991	1.869
night(230)	1.858	1.396	2.220	2.194	2.023
pour_water(1017)	2.279	1.583	2.626	2.608	2.298
rolling_tomato(222)	2.549	1.705	2.883	2.845	2.616
sailormen(300)	1.806	1.310	1.938	1.949	1.683
Total	2.024	1.452	2.366	2.357	2.094

Table 18. Simulation of sub-resolution sequences cut from D1

Sequence	720*480	(360*480)*2	(240*480)*3
crew(300)	2.347	2.345	2.342
harbour(300)	1.991	1.992	1.990
night(230)	2.220	2.220	2.217
pour_water(1017)	2.626	2.644	2.646
rolling_tomato(222)	2.883	2.891	2.894
sailormen(300)	1.938	1.932	1.924
Total	2.366	2.372	2.370

4.1.2 Comparison of Tool Performance

In LALVC, we have four modes including Skip, DC, Diff and Raw modes. We compare the gains of various tools. The results of comparisons are shown in Figure 32, Figure 33 and Figure 34. In Figure 32, the Diff mode enhances the improvement for the two slow motion sequences. In Figure 33, the mode decision selects the raw data for encoding of fast motion sequences. Consequently, the Skip and Diff modes do not show improvement for the sequences. In Figure 34, the Skip mode retains the best performance for computer generated sequences, which covers most of area in screen display that is still for successive frames.

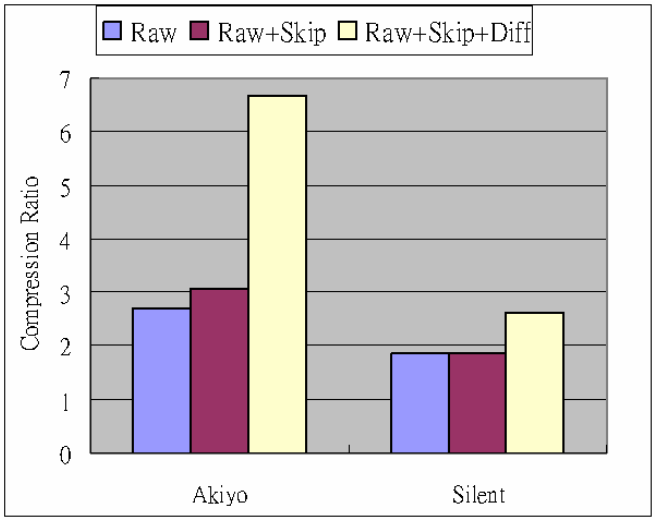


Figure 32. Tool performance of low motion sequences

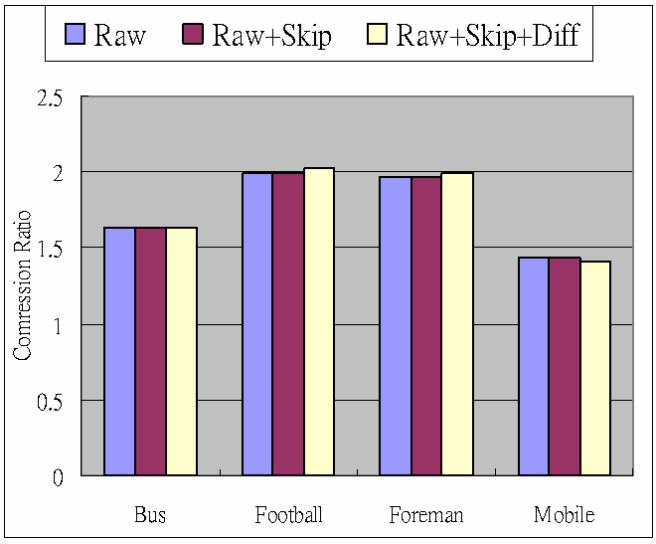


Figure 33. Tool performance of fast motion sequences

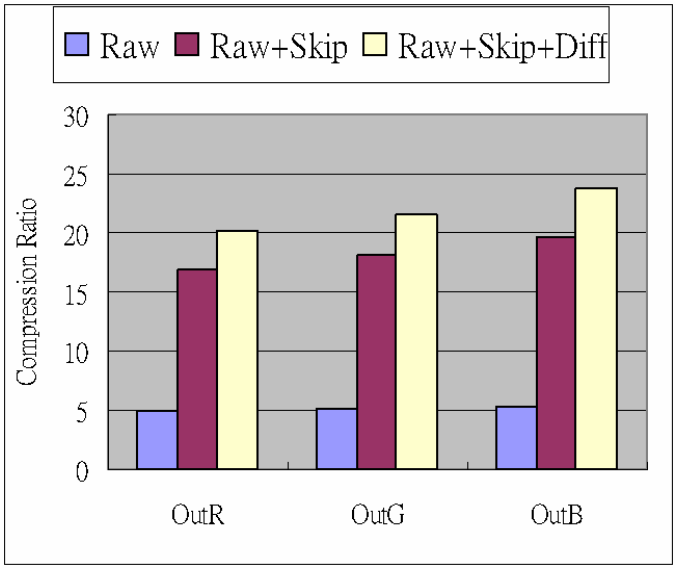


Figure 34. Tool performance of computer generated sequences

4.1.3 Execution Speed Evaluation

4.1.3.1 Profile analysis

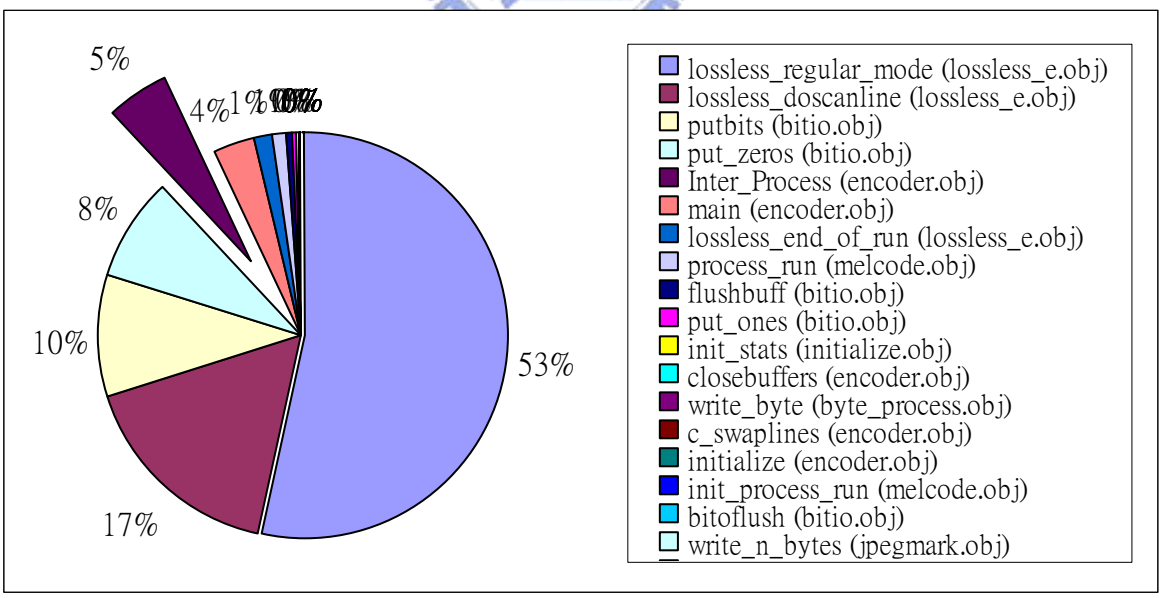


Figure 35. Profiling of LALVC encoding modules

In LALVC, the major difference of computation load in encoder and decoder is the inter-process including mode decision and residuals production. We run the software profile

and show the results in Figure 35. The platform is P4 2.0GHz desktop. OS is Windows XP. The test sequence is Foreman. In the profiling analysis, 5% of execution time is occupied by the inter-process. To evaluate whether the codec is symmetric, we do the simulations for statistics of run time.

4.1.3.2 Encoding and decoding rates

Table 19. Coding speed for various natural sequences

	Time(sec)		Coding rate (fps)	
Sequence	Encode	Decode	Encode	Decode
Akiyo	2.78	2.23	107.87	134.29
Bus	2.97	2.28	50.54	65.76
Football	4.86	3.59	51.45	69.56
Foreman	6.02	4.5	49.87	66.67
Mobile	6.31	5.73	47.53	52.32
Silent	5.97	4.05	50.27	74.11
		Average	55.36	71.46

For real encoding case, the encoding rate is 55.4 fps on the average for nature sequences. The decoding rate is 71 fps, which present the processing of 16 more frames than encoding. The detail results are shown in Table 19. In the other case, computer generated sequences are easy for execution than natural sequences. In Table 20, 133 fps is the encoding rate. The decoding rate is up to 584 frames.

Table 20. Coding speed of computer generated sequences

	Time(sec)		Coding rate (fps)	
Sequence	Encode	Decode	Encode	Decode
OutR	0.75	0.17	133.33	584.8
OutG	0.77	0.17	130.72	584.8
OutB	0.74	0.17	136.05	584.8
		Average	133.33	584.8

4.1.3.3 Complexity Analysis

LALVC is low complex, which is proper for software implementation and hardwired circuit design. For the preprocessing, if the frame resolution is $N*N$, LALVC needs $(7N^2-6N)$ adders to execute the mode decision and zero-motion prediction residual production as in Figure 13. In addition, we reduce the context model from three to two gradients to retain the identical performance with less memory requirement. The fixed predictor and entropy coder have been proven to have low complexity in JPEG-LS. The execution time of LALVC algorithm is observed under P4 2.0 GHz Desktop and Windows XP. The results show that the encoding rate is about 55 frames per second (fps) for nature CIF sequences and near 133 fps for computer-generated CIF sequences. For the LALVC decoder, the averaged decoding rates are 70 fps and 584 fps for the natural video and synthetic video bitstreams, respectively.

In [11], the mode decision needs the side information that represents a block (5 by 5 pixels) of data. After calculating the MSE for spatial, spectral and temporal predictors, the suitable one is chosen. The MSE computation is higher than LALVC.

4.1.4 Hardware Implementation Result

The simple image coder JPEG-LS, which includes fixed predictor and context model, is implemented in hardware [14]. The hardware implementation reveals that the chip area is dominated by the SRAM size than circuits of function blocks. The memory requirement is discussed in the previous section. The following hardware implementation does not include the real SRAM. The gate count statistics in Table 21 only cover the function blocks used in LALVC without the SRAM. To give the results of hardware implementation, the flow of functions is shown in Figure 36 and the gate count is listed in Table 21.

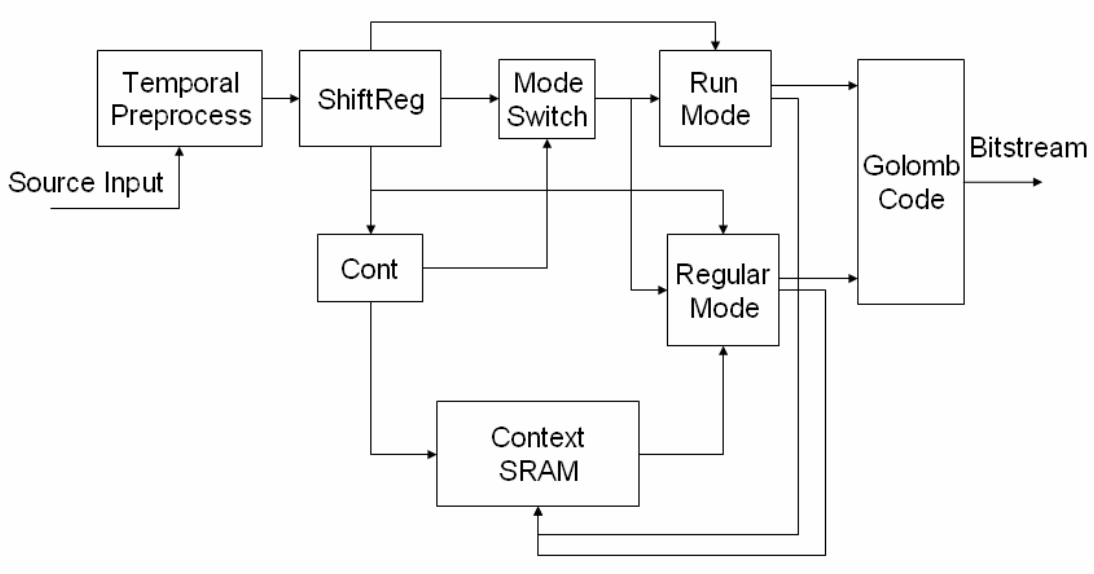


Figure 36. Flow of function blocks in Hardware

Table 21. Gates count statistics of hardware modules (.18μm)

	Area	Gate count
Regular Mode	13497.5	1377.30
Run Mode	88205.24	9000.53
Mode Switch(5ns)	678.8	69.27
Temporal(5ns)	10751.4	1097.08
Cont	3338	340.61
Shift_Reg(5ns)	7882.3	804.32
Total	116470.94	11884.79

4.2 Summary

Simulation results of LALVC including the compression ratio and execution speed are given here. For evaluating the LALVC performance, we take both natural and computer-generated video sequences. For natural video sequences, we take the same test sequences for MPEG standards. The natural video sequences are in CIF (352x288) and YCbCr=4:2:0 formats. Each pixel is presented in 8 bits. Only Y component is used for simulation. The synthetic video

sequences including scrolling web pages and adapting window size are captured from the computer screen.

Table 22. Compression ratio list*

	WinRAR	JPEG-LS	CALIC	LAVLC	Opt_16_line
Akiyo _Y	5.85	6.74	3.43	6.35	6.76
Bus _Y	1.29	1.44	1.09	1.64	1.64
Football _Y	1.67	1.75	1.28	1.99	2.03
Foreman _Y	1.80	1.84	1.26	1.96	2.03
Mobile _Y	1.32	1.36	1.07	1.40	1.43
Silent _Y	2.52	2.61	1.49	2.53	2.64
Browser _R	30.28	12.84	18.20	20.80	20.63
Browser _G	29.73	13.69	18.64	22.15	22.15
Browser _B	29.46	15.07	19.16	24.32	24.50

*The sequences is preprocess with enforced-zero-motion prediction.

The synthetic video sequences are in CIF resolution and RGB format. The comparisons of compression ratio are shown in Table 22 for former six natural sequences and the last three synthetic sequences. As to the lossless compression algorithms, the observations of JPEG-LS, CALIC and WinRAR are provided. WinRAR is commercial software for generic data compression. The performance of context modeling, intra predictor and entropy coder in LALVC is compressed with the algorithms including WinRAR, CALIC and JPEG-LS. Prior to encoding, each input sequence is changed by zero motion prediction and sign remapping to form a new sequence with unsigned prediction residuals. Table 22 gives the comparisons based on the total amount of bits used to represent the new sequences. LALVC has the best performance in the nature sequences on the average. As to the computer-like sequences, LALVC can perform better than JPEG-LS and CALIC on the average. Only WinRAR is better than LALVC in the synthetic video sequences.

Chapter 5

Conclusion

5.1 Contributions

In this thesis, we have presented a low-complexity and low-latency LALVC algorithm for real-time interactive applications for universal multimedia access environment. LALVC consists of three parts covering the preprocessing, the mode-dependent spatial prediction and coding prediction. The preprocessing can efficiently remove the temporal redundancy via the zero-motion prediction and optimized mode decision. The proposed mode decision can adapt the predictor to make accurate prediction on various video sources. In addition, the syntax is simple and easy for the encoder and decoder implementation. The simulation results show that LALVC has significant compression ratios for both natural and synthetic sequences. For hardware implementation issue, we design several schemes for memory reduction. Unrolling the loop execution is beneficial for real-time coding due to parallel execution in hardware. The properties of low complexity and low delay can match real-time and low consumption requirements. Since no multipliers and no dividers are required in LALVC, hardware design architecture is applicable for hardware realization including FPGA implementation or ASIC design.

5.2 Future Works

The structure of LAVLC is based on raster scanning order. The simulation results show the syntax of line-based header information. For a high resolution display, the extension version can adjust the line-level information. Division of the lines into several small portions, and each line is followed by one tag to avoid the large local area variation. The control range of the tag is another issue to evaluate

Color plane coding scheme needs more investigations for achieving the better encoding performance.

If random access is supported, I-frame encoding concept could be incorporated into the structure at cost of the coding efficiency.

For a complete hardware implementation, SRAM module and post-layout simulation should be involved for real simulation.

References

[1] X. Wu and N. Memon, ”Context-based, adaptive, lossless image coding, **”** *IEEE Trans. on Communications*, vol. 45, no. 4, pp.437–444, Apr. 1997.

[2] M.J. Weinberger, G. Seroussi and G. Sapiro, ”The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS,” *IEEE Trans. on Image Processing*, vol.9, no.8, pp.1309-1324, Aug. 2000.

[3] R. Barequet and M. Feder, ”SICLIC: a simple inter-color lossless image coder,” *Proc. DCC '99*, 29-31 March 1999, pp.501–510.

[4] Xiaolin Wu and N. Memon, ”Context-based lossless interband compression-extending CALIC,**”** *IEEE Trans. on Image Processing*, vol.9, no.6, pp.994–1001, June 2000.

[5] E.S.G. Carotti, J.C. De Martin and A.R. Meo, ” Backward-adaptive lossless compression of video sequences,” *Proc. ICASSP'02*. vol.4, vol.4, 13-17 May, 2002, pp.3417-3420.

[6] D. Brunello, G. Calvagno, G.A. Mian and R. Rinaldo, ” Lossless compression of video using temporal information,” *IEEE Trans. on Image Processing*, vol.12, no.2, pp. 132–139, Feb. 2003.

[7] N.D. Memon and K. Sayood, ”Lossless compression of video sequences, ” *IEEE Trans. on Communications*, vol. 44, no.10, pp.1340–1345, Oct. 1996.

[8] S. Todd, G. G. Langdon, Jr., and J. Rissanen, “Parameter reduction and context selection for compression of the gray-scale images,” IBM Jl. Res. Develop., vol. 29 (2), pp. 188-193, Mar. 1985.

[9] S.W. Golomb, **“**Run-length encodings**,”** *IEEE Trans. Inform. Theory*, vol. IT-12, pp.399-401, 1966.

[10] http://www.xs4all.nl/~brw/ds_products/hot_math.html

[11] A. J. Penrose and N. A. D., Eurographics "Extending lossless image compression", Fitzwilliam College, Cambridge, ISBN 0-9521097-8-6,1999, UK '99, 13-15 Apr 1999 (http://www.cl.cam.ac.uk/users/nad/pubs/#COMPRESSION)

[12] M.-F. Zhang, J. Hu and L.-M. Zhang, **“**Lossless video compression using combination of temporal and spatial prediction,**”** *Proc. NNSP’2003*, vol.2, pp.1193-1196, Dec. 14-17, 2003.

[13] “Lossless and near-lossless coding of continuous tone still images(JPEG-LS),”ISO/IEC JTC 1/SC29/WG1 FCD 14495-public draft 1997/7/16

[14] Andreas Sayakis and Michael Piorun, “Benchmarking and hardware implementation of JPEG-LS,” *Proc. ICIP’02*, vol.2, pp.949-952, 2002.

[15] Kyeong Ho Yang, and A. Farid Faryar,”A contex-based predictive coder for lossless and near-lossless compression of video,” *Proc. ICIP00*, vol.1, pp.144-147, 2000.

Appendix A. Testing Sequences

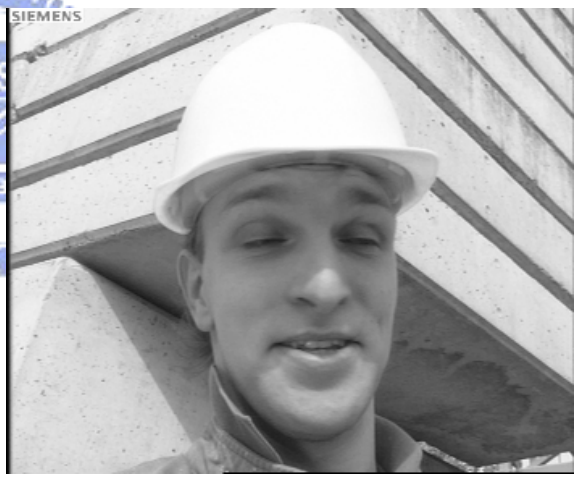
A. CIF format (352*288) natural sequences

1. Akiyo

2. Bus

3. Football

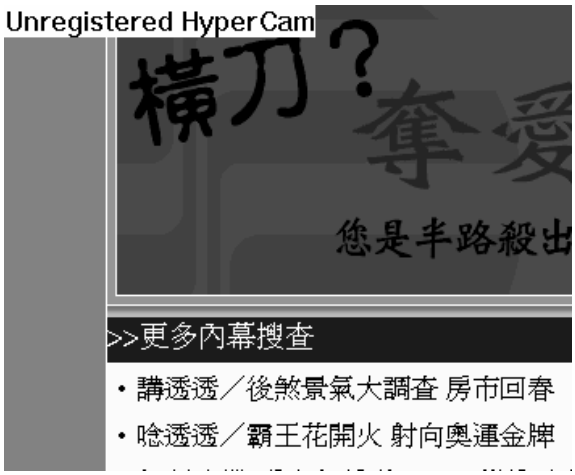
4. Foreman



5. Mobile

6. Silent

B. CIF format (352*288) computer-like sequence



C. D1 resolution (720*480) natural sequences

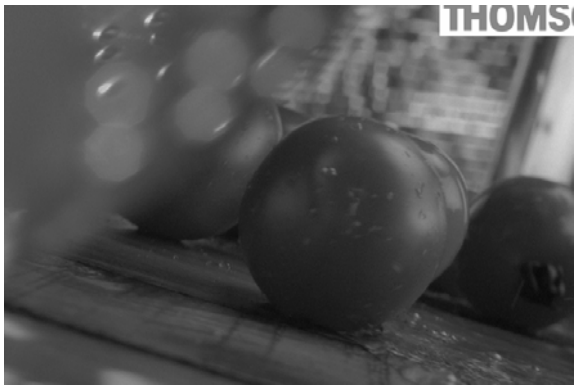
1. Crew

2. Harbour

3. Night

4. Pour_water

5. Rolling_tomato



6. Sailormen