

國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

應用於無線視訊傳輸之可變長度解碼器



**Soft Variable Length Decoding for the Wireless Video Transmission**

研究生：劉子明

指導教授：李鎮宜 博士

中華民國九十三年六月

# 應用於無線視訊傳輸之可變長度解碼器

學生：劉子明

指導教授：李鎮宜 博士

國立交通大學  
電子工程學系 電子研究所碩士班

## 摘要

可變長度編碼在近來的視訊及影像編碼當中一直被廣泛的應用。然而，傳統的解碼方式可能會因為傳輸的錯誤而產生非同步的解碼，甚至產生錯誤傳遞的情況。而為了改善其容錯的能力，越來越多的研究者投入心力在聯合訊源與通道編碼的設計領域上。一種新型的可變長度解碼器已經慢慢的浮現出來，他能夠在頻寬有限和廣播的系統當中抵抗傳輸的錯誤。而這樣的解碼方式通常需要保留許多的狀態，尤其是當編碼表格很大的時候。因此，這種新型的解碼方式在實作的時候，會產生很高的複雜度和需要很大的記憶體容量。

為了減少表格的大小和記憶體讀取的次數，我們提出了一種低複雜度和低記憶體使用量的方式。甚者，我們更提出一種 "Symbol-alias" 的量測方法來提高對於解碼效能的猜測。利用我們所提出的 Black-Box 模型，我們可以在效能以及複雜度上取得最佳的平衡點。

最後，利用我們所提出的效能模型，一個高效能、低複雜度的可變長度解碼器已被我們實現。在可允許的效能損失之下，它不只減低了記憶體的使用量，更減少了表格的大小。而整個系統的模擬是在 MPEG-4/UDP-Lite/UEP/AWGN 的平台上所實現。無論是跟傳統的解碼或者是擁有錯誤更正能力的 RVLC 解碼的比較上，我們平均可以提昇畫面的品質 0.4~2.9dB，並且提供更好的主觀品質。

# Soft Variable Length Decoding for the Wireless Video Transmission

Student: Tsu-Ming Liu

Advisor: Dr. Chen-Yi Lee

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

## Abstract

Variable Length Codes (VLCs) are extensively used in recent video and image coding standard. However, traditional table look-up hard decoding may lose synchronization and induce error propagation over a noisy channel. To improve the error resilience of VLC, more and more researchers pay lots of attention about the joint source and channel design. The soft VLC decoding method has emerged to resist the channel disturbances on the environment of band-limited and broadcasting system. Such design generally needs to maintain many states when the table size grows. Hence, soft VLC decoders have problems of high complexity and high memory access.

To reduce the table size and the number of memory access, we propose a soft VLC decoder with low memory access and low complexity approach. Further, a novel measurement of “symbol-alias” is presented to provide more accurate performance estimation. With the proposed Black-Box model, we can achieve the optimal trade-off between performance and complexity.

Finally, a memory-efficient and low-complexity soft VLC decoder using performance modeling is proposed. It exploits not only modified sorting scheme to reduce the memory access, but also table redundancy to reduce the table size at the cost of minor performance loss. The system evaluation is achieved in the model of MPEG-4/UDP-Lite/UEP/AWGN. We averagely improve the PSNR by 0.4~2.9dB (i.e. 40~80% improvement) and offer better subjective quality compared with the traditional VLC decoding and standard-support RVLC decoding.



## *Acknowledgements*

I would like to express my deepest gratitude to my advisor Dr. *Chen-Yi Lee* for his sophomore enthusiastic guidance and encouragement throughout the research, and give him and his family my best wish faithfully.

Especially, I much appreciate my senior Mr. *Wen-Hsiao Peng* and junior Mr. *Sheng-Zen Wang* for their fruitful discussion and comments during my research. Also, I would like to thanks my senior Dr. *Bai-Jue Shieh* and my *SI2* group mate Mr. *Cheng-Hung Liu* for their great help in the period of my research. In addition, I want to thank all members of the *SI2* group of NCTU for plenty of worthwhile assistance in my graduated lives.

Finally, I give the greatest respect and love to my family and my girl-friend, *Yu-Fen Chuang*. I much admire her thoughtfulness, and I want to express my highest appreciation and dedicate the thesis to her for assisting me to achieve the most important stage in my life. I never let her down and hope her and my family happy now and forever.

# Contents

<b>Chapter 1. Introduction .....</b>	<b>1</b>
1.1 Motivation .....	1
1.2 Joint Source and Channel Design .....	2
1.3 Contributions of this Thesis .....	9
1.4 Thesis Organization .....	11
<b>Chapter 2. Soft Decoding of Variable Length Codes.....</b>	<b>12</b>
2.1 Background.....	12
2.2 Soft Input Soft Output Algorithm.....	13
2.2.1 Algorithm Translation .....	13
2.2.2 Algorithm Modification .....	15
<b>Chapter 3. Memory Efficient Design Approach .....</b>	<b>19</b>
3.1 Adaptive Selection Algorithm.....	19
3.1.1 Modified Sorting Scheme.....	19
3.1.2 Performance Comparison .....	21
3.2 Complexity Analysis.....	22
3.3 Summary .....	24
<b>Chapter 4. Low Complexity Design Approach.....</b>	<b>25</b>
4.1 Symbol Merging Algorithm .....	25
4.1.1 Metric Formulation of “Balance Degree” .....	27
4.2 Table Merging Algorithm .....	28
4.2.1 Code-Word Merging .....	28
4.2.2 Prefix Merging .....	29
4.2.3 Merged Table.....	29
4.3 Performance Evaluation .....	29
4.4 Summary .....	31

<b>Chapter 5. Performance Modeling</b> .....	<b>33</b>
5.1 Black-Box Model .....	33
5.1.1 Algorithm-Sensitive Parameters .....	34
5.1.2 Application-Sensitive Parameters .....	36
5.1.3 Table-Sensitive Parameters .....	36
5.1.3.1 Intra Alias .....	36
5.1.3.2 Inter Alias .....	37
5.3 Performance Estimation .....	38
5.4 Summary .....	41
<b>Chapter 6. Performance Evaluation on MPEG-4</b> .....	<b>42</b>
6.1 Environment setup.....	42
6.1.1 Source Model .....	43
6.1.2 Channel Model .....	45
6.2 Performance Evaluation on MPEG-4/UDP-Lite/UEP/AWGN .....	49
<b>Chapter 7. Conclusions and Future Work</b> .....	<b>54</b>
<b>Bibliography</b> .....	<b>56</b>
<b>Appendix A Symbol-Merging Algorithm</b> .....	<b>60</b>
<b>Appendix B Table-Merging Algorithm</b> .....	<b>78</b>
<b>About the Author</b> .....	<b>84</b>

# List of Figures

Figure 1.1	The on-going tree of error handling .....	3
Figure 1.2	The categories of implementation and representation in JSC design ....	6
Figure 1.3	Symbol-constrained directed graph representation for VLC decoding... 7	
Figure 1.4	Bit-constrained directed graph representation for VLC decoding .....	8
Figure 2.1	High-level description of the decoding procedure with algorithm translation .....	14
Figure 2.2	The algorithm translation between symbol-constrained directed graph and the SISO algorithm .....	15
Figure 2.3	The original (a) and real case (b) of VLC table .....	16
Figure 2.4	The algorithm modification due to the constraint change .....	17
Figure 2.5	High-level description of the decoding procedure with algorithm modification .....	18
Figure 3.1	The graph representation in approximated decoding .....	20
Figure 3.2	The comparison between the AMAP-2 (a) and the proposed <i>Adaptive</i> AMAP-2 (b) .....	21
Figure 3.3	The comparison of performance (a) and memory access (b) vs. SNR	22
Figure 3.4	Complexity analysis in terms of each symbol state numbers.....	24
Figure 3.5	The comparison with complexity issue in terms of state numbers (a) and total state numbers (b) using the VLC table of Figure 2.3.....	24
Figure 4.1	The tree-structured VLC (a) and scalable algorithm with hard and soft decoding (b).....	26
Figure 4.2	A simple VLC table with merge-0(a), merge-1(b) and merge-2(c).....	27
Figure 4.3	The evaluation of execution time (a) and performance (b) with different symbol-merging table in Figure 4.2.....	30
Figure 4.4	The formulation of “Improved Ratio”.....	32
Figure 5.1	The B-B model (a) and the evaluation of source table (b).....	34
Figure 5.2	The relationship between performance and each parameter .....	34

Figure 5.3	The complexity (a) and performance (b) in different ‘z’ .....	35
Figure 5.4	The performance with convergence and saturation point in AMAP-2 (a) and A-AMAP-2 (b).....	35
Figure 5.5	The optimization of performance in different ‘N*’.....	36
Figure 5.6	The symbol-alias of VLC table (a)(b) .....	37
Figure 5.7	The performance evaluation of ‘intra alias’ and ‘inter alias’ (b) in different ‘T’ .....	38
Figure 5.8	VLC tables (a) and measurements (b) for the same source.....	39
Figure 5.9	Coding performance with different VLC coding table.....	39
Figure 5.10	The simulated parameters (a) and PSNR comparison (b) within 50 frames.....	40
Figure 5.11	The comparison on the 1 <sup>st</sup> frames of video sequence.....	41
Figure 6.1	The proposed overall simulation environment of soft VLC decoder... 43	
Figure 6.2	The data partition mode in the MPEG-2 (a) and MPEG-4 (b).....	44
Figure 6.3	The high-level description of ESCAPE code handler on MPEG-2 and MPEG-4.....	45
Figure 6.4	The content (a) and ratio (b) of one video packet in MPEG0-4.....	46
Figure 6.5	The soft input of VLC decoder .....	47
Figure 6.6	The performance improvement (a)(b)with different quantization level	48
Figure 6.7	Overlooking bit errors in application layer.....	48
Figure 6.8	Average PSNR of Y-component for proposed soft and TLU VLD.....	49
Figure 6.9	PSNR vs. AWGN channel performance.....	51
Figure 6.10	PSNR vs. Burst error performance .....	51
Figure 6.11	The comparison between the proposed soft VLD and the RVLD.....	52
Figure 6.12	The comparison between the proposed soft VLC decoder and the standardized VLC decoder for table-merging algorithm.....	53



# List of Tables

Table 4.1	The reduction of table size by symbol-merging scheme.....	28
Table 4.2	The comparison with existing design.....	32
Table 6.1	The trade-off between error correction and channel bandwidth.....	52
Table 6.2	The PSNR improvement within different video characteristics.....	53



# Chapter 1

## Introduction

### *1.1 Motivation*

Variable Length Codes (VLCs), also called Huffman codes [1] are common used to approach the entropy rate of a given data source. They are extensively used in recent image and video coding standards including JPEG, MPEG-1/2/4 and the newly design of H.264 [2]. However, most of the VLC designs are highly sensitive to error disturbances. Table look-up decoding method may render extremely vulnerability and lose synchronization over a noisy channel. Although many conventional methods like automatic repeat request (ARQ) and forward error correction (FEC) reduce the effect of channel errors, these solutions have been found to be expensive in band-limited communications of delay sensitive video signals [3].

Particularly, ARQ-based designs are inadequate for the broadcast transmission due to the necessary of backward channel. Besides, they may induce significant delay that would potentially result in network congestion; While FEC designs may be bandwidth-inefficient when the channel conditions are fairly mild, and fine-tune to a particular error-rate when the channel condition differs. Therefore, it is strongly interest to look for an alternative design to reduce the error sensitivity of variable length encoded video source.

In recent years, more and more researchers pay lots of attention about the source and channel design jointly. To improve the error resilience of VLC, joint source and channel (JSC) design has emerged to resist the channel disturbances on the environment of

band-limited system and broadcasting transmission. Several JSC designers concentrated on variable length encoded data since most of the video application exploited VLC-based compression method. However, the main problems of JSC design are the complicated computation and the greatly memory utilization in the decoding process of the sequence estimation. The reduced complexity or sub-optimal JSC designs [24][25][26] are proposed to diminish the decoding complexity in the VLC-based source transmission. However, these designs are still inadequate for the large source table and the separate source tables. In this thesis, we focus on the implementation of JSC design. Low complexity and memory efficient design approach have been proposed to resolve the error propagation and outperformed the traditional designs on VLC decoding.

## *1.2 Joint Source and Channel Design*

In the past, the designs of source and channel coder have been performed separately. This often makes excellent senses and could be proved by the separation theorem of Shannon [4]. However, Shannon's theorem effectively assumes that source coder removes all data redundancy, and the channel coder inserts additional redundancy to protect the source data due to the impairment of physical channel. This separation does not make as much practical senses. It has been shown that the separation theorem does not hold for all channel conditions [5]. When it does hold, it needs to exploit an optimal source and channel coder pair that may not be suitable for the practical system.

To improve the error robustness on VLCs, all the solutions can be classified into three types (cf. Figure 1.1). They are error resilient, error concealment and error recovery respectively. Error resilience methods are performed in the encoder side, and the respective decoding procedures are defined by the video standard. To make the compressed video data more robust to channel errors, the MPEG-4 standard incorporated several error resilient tools, including data partition (DP), header extension code (HEC) and re-synchronization marker (RM) [6]. On the other hand, decoder provides the error concealment and recovery to improve the video quality. Particularly, the error concealment methods are proposed to conceal the errors, but seem to have its limitation [7]. They often assumed that video errors have been correctly located; otherwise error concealment cannot be properly applied.

Error recovery can be partitioned into three levels that are source level, channel level and joint source-channel level. In the consideration of source-level error recovery, reversible variable length codes (RVLCs) [8] are realized in the MPEG-4 and the newly design of H.264. Many source-level error recovery methods are suggested including RVLC, error resilient entropy coding (EREC) [9] and self-synchronization VLC (SSVLC) [10]. These methods use the syntax and codeword structure to reconstruct the source data and do not consider any channel behavior. The improvement of source-level error recovery is still insufficient. On the contrary, the improvement of channel-level error recovery is significant like the well-known scheme of Viterbi decoder or turbo decoder. However, the usage of channel-level error recovery is very expensive for the band-limited system. The trade-off between source and channel level is proposed that can be termed as JSC design on the soft VLC decoder. The idea of JSC design has been gaining increasing attention in recent years. This is because that the significant growth of multimedia wireless communication on the channels of noisy and band-limited. Besides, the channel conditions about broadcasting on DVB system [11] faced the channel behaviors without backward notification.

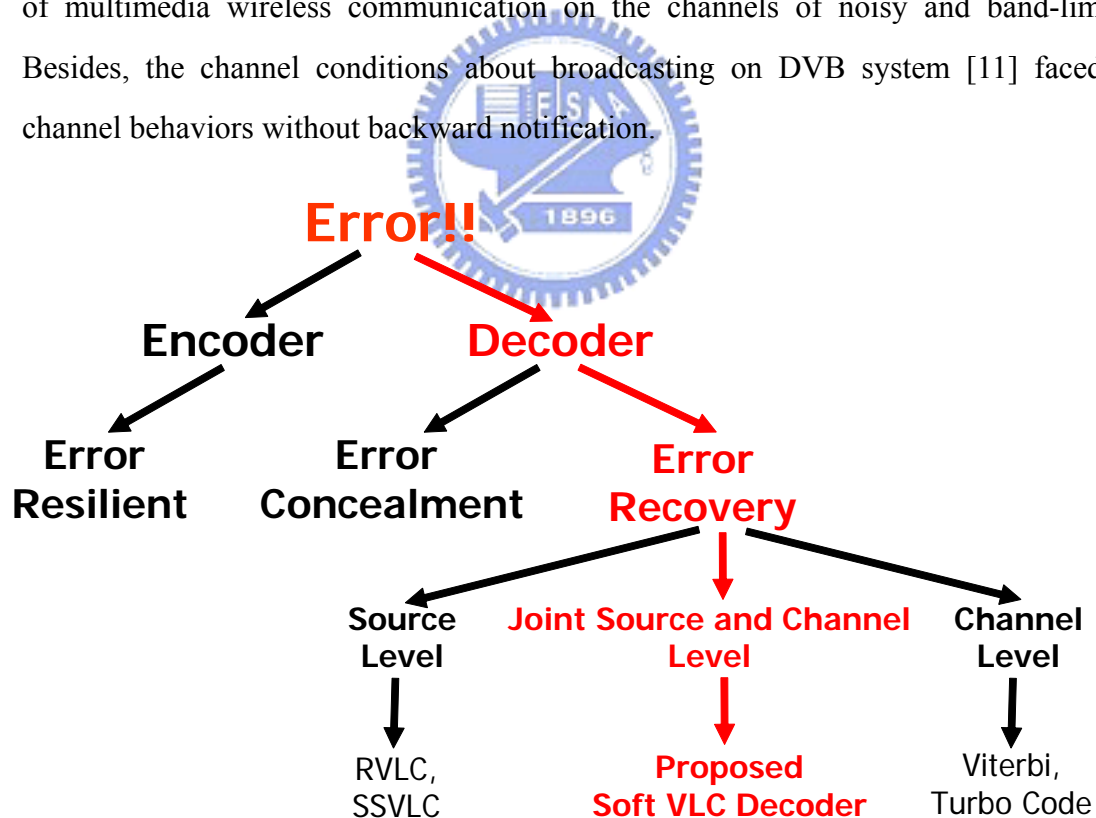


Figure 1.1 : The on-going tree of error handling.

Based on the different derivation or formulation of intermediate metric in JSC design, it can be classified into three categories in [16] (e.g. [18] [21] [26]). We just omit the

complicated derivation of algorithmic metric. Instead, behaviors of these three categories are discussed here and compared with each other. Performance and complexity are the crucial cues for our final decision of implementation method.

## **Maximal Likelihood / Soft-Input Soft-Output Decoding Method**

One category of coder is Maximum Likelihood (ML) decoding method. The ML decoder is investigated in the joint area of source and channel design. Viterbi decoder using ML decoding algorithm is famous for many decades, and be considered as the decoding process of fixed length codes. Most applications exploit variable length codes to compress the source data, but lead to loss of error resilience. A modified version of the Viterbi algorithm [17] may now be used to perform maximum likelihood decoding of VLCs and improve the error robustness [18]. The main problem in applying the Viterbi algorithm directly is the fact that the state transition will result in a variable number of bits. Therefore, it is necessary to keep track of the position of each transition and lead to a great number of states to be survived.

In [19], the authors introduced the Soft-Input Soft-Output (SISO) approach to improve the coding performance when the source data has been corrupted by additive white Gaussian noise (AWGN). The SISO VLC decoder involves no modification to the encoder side. It simply receives input as a packet of known length containing corrupted VLC data, and produces or estimates the codeword sequence that is most likely to the input of the VLC encoder. It behaves as a ML decoding process for VLCs, uses the Hamming distance of hard input and cumulative square errors of soft input as the derivation of intermediate metric. In addition, SISO decoding algorithm is similar with soft output Viterbi algorithm (SOVA) [20] that provides the soft output information as a confidential level or reliability in the back-end decoding process.

## **Maximum A Posteriori Decoding Method**

Maximum A Posteriori sequence estimation, termed MAP decoding for VLCs is investigated. In the last paragraph, we classify the derivation of metric as ML decoding.

Otherwise, we classify the newly derivation of metric as a MAP decoding. The Viterbi algorithm was re-derived with a priori or a posteriori information for MAP decoding [21]. More detailed formulation about intermediate metric is published in several literatures. It can be noted that the main difference between ML and MAP decoding algorithm is the intermediate metric derivation. In practice, the MAP decoding method outperforms the ML decoding method in terms of decoding performance, but offered a complicated computation of metric for more accurate sequence estimation. Many researchers focus on the complexity reduction in algorithmic level [23]-[25]. However, it is still insufficient for the consideration on the long input-sequence and large symbol-table. In the point of comparison between ML/SISO and MAP decoding process, we can see that SISO decoding with ML algorithm approximates closely to MAP decoding algorithm and provides the reliability output and less complexity [16] [22].

## Sequential Decoding Method

Sequential decoding predates the Viterbi decoding. It is discovered by Wozencraft in 1960. The decoding process traverses a tree to find out the possible paths that could be taken depend on the input data. The transition paths are followed or eliminated through the likelihood comparison, threshold or other criteria. Though average decoding complexity is reasonable, there is a great possibility of repeated computation and a wide variation on complexity that depending on error occurrences. For the practical communication system, the complexity is a big problem to fit any channel behavior. Besides, the performance of sequential decoding strongly relies on the instantaneous error events. To improve the coding complexity, fast sequential decoding algorithm using a stack is proposed [26], but the improvements still have its limit compared with ML or MAP decoding method.

Considering the large number of codeword in MPEG-4, the coding complexity of JSC design will become a critical bottleneck. The performance and complexity of sequential decoding will depend on the channel condition, and unsuitable for the practical VLSI implementation. Further, MAP decoding algorithm provides more capability of error correction slightly than ML/SISO [22], but high complexity is its penalty about the

computation of intermediate metric. Consequently, we use ML/SISO decoding algorithm as our implementation of VLC decoder.

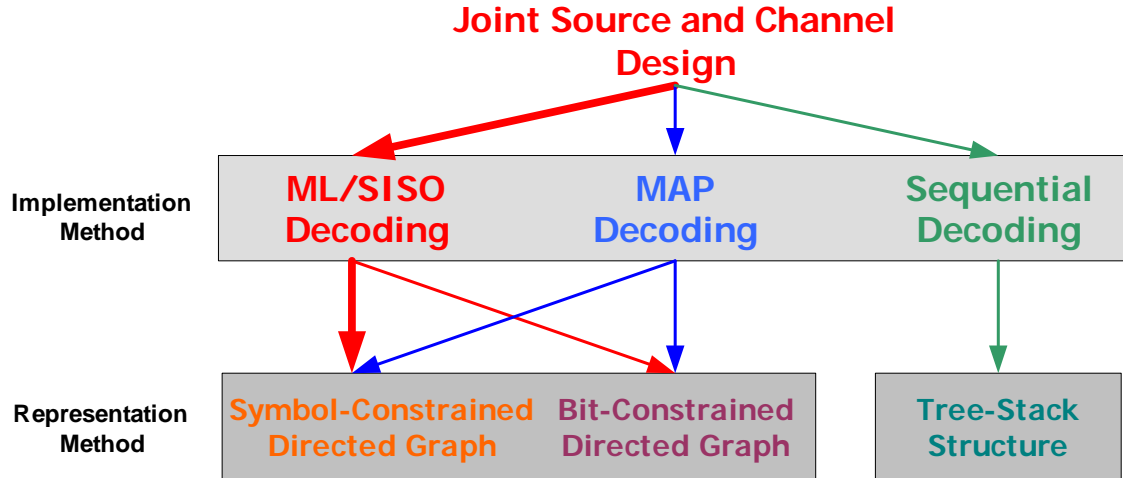


Figure 1.2 : The categories of implementation and representation in JSC design.

For implementation and representation method in JSC design, Figure 1.2 shows the relation between each other. Implementation method has been briefly discussed above, and representation method is composed of tree or trellis structure. Trellis representation can be used as a representation of fix length path label such as Viterbi decoder. The Viterbi decoder kept only one of the paths entering a state as the survivor path and the others are pruned. However, in the case of VLCs, different paths entering a state have consumed a different number of bits from the received sequence and can be extended differently. Therefore, the case of VLCs cannot use a traditional trellis representation anymore and needs more complicated graph representations to be solved. The first works in this area of graph decoding have been proposed by Demir & Sayood [13] and Park & Miller [23]. These new graph representations have been proposed and summarized in [28]. They are symbol-constrained and bit-constrained directed graph respectively. In this thesis, we focus on soft VLC decoding by performing ML/SISO algorithm on the symbol-based VLC trellis decoding [13]-[15].

## Symbol-Constrained Directed Graph

The representation of Symbol-Constrained Directed Graph that we call it as SCDG here is introduced in [13] [28]. The SCDG representation retains many survivors when there are paths with different number of *symbols* coming at the considered state for a

given *bit* position. Example of SCDG decoding representation is described in Figure 1.3 for the VLCs of dimension  $T = 3$  and codeword sets  $\{0, 10, 11\}$ .

There are three-axis that should be notified in Figure 1.3, they are symbol-step  $i$ , codeword-step  $j$  and bit-step  $k$ . Each symbol step represents the number of decoded symbols. In Figure 1.3, the total decoded number  $N$  is equal to 3. This information can be retrieved through the syntax or the coding behavior of the JSC decoding process. In addition, each codeword-step stands for the different code-symbol in the pre-defined VLC table. Meanwhile, each codeword-step contains the different bit-step depending on the symbol-step  $i$ . Each square is the bit-state, and the decoded bit-number is resided in the center. Each dotted square or rectangle keeps the same codeword  $j$  for a given symbol-step  $i$ . We can see that each transition path from one square to the other square exploits the transition probability. The pruning operation will be performed when there are two arrows pointing to the same bit-state. To obtain the final solution, decoder will stop constructing this graph in symbol-step 3 due to the known information  $N$ . Further, we choose the three bit-states with dotted circles as our candidates because they have the known constraint (i.e. 3-symbol, 6-bit). After the comparison of intermediate metric, we can choose the smallest one as our decision state, and trace-back to decode the left symbols. More detailed decoding process will be recalled in chapter 2.

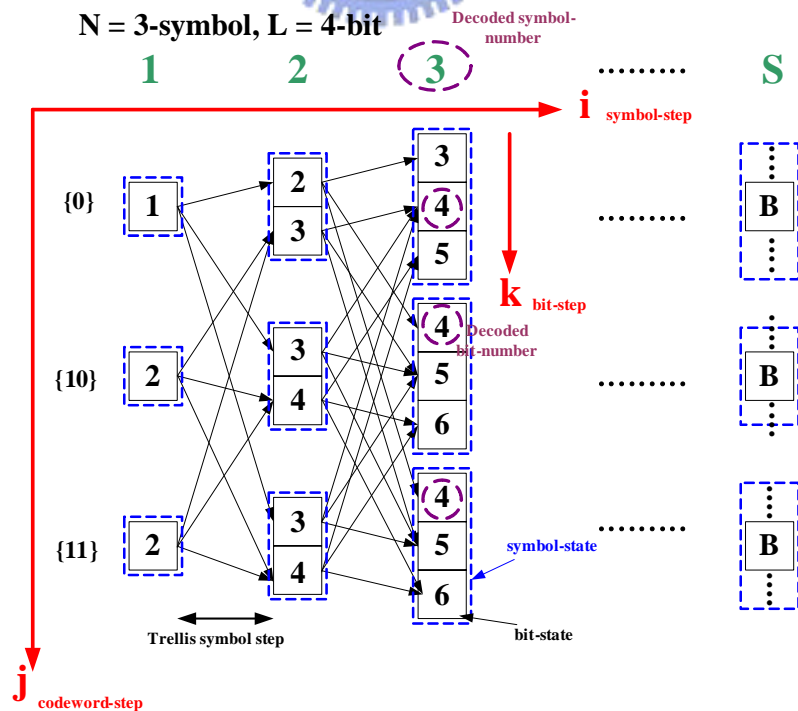


Figure 1.3 : Symbol-constrained directed graph representation for VLC decoding.



## Bit-Constrained Directed Graph

In addition to the SCDG representation, Bit-Constrained Directed Graph that we call it as BCDG here is introduced in [23] [28]. The BCDG representation retains many survivors when there are paths with different number of *bits* coming at the considered state for a given *symbol* position. Example of BCDG decoding representation is described in Figure 1.4 for the VLCs of dimension  $T = 3$  and codeword sets  $\{0, 10, 11\}$ . As the discussion of SCDG, the decoding process of BCDG is similar to the SCDG except that the roles of bit and symbol are exchanged. Similarly, we can perform JSC decoding process of VLCs with BCDG representation.

However, the transition path in BCDG is more complicated than SCDG. For the consideration of coding complexity, we need two-dimensional pointers to address where the arrows point to. This complexity becomes more prominent on the implementation of large VLC tables, such as the AC-coefficient table with 103 symbols in MPEG-4 [30]. Therefore, we choose the representation of SCDG as our implementation in this thesis. Although the SCDG representation may lose a little performance when the sub-optimal solution is imposed, it is of great worth when dealing with the large VLC tables.

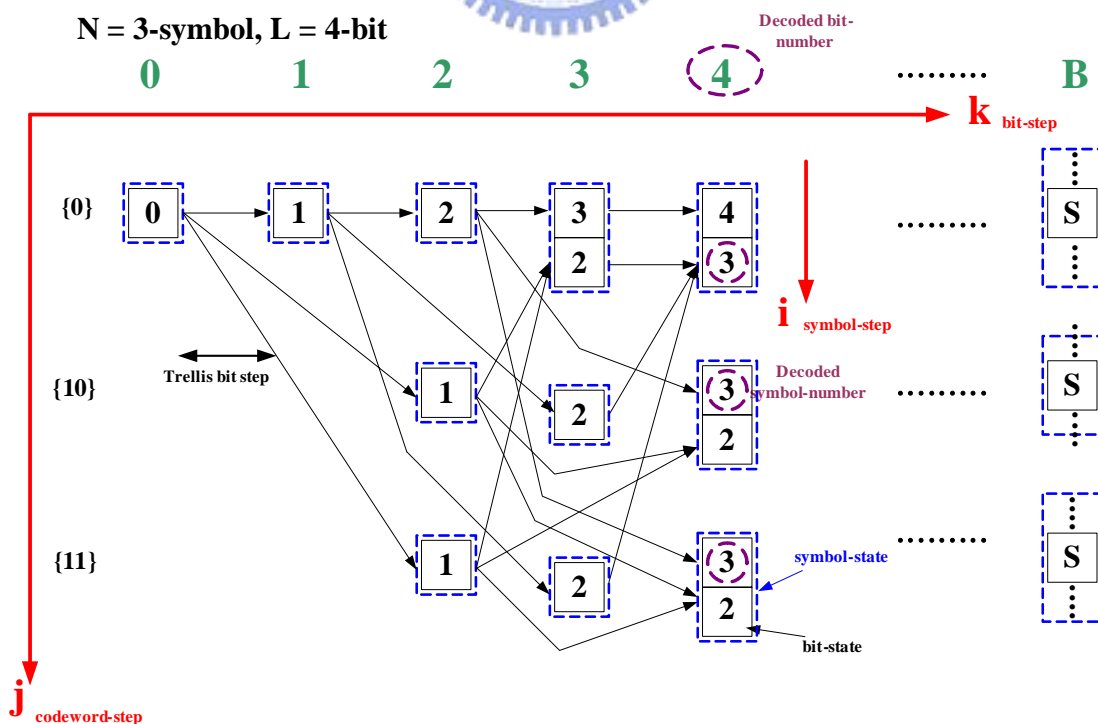


Figure 1.4 : Bit-constrained directed graph representation for VLC decoding.

These representation methods perform well for both hard and soft input, but show its error correction capability for soft input in this thesis. In the ML/SISO decoding algorithm, the improvement can be achieved when compared with classical table look-up decoding method is significant, but the complexity is prohibitive.

In this thesis, we will focus on the implementation of practical application, such as MPEG-4 and H.264. We use Soft-Input Soft-Output (SISO) decoding algorithm as our basis of metric derivation. Compared to the Maximum A Posteriori (MAP) decoding algorithm and sequential decoding algorithm, SISO algorithm performs the optimal trade-off between performance and complexity. Further, it utilized a simpler metric (i.e. absolute difference) to improve the error resilience on the decoding process of VLCs. From the graph representation point of view, we choose the SCDG as our graph representation of SISO decoding algorithm. Finally, we outline our contribution of this thesis in the next section which including the algorithm simplification and complexity reduction. Further, a memory efficient and performance modeling is proposed to achieve the low memory utilization and optimal performance.

### *1.3 Contribution of this Thesis*

From the previous statements, the JSC design algorithm chosen is the SCDG-based ML/SISO VLC decoding method. This new decoding technique for variable length codes considered here provides channel protection without the necessary of extra bandwidth. The proposed VLC decoder can be considered as an add-on module on the primitive structure. Therefore, it is compliant to the present video decoder.

To improve the error resilience, the soft VLC decoders with joint source and channel design have been proposed [23]-[25]. Such algorithms generally need to maintain many states when the table size grows. Hence, soft VLC decoders have problems of high complexity. Reduced complexity algorithms with sub-optimal solution have been made [24]. However, the improvement in [24] is not significant with larger VLC table. In this thesis, we propose a scalable soft VLC decoder (Scalable Soft VLD) to reduce the complexity. Firstly, our approach includes algorithm translation and table size reduction. To simplify the algorithm, we translate the metric derivation in Soft-Input Soft-Output algorithm [19] into the symbol-constrained directed graph (SCDG) for the soft VLC

decoding. Through the help of graph representation, we develop a modified sorting scheme that can achieve the same decoding performance with fewer states. Further, it can obtain the less number of memory accesses for the low-power demand. To reduce the table size, we proposed a symbol-merging algorithm. We merge two symbols with the same prefix into one symbol. By the symbol-merging algorithm, we can greatly reduce the table size as well as complexity at the cost of minor performance loss.

However, to deal with the different tables (intra and non-intra table) with different types of frame in MPEG standard, we propose a table merging method to integrate the different tables into one table. The proposed soft VLC decoder can employ this single merged-table and deal with the requirement of different VLC coding tables (i.e. intra or non-intra table) instead of duplicated configuration for the different VLC table. In summary, compared with [29][31], the proposed symbol-merging and table-merging algorithms achieve high capability of integration and flexibility.

In [16], the authors used the minimal Hamming distance ( $d_H$ ) to quantify the relation between table and performance. But, it is still inaccurate when the different tables reach the same  $d_H$ . We propose a novel measurement to improve the accuracy of performance estimation. Further, we reduce the penalty of over-design and observe the tendency of performance through the proposed Black-Box model. Thus, the proposed model reaches the optimal trade-off between performance and complexity.

The proposed scalable soft VLC decoder using performance modeling is verified with not only a simple table but also a practical MPEG-4 table. From the analysis of simple VLC source data, our algorithm can averagely save 15% of memory access in comparison with the state-of-the-art algorithms. Further, we can obtain the optimal parameters for a given table and decoding algorithm through the Black-Box model. Finally, our scheme shows more than 1dB PSNR improvement as compared with the straightforward table look-up decoding in AWGN or bursty channel.

In addition, the proposed scheme is also compared with different coding configuration such as the SSVLC [10] and RVLC [30]. Compared with the standard-support RVLC decoding method, our algorithm achieved more than 0.5dB improvement at the environment of SNR=10dB. Further, the VLC coding is more efficient than RVLC in terms of coding efficiency. There is not any side information to

be transmitted and the proposed decoder is bandwidth efficient.

## *1.4 Thesis Organization*

The rest of this thesis is organized as follows. Chapter 2 briefly introduces the SISO algorithm [19] and presents our proposed *adaptive* AMAP-2 for reducing the number of memory access in chapter 3. Chapter 4 shows our symbol-merging and table-merging method for complexity reduction. Chapter 5 describes the proposed Black-Box model for the optimal trade-off between performance and complexity. Chapter 6 presents the complexity and performance evaluation on MPEG-4. Finally, chapter 7 summarizes our work and discusses some topics for future research.



# Chapter 2

## Soft Decoding of Variable Length Code

### 2.1 *Background*

In the most image/video compression, VLCs decoding is considered as table look-up method and performed bit by bit. The input of entropy decoder assumed to be a sequence of “hard” bits that no soft information is available. However, soft information can be associated with each information bit in a noisy environment. It can be realized either on the channel observations in the case of un-coded transmission, or through soft-output channel decoders (e.g. SOVA or turbo coder) when channel coding is employed.

Based on the soft input of VLC decoder, many publications [24][31] proved that the performance improvement is noticeable than the traditional VLC decoders. Compared with the FEC and ARQ method, soft VLC decoder is bandwidth-efficient and channel-robust in the noisy environment. We choose the SISO/ML algorithm as the core algorithm of soft VLC decoder because of the implementation cost and real-time consideration. To apply the SISO/ML algorithm into the practical system (e.g. MPEG-4, H.264), there are some modifications required. We address the translation between the conventional SISO algorithm and the modified SISO on the following. Further, we modify the traditional source VLC table by introducing some symbol-information. After that, we can facilitate the system integration on the soft VLC decoder.

## 2.2 Soft-Input Soft-Output Algorithm

SISO decoding technique [19] is considered as an exhaustive decoding procedure to resist the error disturbance in the noisy channel. It estimates and searches on the tree-like path in the existence of additive white Gaussian noise (AWGN). The input sequence is a packet-based transmission through packetization. We don't exploit the soft output for the iterative decoding because of the consideration of the real-time video transmission. It uses  $L$  bits and equivalently  $N$  symbols to represent the priori information in one packet. Specifically, the SISO algorithm chooses the estimated sequence  $\mathbf{X}$  as the one that maximizes the joint probability for the observed sequence  $\mathbf{Y}$ . The estimated sequence that maximizes the joint probability  $Pr(X, Y)$  is indicated as  $X^* = \{x^*(1), x^*(2), \dots, x^*(N)\}$ . The optimal codewords can be developed as Equation 2.1, where the probability  $P^*$  is the sequence of codewords which maximize  $Pr(X, Y)$ . More detailed derivation and description have been shown in [19]. Based on the similar estimation, we perform the algorithm translation to simplify the SISO algorithm when the table size or decoded symbol grows.

$$\begin{aligned}
 x^*(N) &= \arg \max_{i, i \in [1, k]} \{P^*(N-1, L-l_i) \cdot \Pr\{y_{L-l_i+1}, y_{L-l_i+2}, \dots, y_L | x(N)=i\} \cdot p_i\}, \\
 X(1, L) &= \arg \max_{i, i \in [1, k]} \{\Pr\{y_1, y_2, \dots, y_L | x(1)=i\} \cdot p_i\}
 \end{aligned} \tag{2.1}$$

### 2.2.1 Algorithm Translation

To help the understanding of our simplified algorithm, we utilize a symbol-constrained directed graph representation [13][24] for the symbol-based VLC trellis decoding [14][15]. Figure 2.1 depicts the high-level description of the decoding procedure. The overall algorithm translation can be partitioned into two main parts. The one is the state-trellis construction. Because the SISO algorithm is an exhaustive search, it will result in the exponential growth of complexity with the increase of sequence length or table size. This state-trellis construction require the adder, shifter and multiplexer to perform the similar function of ACS unit in the Viterbi decoder. In addition, the other one is the trace-back decoding procedure. Firstly, it searches the best candidates

conforming to the matching criterion. This criterion is feed-forward from the packet header and provides the priori or soft information to the back-end VLC decoding procedure.

```

SoftVLD_Procedure ( )
{
    // Step1 : Initialization.
    for(j=0;j<LUT_size;j++)
    {
        for(L=0;L<VLC_CL;L++)
            // Step 1.1 : assign the intermediate metric of each state in the first symbol step.
    }

    // Step2 : Generating state trellis.
    for(i=0;i<N;i++)
    {
        while( search the minimal metric from the previous state)
        {
            // Step2.1 : [Add] – add the previous metric to form the present state metric.
            // Step2.2 : [Compare & Select] – compare with the other state metric to select
            the minimal one as the final candidate in present state.
        }
    }

    // Step3 : Trace back to decode symbols.
    while( search the final states(i.e. i==N-1))
    {
        if(state pointer==input size)
            // Step3.1 : label the start point in the trace-back process.
    }

    for(i==N-1;i>=0;i--)
    {
        // Step3.2 : Look-up the previous states of present state.
        // Step3.3 : decode each codeword and look-up the symbol-information.
    }
}

```

Figure 2.1 : High-level description of the decoding procedure with algorithm translation.

For the illustration of our algorithm translation, we use a simple example to address this translation. Firstly, assume we have a simple VLC table with only 3 symbols {0,10,11} and a packet that includes 3 bits (and equivalently 2 symbols) with content as ‘0 10’. After BPSK modulation, the modulated sequence is {-1,+1,-1}. When the packet is transmitted over the AWGN channel, the received packet may become {-0.8, -0.05, -0.2} (i.e. error occurred in the second bit).

Figure 2.2 depicts the graph representation for this example. The intermediate metric  $D^*(i,j)$  denotes the cumulative square error of  $i$ -th symbol and  $j$ -th bit in each symbol-state.  $S(m,n)$  is the symbol state decoded with  $m$ -symbol and have the index of

n among the identical value of m. The number inside each square is just the same as the ‘j’ of  $D^*(i,j)$ . The operation of ‘minimum’ is exercised in the states  $S(2,1)$ , which is entered by more than 2 arrows for the same states. Furthermore, the minimal metric after the comparison is survived and the others are pruned. There is no need to calculate the state metric  $D^*$  of  $S(2,3)$  and  $S(2,5)$ , and return the null value (i.e.  $\phi$ ) because the decoded bit pointer exceeds the priori bit information (i.e.  $4 > 3$  bits). Therefore, we can decide the shaded squares as the final candidates. The  $S(2,2)$  is the minimum among them, survives and traces back to the  $S(1,0)$  to decode the bitstream as  $\{0,10\}$  for the correct decoding.

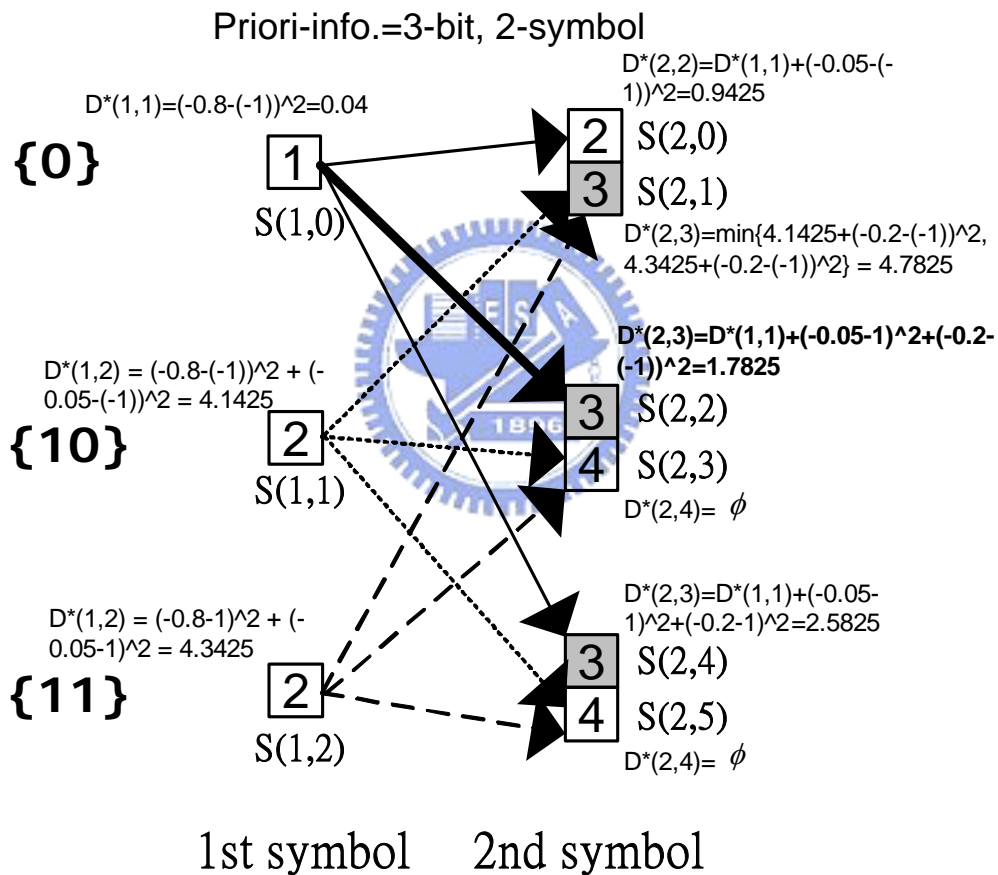


Figure 2.2 : The algorithm translation between symbol-constrained directed graph and the SISO algorithm.

## 2.2.2 Algorithm Modification

### Source Table Modification

To apply our algorithm to the MPEG-4 standard, we introduce the ‘sign’ and ‘LAST’



field from the original Huffman table. The extra fields of ‘sign’ and ‘LAST’ are essential for the decoding procedure of SISO in MPEG-4. In Figure 2.3(a), we modify the simple VLC table as Figure 2.3(b). In our proposed approach, we exploit the number of ‘LAST’ in one packet to represent the modified priori information. The number of ‘LAST’ in one packet is defined by MPEG-4 standard and extracted from the packet header.

To deal with the “s” parameter appended in each symbol, we use a simple hard decoding with table-look-up method. The induced ‘sign’ field in Figure 2.3(b) represents the number of “s” in each symbol. The ‘sign’ field is 1 when the “s” of each symbol is appended by 1-bit. More discussion about the ‘sign’ field is provided in the scalable soft VLC decoder of chapter 4.

Code length	Code word
1	1
3	011
4	0100
4	0101

(a) Simple VLC table

Code length	Code word	sign	LAST
2	1s	1	1
4	011s	1	0
5	0100s	1	0
5	0101s	1	0

(b) Simplified MPEG VLC table

Figure 2.3 : The original (a) and real case (b) of VLC table.

### Priori-Info. Modification

After the modification of VLC coding tables, it is crucial to modify the priori information since the original information cannot be extracted within the coding procedure for the practical application such as MPEG-4 or H.264. However, it is feasible to obtain the information on the number of blocks contained in the texture partition when decoding headers and motion partition. This information can easily be exploited by counting the number of occurrences of LAST field being equal to 1. Thus, the knowledge of the number of blocks can be considered as an “*a priori-information*” that can be used as the number of symbols to select a likely path. Further, it’s available to the user without requiring any side information to be transmitted.

This modification induces a little performance loss due to the additional candidates to

be selected in the Step3.1 of Figure 2.1. The difference with this modification can be described in Figure 2.4. The traditional soft VLC decoder [13][19] used the constraint of known symbol numbers as the algorithmic priori-information. However, this information should be transformed into the numbers of specified symbols. We can use the “EOB” symbol of MPEG-2 and “LAST” symbol of MPEG-4 (i.e. specified symbol) as the algorithmic constraint within the trace-back procedure. But, this modification induces the extra candidates from the start point to the end point with LAST number constraint in Figure 2.4. To achieve the standard-compliant and bandwidth-efficient design, this modification is essential and the induced performance loss is inevitable.

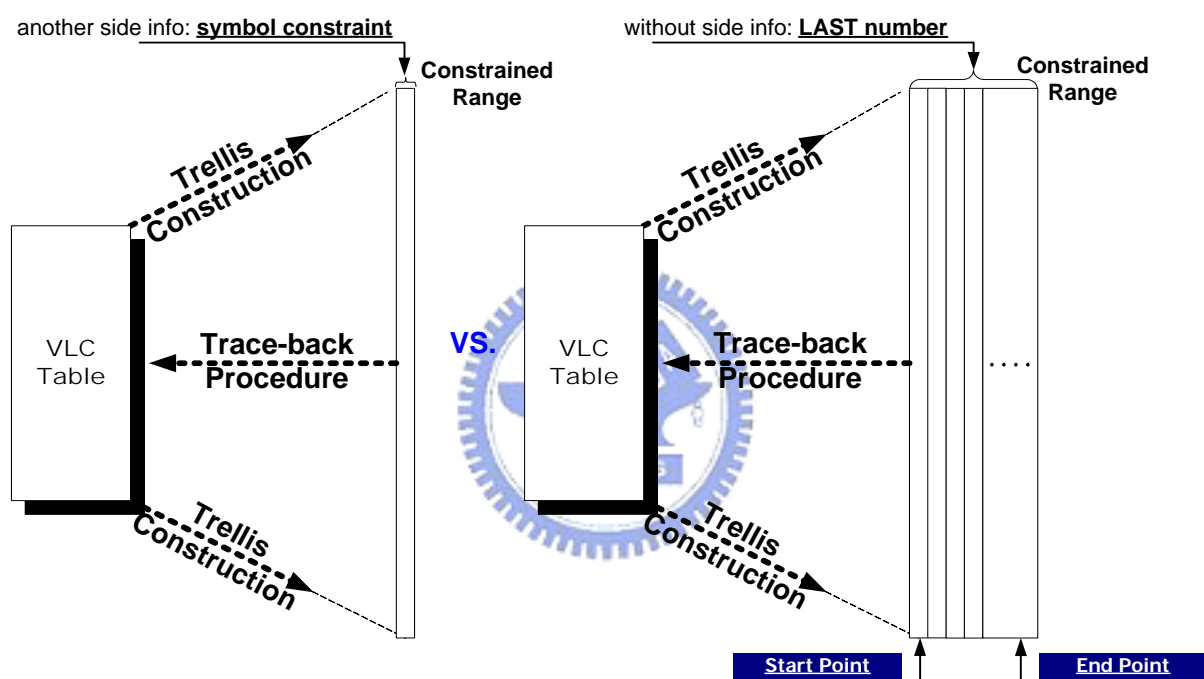


Figure 2.4 : The algorithm modification due to the constraint change.

In summary, based on the above algorithm modification, we show the modified high-level description in Figure 2.5. The modifications are labeled with shaded region. Firstly, we have to introduce the other term of “SIGN” to perform the metric calculation in Step1.2 and Step 2.1.1. This term is calculated by absolute difference and decoded with hard decoding scheme. Secondly, the constrained range (see Figure 2.4) has been extended and re-calculated in Step 3.1. Therefore, we can easily apply this SISO/ML soft VLC decoding algorithm into the practical VLC coding table such as the AC TCOEF tables in MPEG-2 or MPEG-4. More simulation and discussion will be

addressed on the following chapters.

```
SoftVLD_Procedure ( )
{
    // Step1 : Initialization.
    for(j=0;j<LUT_size;j++)
    {
        for(L=0;L<VLC_CL;L++)
            // Step 1.1 : assign the intermediate metric of each state in the first symbol step.

            // Step 1.2 : adding the extra sign bit into the formulation of metric.
    }

    // Step2 : Generating state trellis.
    for(i=0;i<N;i++)
    {
        while( search the minimal metric from the previous state)
        {
            // Step2.1 : [Add] – add the previous metric to form the present state metric.

            // Step 2.1.1 : adding the extra sign bit into the formulation of metric.

            // Step2.2 : [Compare & Select] – compare with the other state metric to
            select the minimal one as the final candidate in present state.
        }
    }

    // Step3 : Trace back to decode symbols.

    while( search the final states (i.e. LAST start point <= l <= LAST end point ))
    {
        if(state pointer==input size)
            // Step3.1 : label the start point in the trace-back process.
    }

    for(i==N-1;i>=0;i--)
    {
        // Step3.2 : Look-up the previous states of present state.
        // Step3.3 : decode each codeword and look-up the symbol-information.
    }
}
```

Figure 2.5 : High-level description of the decoding procedure with algorithm modification.

# Chapter 3

## Memory Efficient Design Approach

### 3.1 *Algorithm with Adaptive Selection*

The SISO algorithm requires many states since practical MPEG-4 tables have many entries. It becomes inadequate for the VLSI implementation when the number of survival states grows. To reduce the number of states as well as memory access, we propose an *adaptive* AMAP-2 (A-AMAP-2) to reduce the memory accesses.

#### 3.1.1 Modified Sorting Scheme

In [24], the author introduced the approximated decoding method 2 (AMAP-2) to improve the coding performance with low complexity. However, their approach is not robust to the variation of channel condition. They induced more states to retrieve the metric in the error-occurred region and increased the penalty to error-free region. They tried to find the fixed ‘M’ state in the sense of smaller state metric  $D^*$  and sorted among them in each symbol step.

To against the variation of channel condition, we propose to adaptively select the states and reduce the number of survival states. Our adaptive scheme is more robust to the channel observance and provides the variable states in each symbol step to select the best states. To address our improvement and differences as compared with the AMAP-2 [24], we use the simple VLC table in Figure 2.3(b) as an example. The corresponding graph representation is developed in Figure 3.1 (a). To clearly show the metric variation in each state, we just omit the arrows and the indication of ‘LAST’. In Figure 3.1 (b), we show the sorting algorithm via the number of states in AMAP-2. By pruning the

square of the same bit-position in Figure 3.1(b), we obtain Figure 3.1(c) that can be used in comparison with our proposed A-AMAP-2.

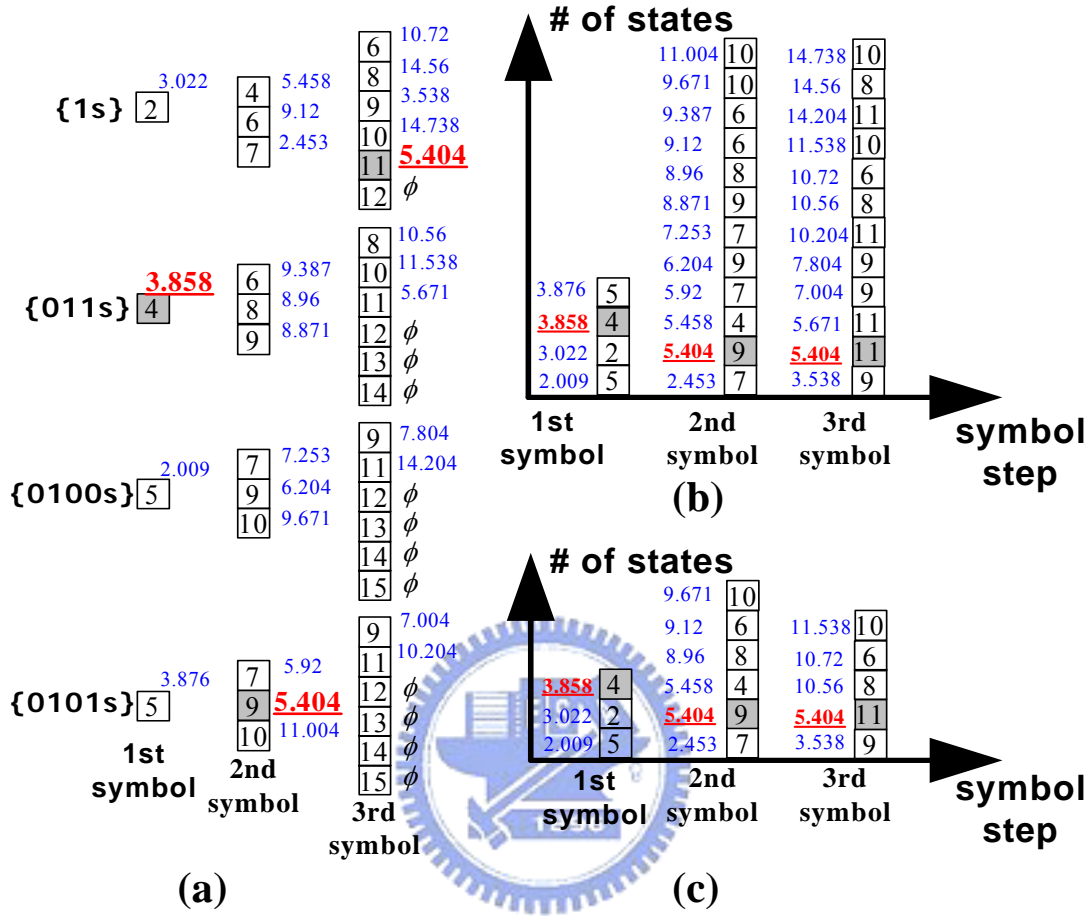


Figure 3.1 : The graph representation in approximated decoding 2 [24].

From Figure 3.2, we can see that the main difference of AMAP-2 and A-AMAP-2 is the sorting scheme in the Y-axis. Figure 3.2(a) shows that AMAP-2 requires at least 3 (i.e.  $M_{AMAP-2}$ ) states for correct decoding given the specified threshold. The correct states are labeled with the shaded region. In Figure 3.2(b), by employing the  $D^*$  in the sorting algorithm instead of the number of states, the state metric range above the minimal metric for the correct decoding is 4 (i.e.  $M_{A-AMAP-2}=6-2$ ). As a result, we can find that there are 9-state and 8-state survived in AMAP-2 and A-AMAP-2 respectively for the correct decoding. Such improvement on the state number reduction increases when the errors occur infrequently. More simulation results are provided in Figure 3.3. This novel scheme adaptively selects the number of survived states in each symbol step,

and that's why we call it as the *Adaptive* AMAP-2 (i.e. A-AMAP-2).

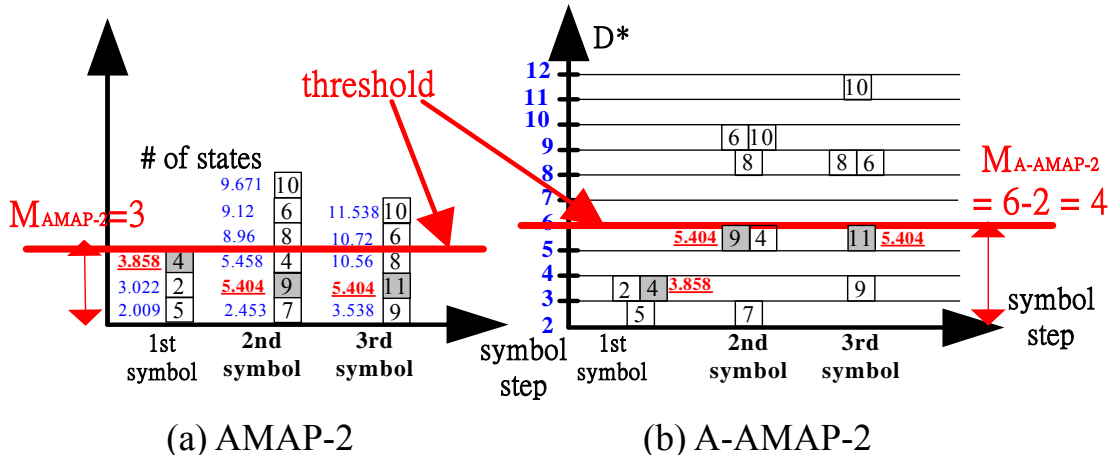


Figure 3.2 : The comparison between the AMAP-2 (a) and the proposed *Adaptive* AMAP-2 (b).

In Figure 3.2(b), the propose A-AMAP-2 survives more states when the errors are occurred frequently, such as 1<sup>st</sup> symbol step. Furthermore, fewer states are survived in the less error region, such as 3<sup>rd</sup> symbol step. It is more robust to the channel observance and provides the variable states to be survived in each symbol step.

### 3.1.2 Performance Comparison

The proposed A-AMAP-2 adaptively selects the number of states in each symbol step and reduces the number of memory access. The variable best state selection is presented. Many states are survived when the error occurred, and fewer states are survived in the error-free region. In Figure 3.3, we assume that one survived state will cost one access of memory element. In addition, we choose a specified threshold for AMAP-2 and A-AMAP-2 individually. This specified threshold could be optimized and decided by the proposed B-B model in chapter 5. After that, we choose an optimal threshold as the simulated parameter, and compare the performance of our proposed A-AMAP-2 and the AMAP-2 versus channel condition. Obviously, given the same performance, our algorithm occupies less memory space and memory accesses in high SNR. Averagely, our algorithm saves 15% of memory access as compared with AMAP-2 [24].

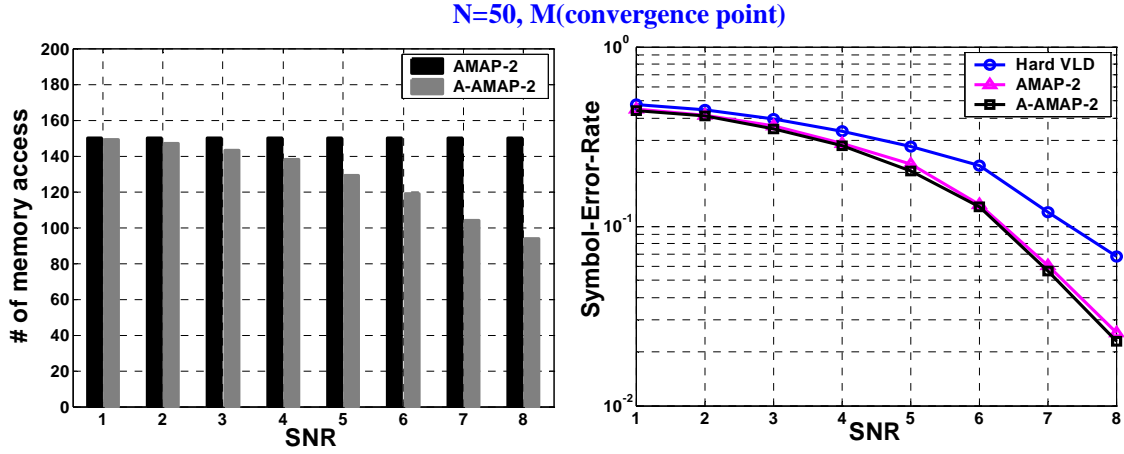


Figure 3.3 : (a) The comparison of performance (a) and memory access (b) vs. SNR.

### 3.2 Complexity Analysis

With the soft decoding of variable length code, comparators, multiplexers and storage elements are essential for the VLSI implementation. However, each storage elements (i.e. each state) also require corresponding modules inclusive of adder, multiplexer and shifter. Therefore, reducing number of storage element reduces not only the implementation cost but also the memory access times for the power-saving demand.

To formulate the complexity issue on the soft VLC decoder, we introduce some parameters to analyze the overall complexity in terms of the numbers of states. Since the number of states will grow with the sequence length and the number of code-length-type in the VLC coding tables. We introduce the total numbers of codeword  $T = \{CW_0, CW_1, \dots, CW_T\}$  and the number of code-length-type  $S = \{CL_0, CL_1, \dots, CL_S\}$  in the pre-defined VLC coding table. Moreover, the symbol number  $N$  is the received symbol constraint. If this constraint  $N$  cannot be noted before the coding procedure, we can use the decoded number of specified symbols (i.e. # of LASTs) instead for the practical application. The “*optimal*” soft VLC decoding, which means no any states are reduced has the best performance at the price the high complexity and high memory access. The number of states in optimal soft VLC decoder is depicted in Equation 3.1.

$$Optimal : States\_per\_Stage(N) = (CL_S - CL_0) \times (N - 1) \times T, N > 1 \quad (3.1)$$

$$AMAP - 1 : States\_per\_Stage(N) = (CL_s - CL_0) \times (N - 1), N > 1 \quad (3.2)$$

$$AMAP - 2 : States\_per\_Stage(N) = b_{max} \times N \quad (3.3)$$

Due to the implementation of large VLC table, we will pay more attention on the complexity formulation of sub-optimal solution in soft VLC decoder. In [24], authors presented **AMAP-1** to reduce the state numbers. The performance of AMAP-1 is almost the same with the optimal soft decoding method, since the pruning algorithm won't affect the optimal sequence selection in the trace-back decoding procedure. However, from the Equation 3.2, the number of states in AMAP-1 is still too large to implement in large VLC table. Another sub-optimal solution in [24] is **AMAP-2** that keeps the  $b_{max}$  best states at each trellis symbol step, and the formulation is described in Equation 3.3. The above equations are assumed that the code length of input source table are continuous and then approximated by the proposed formation in Equation (3.1) ~ (3.3). We show the example in Figure 3.4 to address the complexity of optimal decoding algorithm.

After the analysis of state numbers in each symbol step, we address the total numbers of states required in the soft decoding procedure. In Figure 3.5, the state numbers of optimal decoding and AMAP-1 decoding method are dramatically increased with the received symbol number (or received sequence length)  $N$ . However, the sequence length decided by the system-level controller or the packet size for the realistic application. From the algorithmic point of view, parameters  $T$  and  $S$  affect the increased degree of algorithmic complexity. These parameters are decided by the pre-defined VLC coding table. Therefore, reducing the number of entries in coding table or the number of tables can greatly reduce the numbers of states as well as the overall complexity. AMAP-2 is a sub-optimal and low-complexity solution for the realization of soft VLC decoder. But the reduced-complexity is still not enough when the table size or sequence length grows. We will focus on the reduction of table size in the next chapter.



Codeword	CodeLength
1s	2
011s	4
0100s	5
0101s	5

T = {1s, 011s, 0100s, 0101s}

S = {2, 4, 5}

Symbol Step N	1	2	3	4	5
Optimal => (5-2)x(N-1)x4, N>1					
# of state in each symbol step	4	12	24	36	48

Codeword	CodeLength
0s	2
10s	3
11s	3

T = {0s, 10s, 11s}

S = {2, 3}

Symbol Step N	1	2	3	4	5
Optimal => (3-2)x(N-1)x3					
# of state in each symbol step	3	6	9	12	15

(a) Example 1

(b) Example 2

Figure 3.4 : Complexity analysis in terms of each symbol state numbers.

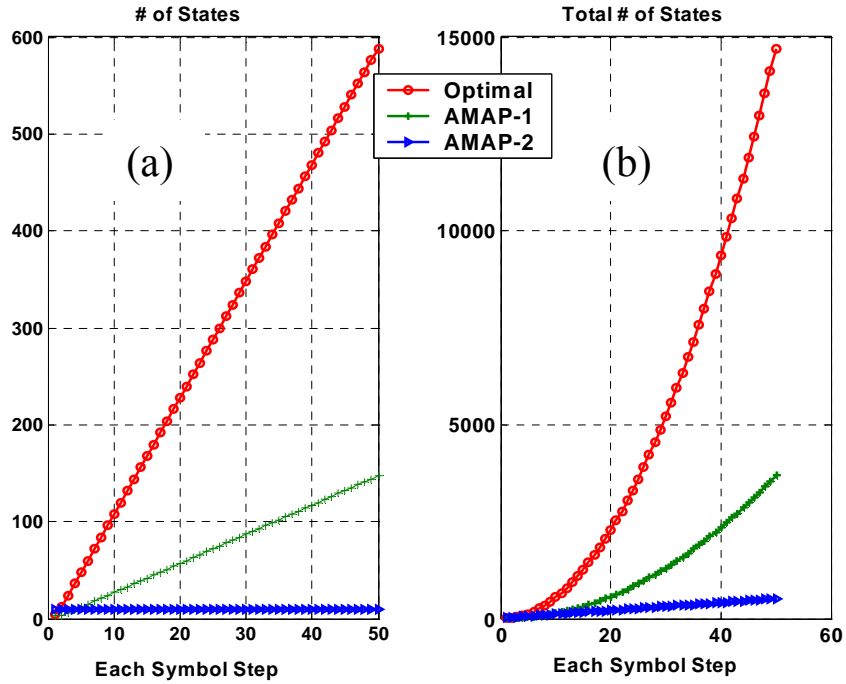


Figure 3.5 : The comparison with complexity issue in terms of state numbers (a) and total state numbers (b) using the VLC table of Figure 2.3.

### 3.3 Summary

In this chapter, the memory-efficient algorithm and complexity analysis of *adaptive soft VLC decoder* has been presented. Based on the modified sorting scheme, the proposed *Adaptive AMAP-2* becomes more channel-robust than traditional AMAP-2. Our proposed algorithm averagely saves 15% of memory access at the condition of identical coding performance. Further, we introduce some parameters to analyze the overall complexity in terms of state numbers. The advanced analysis and formulation of performance are described in chapter 5.

# Chapter 4

## Low-Complexity Design Approach

### 4.1 *Symbol-Merging Algorithm*

The main problem of soft VLC decoding is the many states and the complicated metric computation when the sequence length or table size grows. To apply the SISO algorithm to the MPEG-4 system, it is essential to reduce the table size. Thus, we propose a scalable scheme with symbol merging algorithm.

We utilize the redundancy exhibiting in different symbols to perform the merging algorithm. We consider a simple VLC table as a tree-structure in Figure 4.1(a). The proposed symbol-merging scheme searches the symbols with identical prefix and merges them into single merged-symbol. In Figure 4.1(b), the original SISO decoding algorithm is a special case that is when  $z$  is equal to 0 (i.e. Base  $T_0$ ). In other words, there is no hard decoding performed except 'sign' bit. Such case achieves the highest performance with the penalty of the largest complexity. However, the code-length of prefix symbol with soft decoding will decrease when the index ' $z$ ' increases. Meanwhile, the number of bits with hard decoding will increase. As a result, it can be considered as a hybrid scheme that combines the hard decoding and the soft decoding.

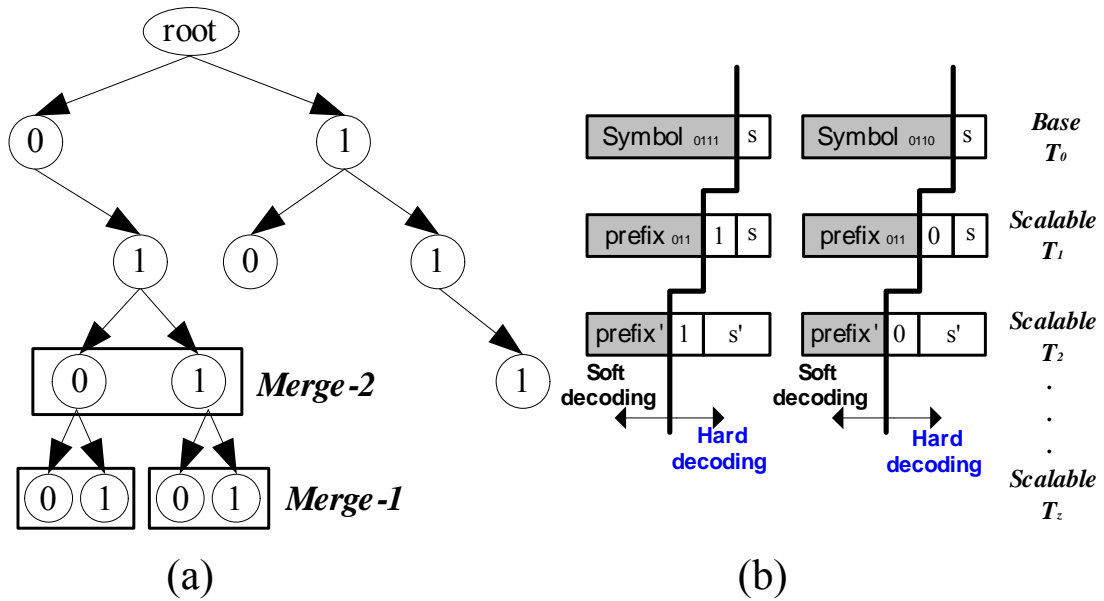


Figure 4.1 : The tree-structured VLC (a) and scalable scheme with hard and soft decoding (b).

The symbol-merging scheme can be operated only on a certain specified condition. Two codeword symbols can be merged only on the same symbol information including the identical “LAST” and “SIGN” field. In addition, these two codeword symbols have to own the equivalent prefix code and only different on the one-bit suffix code. The detailed high-level description has been shown in Appendix A. The AC TCOEF tables in MPEG-2 and MPEG-4 have been reduced to a reasonable size after the symbol-merging scheme. In addition, the merging conditions are also related to the symbol-information of “SIGN” and “LAST”, that’s why there are different merging result on MPEG-4 intra and non-intra table with all the same codeword (see Table 4.1).

We use a simple example to illustrate the proposed scheme in Figure 4.2 where ‘ $T_i$ ’ represents the number of symbols after the operation of *Merge- $i$* . As shown, after the operation of ‘Merge-1’, the table size is decreased by 2. Further, with the ‘Merge-2’ operation, the total number of symbols becomes 3. The introduced ‘sign’ field represents the number of “s” appended in the corresponding symbol. The ‘sign’ field will increase when both of symbols with the identical “SIGN” and “LAST” have been merged into one.

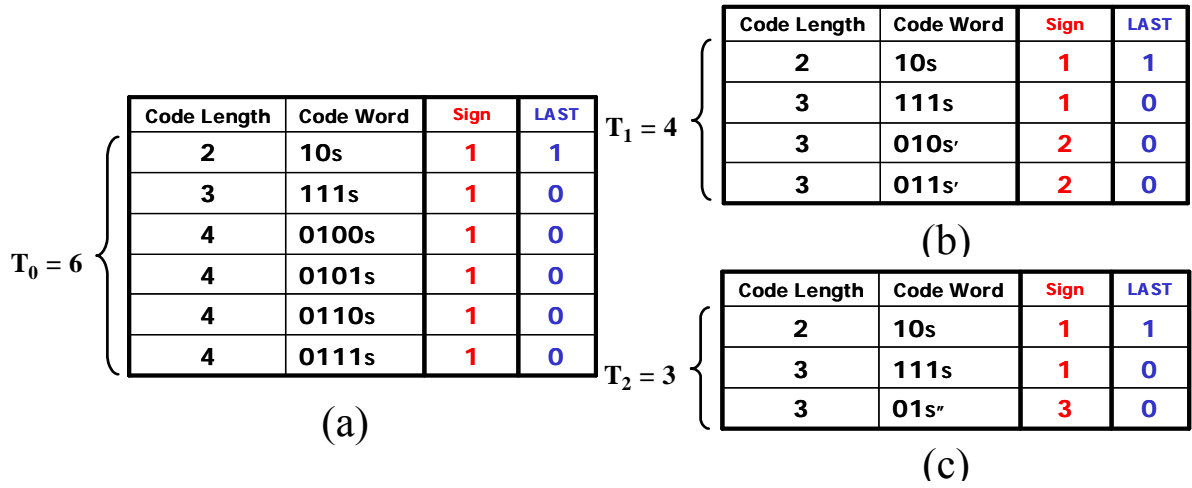


Figure 4.2 : A VLC table with Merge-0 (a), Merge-1(b) and Merge-2(c) operation.

#### 4.1.1 Metric Formulation of “Balance Degree”

It can be noted that the more the merged-symbol have been developed, the great the merging-efficiency can be achieved. Therefore, to quantify the number of symbols after the symbol-merging scheme, we propose the metric of ‘Balance Degree’ (B.D.) in Equation 4.1. The metric of B.D. is between 0 and 1. In Equation 4.1, the denominator represents the maximal value as well as a special table with complete tree-structure. It leads to “ $z \times 0.5$ ” after the  $z$  times of summation where the ratio of  $T_{i+1}$  over  $T_i$  is fixed at 0.5. Therefore, the branch degree of Figure 4.2 is 58% in the condition of “ $z=2$ ”.

To prove that B.D. is a meaningful number to our merging scheme, we measure the B.D. using the AC TCOEF tables in MPEG-2 and MPEG-4. As shown in Table 4.1, we find that the higher of the B.D., the more reduction of the table size. The B.D. value of non-intra table is larger than that of intra one. It can be explained by the fact that there is more redundancy exploited in terms of symbol-structure. That is to say the non-intra table is more efficient than intra table after performing the symbol-merging scheme.

$$B.D.(z) = \frac{\text{real reduction}}{\text{complete reduction}} = \frac{\sum_{i=0}^{z-1} \left(1 - \frac{T_{i+1}}{T_i}\right)}{\sum_{i=0}^{z-1} (1 - 0.5)} = \frac{\sum_{i=0}^{z-1} \left(1 - \frac{T_{i+1}}{T_i}\right)}{z \times 0.5} \quad (4.1)$$

Table 4.1 : The reduction of table size by symbol-merging algorithm.

<b>Standard</b>	<b>MPEG-2</b>		<b>MPEG-4</b>	
<b>Table</b>	INTRA <sub>TB-15</sub>	NON-INTRA <sub>TB-14</sub>	INTRA <sub>TB-15</sub>	NON-INTRA <sub>TB-14</sub>
<b>T<sub>0</sub></b>	113	114	103	103
<b>Scalable T<sub>1</sub></b>	65	60	61	56
<b>Scalable T<sub>2</sub></b>	45	34	48	38
<b>B.D.(2)</b>	73.2%	90.7%	62%	77.8%

## 4.2 Table-Merging Algorithm

It is essential for switching tables on the decoding process of soft VLC decoder, since there are intra and non-intra AC coefficient in the AC partition of whole bit-stream. Further, table-merging method is demanded on the fast switching capability of VLC decoder, such as the context-adaptive VLC in H.264. Consequently, to share the same soft VLC decoder on the different VLC table, we propose a novel soft VLC decoder with table merging algorithm to reduce the implementation cost and memory accesses.

We propose codeword merging and prefix merging method to realize the Table-Merging scheme. These merging methods are a lossless merging and harmless to the performance of soft VLC decoder; while the symbol merging algorithm in section 4.1 is a lossy merging scheme, since the performance of decoder will degrade with the number of merging (see Figure 4.3). We show the more detailed high-level description in Appendix B, and elaborate the merging algorithm in the following literature.

### 4.2.1 Code-Word Merging

Although most VLC coding tables are generated based on the Huffman procedure, one codeword still has high probability to exist in many coding tables. If this case is occurred, it is unnecessary to duplicate the codeword information in memories for every

table that uses this codeword. A codeword merging is applied to set this codeword as a merged codeword and reuse the codeword information when the coding tables are required. Therefore, the information redundancy among coding tables is exploited. The stored data are reduced from many identical codewords to one merged codeword.

#### 4.2.2 Prefix Merging

According to the Huffman property, one codeword cannot be the prefix of another codeword in a table but this rule does not hold among different tables. Frequently, a short codeword in one table will be the prefix of a long codeword in other tables. When these codewords are found, a prefix merging is performed by storing the long codeword as a merged VLC codeword and the lengths of the VLC codewords in each table. As a result, the information redundancy among tables is further exploited.

#### 4.2.3 Merged Table

A table merging process is accomplished by applying both codeword-merging and prefix-merging to the codewords of all AC TCOEF tables. The required table information, which is to recover VLC coding tables from merged table, has to be a superset of the stored data of two merging methods since it is hard to distinguish which method is used to generate a merged codeword. Hence, every VLC code-length of all tables has to be stored individually and will not be reused even though codeword merging is performed. To select the merged codewords of VLC table quickly, additional information, a valid-bit, is utilized to indicate whether a merged codeword belongs to the table. Thus, the table information of a coding table is the valid-bit and VLC code-length of every merged codeword (see Appendix B). The overall memory requirement is reduced because merged codewords are stored once and reused by all AC TCOEF tables.

### 4.3 *Performance Evaluation*

We propose the symbol-merging method to reduce the complexity at the expense of

little performance loss. There are tradeoffs between the complexity reduction and performance loss. In Figure 4.3, the complexity in terms of execution time reduces greatly at the cost of little performance degradation. Figure 4.3(b) describes that the performance loss will dominate the overall system performance (i.e. symbol error rate) when ‘ $i$ ’ is larger than 2 (i.e.  $Merge-i > Merge-2$ ).

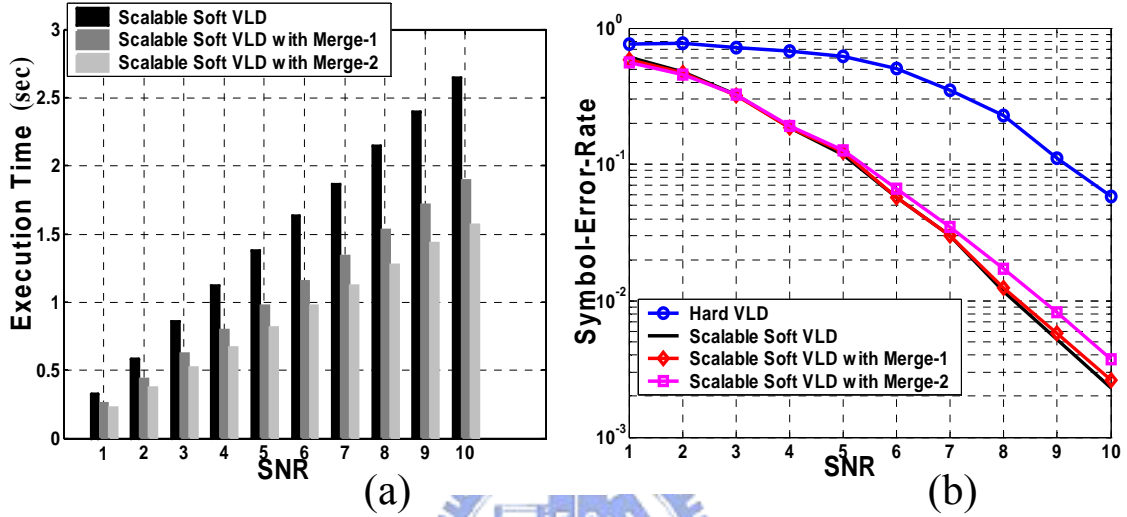


Figure 4.3 : The evaluation of execution time (a) and performance (b) with different symbol-merging table in Figure 4.2.

To improve the flexibility of soft VLC decoder with the different AC TCOEF coding tables (i.e. intra and non-intra), we perform Table-Merging scheme to reduce the implementation cost and computational complexity. In the table configuration of soft VLC decoder (see Table 4.2), [31] uses two soft VLC decoders with MAP decoding operating on intra and non-intra blocks respectively. It's not intuitive for the hardware implementation and system integration. It may require additional information to partition the intra and non-intra blocks into different channels. The integration overhead and implementation cost made it unreliable for the cost-effect design approach. In [29], the authors implement a soft VLC decoder with sequential algorithm. It used single-like soft VLC decoder to reach the different VLC table requirement. However, the entries of AC TCOEF tables are extensive and induce unexpected memory access and computational complexity. To resolve the problems of complexity, we propose a novel merging scheme to reduce the table size and merge the different tables into one table.

Based on our proposed soft VLC decoder, a comparison with existing designs is given in Table 4.2. We implement the soft VLC decoder with SISO/ML algorithm. However, due to the different anchor configurations and source characteristics among them, we additionally list “*Improved Ratio*” (Equation 4.2) to declare the performance relation of upper bound (i.e. no error), soft VLC decoder, and anchors.

More discussion about “improved ratio” can be addressed in Figure 4.4. In general, it can be noted that soft VLD has an improvements of x dB as compared with the anchor. However, the value of x is an absolute-local metric since this metric may vary with different source (e.g. bit rate) and channel (e.g. channel condition) environment. To achieve a fair comparison, we propose a measurement of “improved ratio” to equalize among them. We consider the performance not only the lower bound (i.e. anchor) but also the upper bound (i.e. no error) to obtain the ratio among them. Based on the induced “improvement ratio”, Table 4.2 depicts about 80% capability of error recovery in our proposed design can be achieved. Finally, we propose a low complexity soft VLC decoder to realize the large VLC table in the MPEG standard at the expense of minor performance loss.

$$\text{Improved Ratio} = \frac{\text{Perf}(\text{Soft VLC Decoder}) - \text{Perf}(\text{Anchor})}{\text{Perf}(\text{No Error}) - \text{Perf}(\text{Anchor})} \quad (4.2)$$

#### 4.4 Summary

In this chapter, the algorithm and system implementation of *scalable soft VLC decoder* with a novel symbol-merging and multi-table-merging approach have been presented. Based on the symbol-merging algorithm, we can greatly reduce the table size with the price of minor performance loss. Further, to improve the table configuration on the decoding process of switching table, we present a table-merging scheme to improve the efficiency of soft VLC decoder when operating on the multiple tables. For the practical applications, an efficient and low-complexity soft VLC decoder is fulfilled on the joint source and channel design.



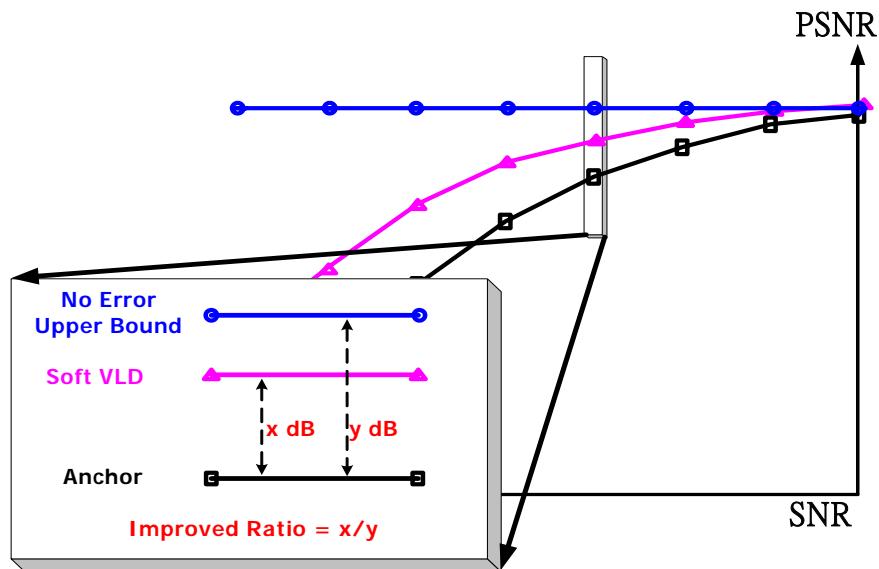


Figure 4.4 : The formulation of “Improved Ratio”

Table 4.2 : The comparison with existing design.

<i>Soft VLC Decoder</i>		<i>Proposed</i>	<i>[29]</i>	<i>[31]</i>
<i>Implementation Method</i>		MPEG-4+SISO/ML	MPEG-4+Sequential	MPEG-4+MAP
<i>Table Configuration</i>		Reduced-Single	Single	Separated
<i>Anchor</i>	<i>RM</i> <sup>1</sup>	Enable	Enable	Enable
	<i>DP</i> <sup>2</sup>	Enable	N/A	N/A
	<i>EC</i> <sup>3</sup>	Disable	Enable	Disable
<i>Source Characteristics</i>		Foreman, QCIF, 64kbps, I-P-P, 300bits/packet	Foreman, CIF, 800kbps, I-P-P, 4000bits/packet	Foreman, 0.164bits/pel, QCIF, I-P-P
<i>Testing Environment</i>		AWGN+BPSK	AWGN+BPSK	AMC <sup>4</sup>
<i>Improvement</i>		1.2dB	8dB	6dB
<i>Improved Ratio</i>		79.28%	80%	52.72%

<sup>1</sup> Resynchronization Markers.    <sup>2</sup> Data Partition.    <sup>3</sup> Error Concealment

<sup>4</sup> Additive-Markov-Channel model for slow fading wireless channel.

# Chapter 5

## Performance Modeling

### 5.1 *Black-Box Model*

To optimize our proposed scalable soft VLC decoder in chapter 4, it is crucial to reach the optimal trade-off between performance and complexity. We propose a Black-Box model (i.e. B-B. Model) to formulate the performance and introduce some parameters to describe the complexity. They are independent and composed of algorithm-sensitive, application-sensitive and table-sensitive. In the table-sensitive, we propose a novel measurement of “symbol-alias” to provide accurate performance estimation for the different tables. Finally, the proposed Black-Box model can reach the optimal parameters for a given table and decoding algorithm. Figure 5.1(a) depicts the proposed Black-Box (i.e. B-B) for the performance modeling and uses Figure 5.1(b) as the source VLC table for the following illustration.

In Figure 5.2, the proposed model can be viewed as a parameterized decoder, which is formulated and configured by some significant parameters. From the previous statements, the performance of proposed soft VLC decoder can be parameterized by three factors and elaborated in the following description.

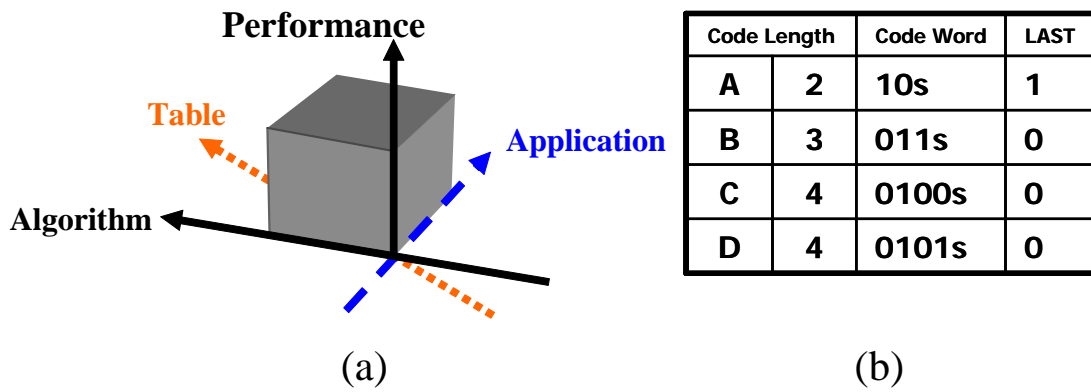


Figure 5.1 : The B-B model (a) and the evaluation of source table (b).

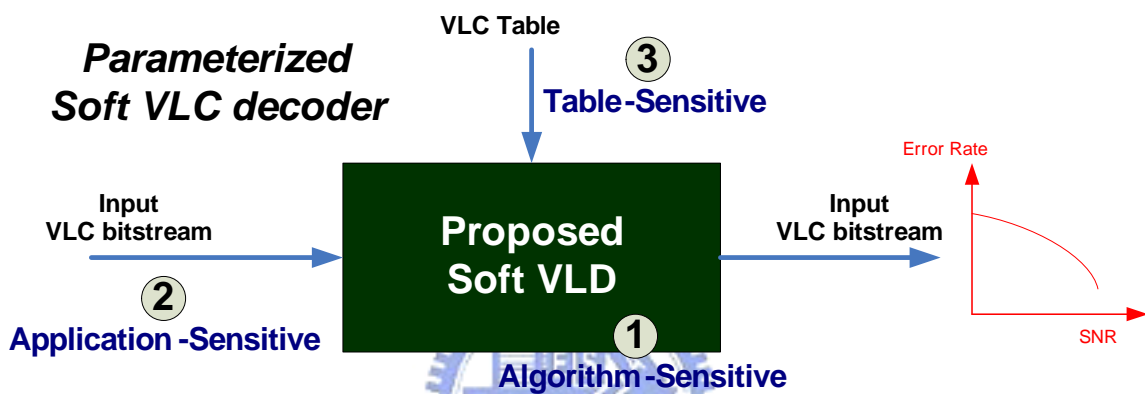


Figure 5.2 : The relationship between performance and each parameter.

### 5.1.1 Algorithm-Sensitive Parameters

The algorithm-sensitive parameters are sensitive to algorithms; that is to say, different algorithms are characterized by different parameters. Using the proposed algorithm in chapter 4 as an example, the parameter ‘z’ is considered as an essential factor to approach the trade-off between performance and complexity. Figure 5.3 describes that the optimal choice is achieved when z is equal to 1 (Merge-1). The parameter ‘N\*’ (see chapter 5.1.2) does not affect the final results. Thus, “Merge-1” obtains the maximal reduction of complexity at the cost of minor performance loss.

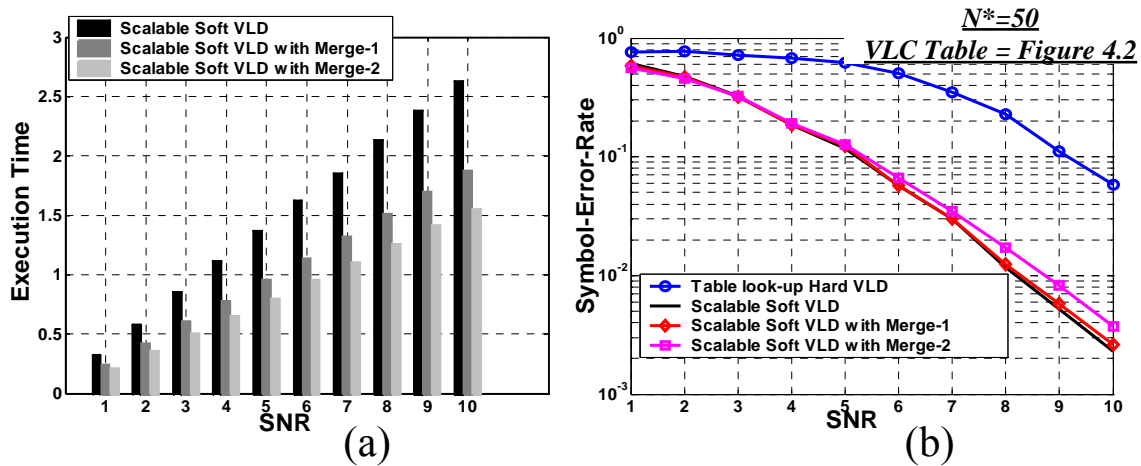


Figure 5.3 : The complexity (a) and performance (b) in different ‘z’.

After the analysis of parameter ‘z’, we will focus on the A-AMAP-2. From the analysis of performance and complexity in chapter 3, it’s more imperative to decide the number of ‘M’ to be survived (see Figure 3.2). The larger M achieves the higher performance at the price of high complexity. The problem occurs also in the smaller M. Thus, inappropriate M will be harmful to the performance or complexity.

The empirical value of M is determined from experiments. We define the saturation and convergence point to approach the optimal value. Given a simple table in Figure 4.2(a), Figure 5.4 depicts the measurement of ‘saturation M’ and ‘convergence M’. The symbol error rate will decrease with the increasing ‘M’. Intuitively, we select the convergence point as ‘M’ for the tradeoffs between performance and complexity. We also use the convergence point in the verification of MPEG-4 standard.

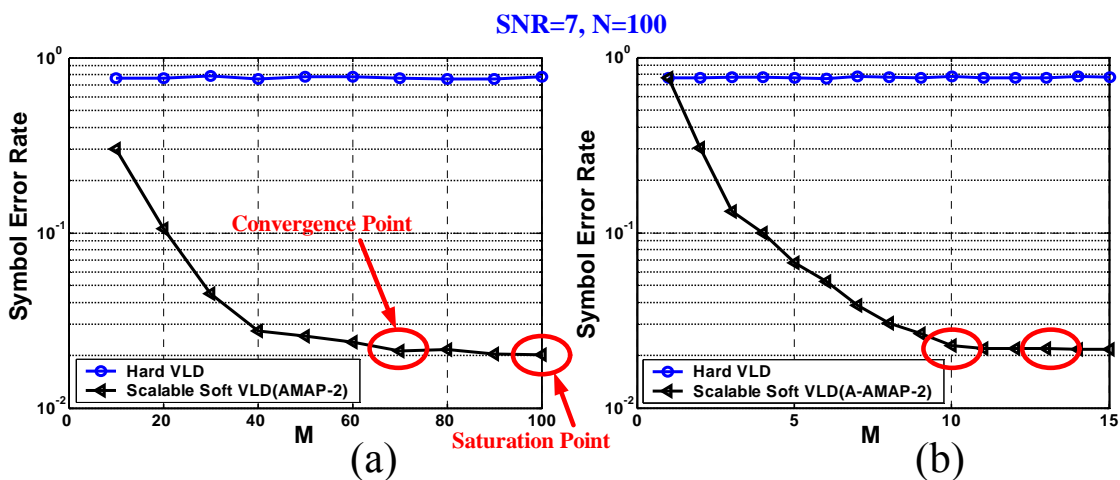


Figure 5.4 : The performance with convergence and saturation point in AMAP-2 (a) and A-AMAP-2 (b).

## 5.1.2 Application-Sensitive Parameters

The application-sensitive parameters are unrelated to the algorithms and decided by the extrinsic applications. The packet size ‘N’ is the most impressive factor to achieve the optimal performance. In this section, we regard it as the decoded number of symbols ‘N\*’ for the simplification. Figure 5.5(a) describes that the performance can be expressed by the normalized symbol error rate (i.e. SER) and overhead. In the overhead computation, we assume that the 15-bit resynchronization marker is inserted in the period of N-symbol. From Figure 5.5(b), it achieves the minimal SER and overhead when ‘N\*’ is equal to 60. There is no need to consider the algorithmic complexity because it has to be optimized from the algorithm-sensitive parameters.

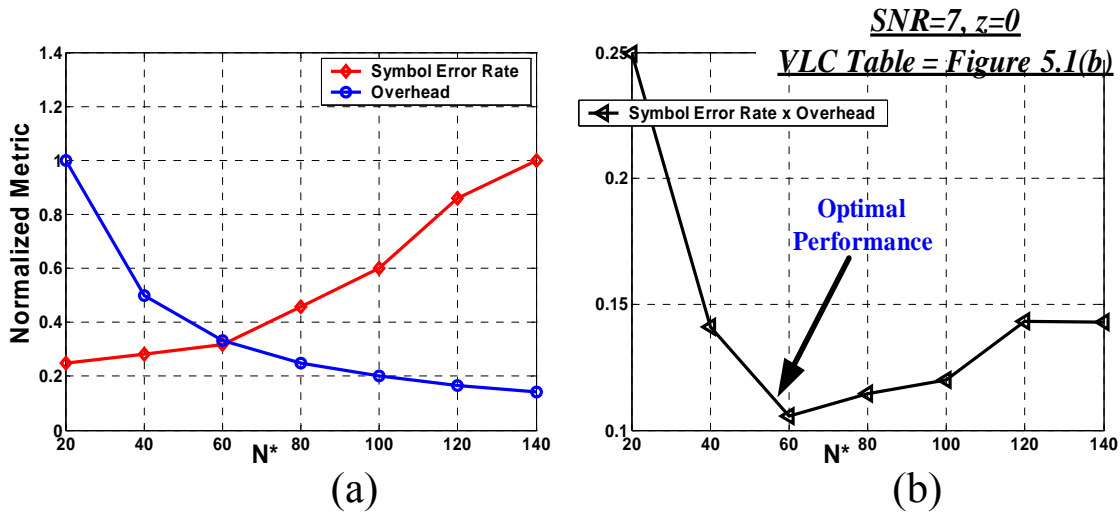


Figure 5.5 : The optimization of performance in different ‘N\*’.

## 5.1.3 Table-Sensitive Parameters

To estimate the performance with different tables, the authors in [16] used the minimal Hamming distance ( $d_H$ ) to quantify the relation between VLC table and performance. It is still inaccurate when the different tables reach the same  $d_H$ . We propose a novel measurement of ‘symbol alias’ to quantify their relation and provide more accurate performance estimation when their Hamming distances are the same.

### 5.1.3.1 Intra Alias

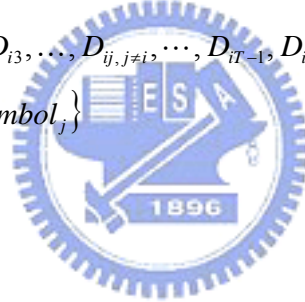
We introduce the table-sensitive parameters ‘T’ and the symbol alias to obtain more accurate estimation. The parameter ‘T’ (i.e.  $T_i$  in Figure 4.2) denotes the number of symbols for a given VLC table. Besides, the symbol alias comprises two components. One is the distance of “Inter Alias” (i.e.  $Dist_{inter}$ ) described by Equation 5.1. The other is the distance of “Intra Alias” (i.e.  $Dist_{intra}$ ) described by Equation 5.2. Figure 5.6 depicts the results of the following equations. The  $1/Dist_{inter}$  is the number of elements for the specific set, which calculates that whether the code-length of one symbol is the combination of the others. The  $Dist_{intra}$  is the summation of minimal Hamming distance for each symbol.

$$Set_{inter} = \left\{ \{x_1, x_2, x_3, \dots, x_T\} \mid \sum_{i=1, x_i \neq \max\{x_i\}}^T x_i = \max_i x_i \right\} \quad (5.1)$$

$$Dist_{inter} = \frac{1}{NumOfElement(Set_{inter})}$$

$$Dist_{intra} = \sum_{i=1}^T \min(D_{i1}, D_{i2}, D_{i3}, \dots, D_{ij, j \neq i}, \dots, D_{iT-1}, D_{iT}) \quad (5.2)$$

where  $D_{ij} = d_H\{symbol_i, symbol_j\}$



### 5.1.3.2 Inter Alias

The inter alias is more sensitive to ‘T’ than intra alias and induces more performance loss. In Figure 5.7(a), the intra alias of TB-I results from the bit alias with the symbols of identical code-length. The increase of ‘T’ provides the increase of SER in the soft decoding. However, the SER of table look-up decoding decreases when the ‘T’ grows. The reason is that the extra symbols prevent the decoded-symbol loss and error propagation. Further, the inter alias of TB-II results from the code-length alias with the symbols of different code-length. In Figure 5.7(b), both of SER increase and provide more performance loss than intra alias.

Code Length	Code Word	
A	2	10
B	3	011
C	4	0100
D	4	0101

(a) Simple table in Figure 5.1(b)

Code Length	Code Word	
A	$x_1$	symbol <sub>1</sub>
B	$x_2$	symbol <sub>2</sub>
C	$x_3$	symbol <sub>3</sub>
D	$x_4$	symbol <sub>4</sub>

(b) Source table in Eq.(5.1)-(5.2)

$$1/Dist_{inter} = 2$$

$$\rightarrow \{D, A, A\} \text{ or } \{C, A, A\} \quad (\text{i.e. } 4=2+2)$$

$$Dist_{intra} = 5$$

$$\rightarrow 2 + 1 + 1 + 1$$

Figure 5.6 : The symbol alias of VLC table (a)(b).

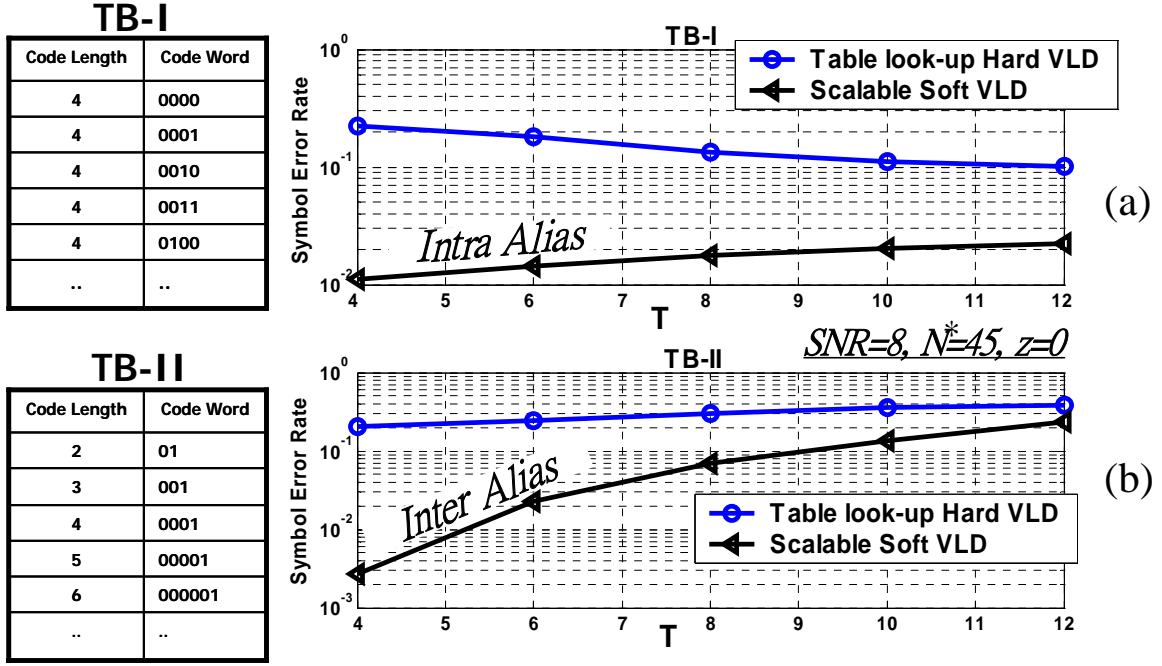


Figure 5.7: The performance evaluation of ‘intra alias’ (a) and ‘inter alias’ (b) in different ‘T’.

### 5.3 Performance Estimation

To prove that the proposed symbol alias is a meaningful number, we use the VLC table in [16] to recognize the difference of performance. In Figure 5.8(a), we inverse the underlined bits in C2c and C2d for the correct VLC decoding. For our proposed measurement of Figure 5.8(b), the Codes C2a achieves the worst performance because of the lowest  $Dist_{inter}$  that induces more performance loss than  $Dist_{intra}$ . Further, there is no inter alias exist in C2c and C2d. The  $Dist_{inter}$  is fixed at infinite ( $\infty$ ) by default. The C2c obtains higher performance than C2d because of the higher  $Dist_{intra}$  of C2c. The relation of performance with different tables is identical to [16] (see Figure 5.9). In Figure 5.8(b), the higher distance of coding table will lead to the less occurrence of symbol-alias (i.e. higher performance). The performance of C2c and C2d cannot be recognized in [16] when each symbol probability is unknown. We provide more accurate estimation than [16] and reduce the design time for the performance evaluation among different tables.

Performance(C2c) > Performance(C2d) > Performance(C2a)

<i>Codes C2a</i>			<i>Codes C2c</i>			<i>Codes C2d</i>		
Code Length	Code Word		Code Length	Code Word		Code Length	Code Word	
A	1	0	A	3	010	A	2	00
B	2	10	B	3	101	B	2	11
C	4	1100	C	4	0011	C	5	01010
D	4	1101	D	4	1100	D	5	10101
E	4	1110	E	5	01001	E	5	10000
F	5	11110	F	5	01111	F	5	01111
G	6	111110	G	5	10010	G	5	11000
H	6	111111	H	5	11100	H	5	00110

T=8

(a)

	Proposed		[16]
Table	Dist <sub>inter</sub>	Dist <sub>intra</sub>	Min(d <sub>TV</sub> )
C2a	1/7	23	1
C2c	1	22	2
C2d	1	13	2

(b)

Figure 5.8 : VLC tables (a) and measurements (b) for the same source.

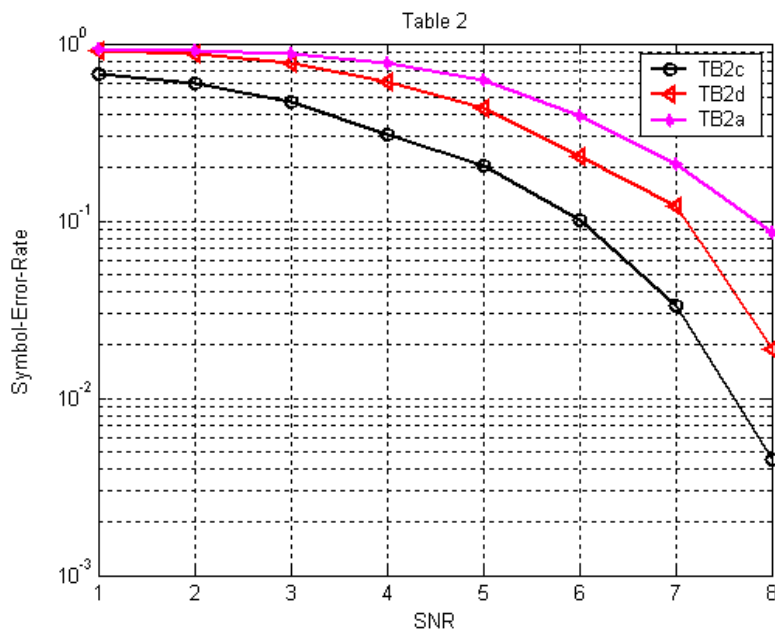


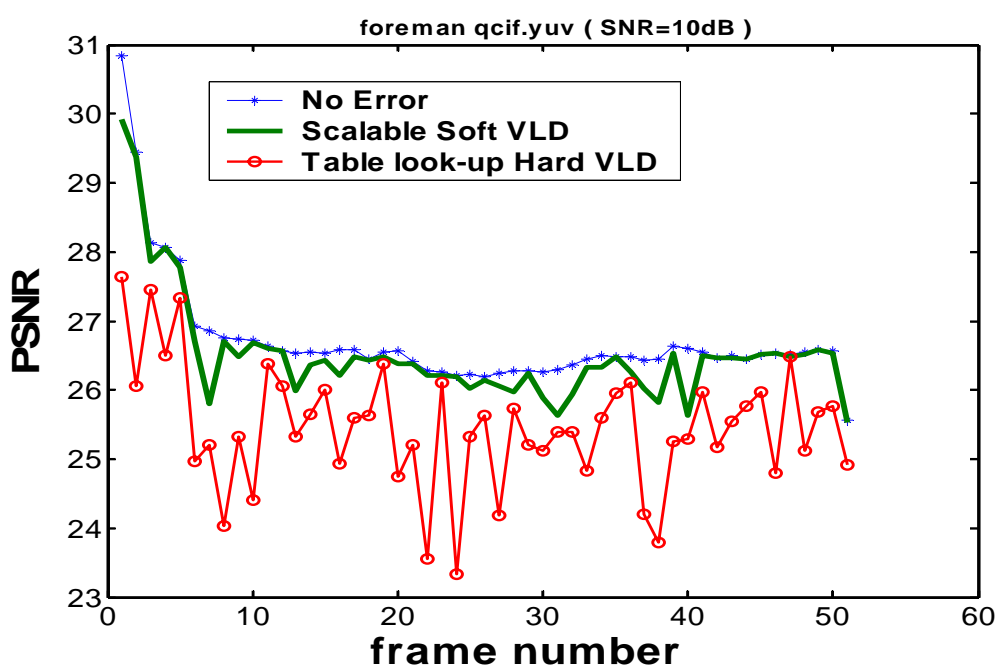
Figure 5.9 : Coding performance with different VLC coding table [16]



Based on the proposed B-B model, the soft VLC decoder is parameterized. To optimize the performance and the complexity, we include the B-B model in our evaluation of scalable soft VLD. We use foreman (QCIF) as our test sequence and encode the sequence at 64kbits/s and 15fps (No P-frame). In Figure 5.10(a), our proposed scalable soft VLD shows that more than 1dB PSNR can be gained compared with the table look-up decoding at BER= $10^{-3}$  (SNR=10dB). Further, the parameters determined by the B-B model are listed in Figure 5.10(b). The ‘T’ is determined from Table 4.1 with the given MPEG-4 table and the others are determined through our proposed B-B model. We choose ‘z’ as 2 for the complexity reduction. Then, we choose ‘N’ as 300 bits for the performance optimization. The performance improvement of our proposed scheme will become more prominent when the upper bound of ‘No error’ is increased. In the subjective quality comparison of Figure 5.11, our scheme shows better quality.

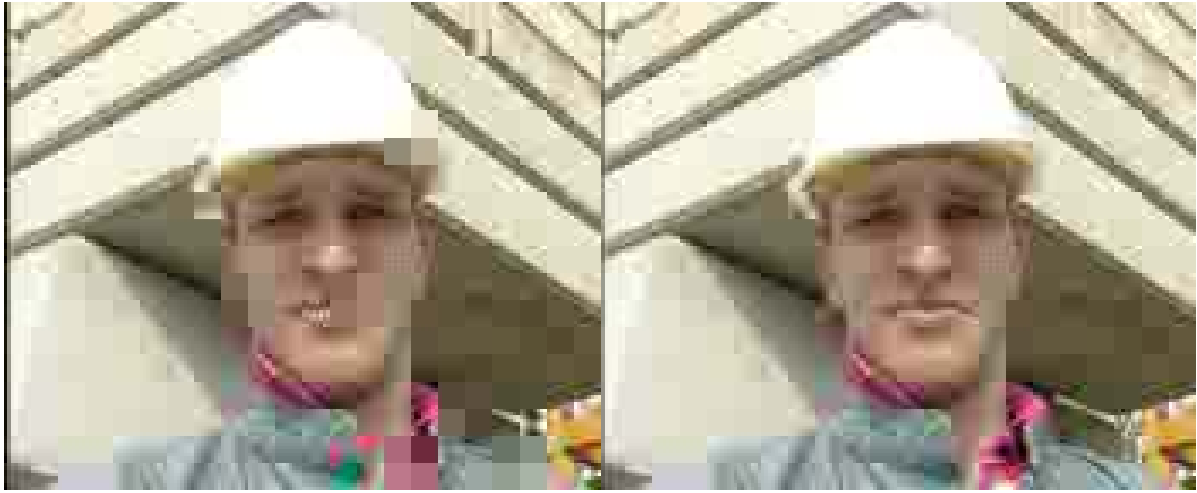
B-B Model	
Algorithm Parameter (z)	2
Application Parameter (N)	~300bits/packet
Table Parameter (T)	48 (see Table 4.1)

(a)



(b)

Figure 5.10 : The simulated parameters (a) and PSNR comparison (b) within 50 frames.



(a) Table Look-up Hard Decoding      (b) Scalable Soft VLC Decoding

Figure 5.11 : The comparison on the 1<sup>st</sup> frame of video sequence.

## 5.4 Summary

In this chapter, the *parameterized soft VLC decoder* using a new performance modeling approach has been proposed. We present a novel measurement of “symbol-alias” to improve the accuracy of performance estimation. Simulation results show that our proposed measurement provides more accurate performance than [16] for the different tables. With the proposed B-B model, we can achieve the optimal trade-off between performance and complexity. For the proposed soft VLC decoder using performance modeling, we can averagely improve the PSNR by 1dB and offer better subjective quality as compared with the table-look-up hard decoding.

# Chapter 6

## Performance Evaluation on MPEG-4

### 6.1 *Environment Setup*

Until now, we have introduced and established this new design. We propose to study the feasibility and interest of soft VLC decoding for the existing video standards such as MPEG-4. Thus, we verify our proposed scalable soft VLC decoder (i.e. scalable soft VLD) over the AWGN channel using BPSK modulation. The input sequence is MPEG-4 encoded with the re-synchronization marker and the data partition. In data partition mode, we have assumed that the texture part, composed of a sequence of VLC code-words, be corrupted by AWGN. The other parts are of error free. This assumption can be achieved by exploiting the UEP or RCPC Codes [32]. To the ESCAPE code, we simply use hard decoding. Further, we use the soft output of quantizer as our input bit-stream of soft VLC decoder. The soft VLD can overlook the bit-errors (i.e. Figure 6.7) from the quantizer of physical layer since we assumed that an UDP-Lite protocol [33] is applied to our simulation model.

The overall simulation chain is depicted in Figure 6.1, and the major function blocks are addressed on the following sections. We partition all of them into two main parts. One is the function block of source coder, and the other is the function block of channel coder. The proposed soft VLC can be considered as an error-correcting function block. It's a joint source and channel design evaluating on the practical MPEG-4 system.

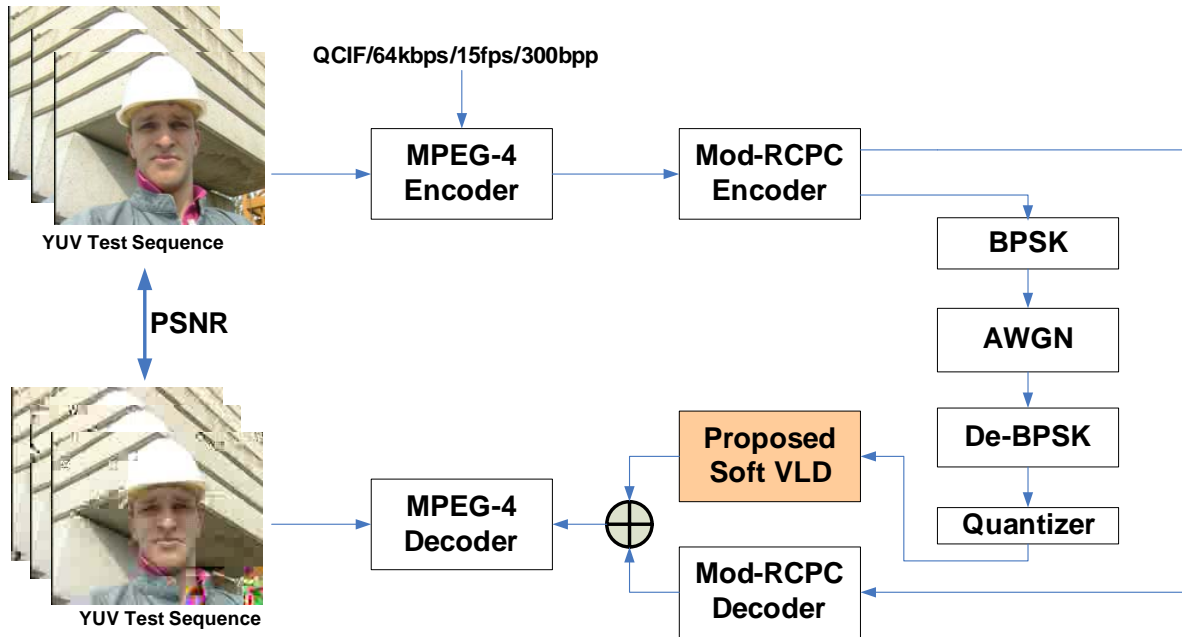


Figure 6.1 : The proposed overall simulation environment of soft VLC decoder.

### 6.1.1 Source Model

#### Resynchronization Marker

Resynchronization marker (i.e. RM) is one of the error resilient tools in MPEG-4. it attempts to enable resynchronization between the decoder and the encoded bit-stream. This is especially helpful in the case of bursty errors as it provides the decoder with the capability of “refresh start”. Further, to apply the data partition illustrated in the next paragraph, the RM tools have to be enabled on our simulation chain. The number of inserted RM will affect the coding efficiency. The more the number of RM leads to the less the coding efficiency. The trade-off has been optimized by the proposed B-B model in chapter 5, and the RM is inserted with a period of 300bits in Figure 6.1. (i.e. 300bits/packet)

#### Data Partition

To achieve better error isolation in the video packet and fixed interval synchronization approached, MPEG introduced data partition. When the data partition syntax is exploited, the video bit-stream is divided into two bit-streams by inserting a unique

marker among them. Each of them has a different sensitivity to channel errors. As shown in Figure 6.2(a), I-frame partitions consist of a header, DC DCT coefficients and AC DCT coefficients separated by a DC marker. As far as P frames are concerned, partitions consist of a header, a motion partition and a texture partition, separated by a motion marker. In addition, the data partition in MPEG-2 performs roughly the same with the one in MPEG-4 (i.e. Figure 6.2(b)).

Error resilient tools produce a further improvement of the received video quality if exploited at channel coding level. Because soft VLC decoder only applied to the AC transform coefficients of the bit-stream, it's essential to use the data partition tool with the purpose of performing unequal error protection (i.e. UEP), discussed more detailed on RCPC Codes. (see chapter 6.1.2)

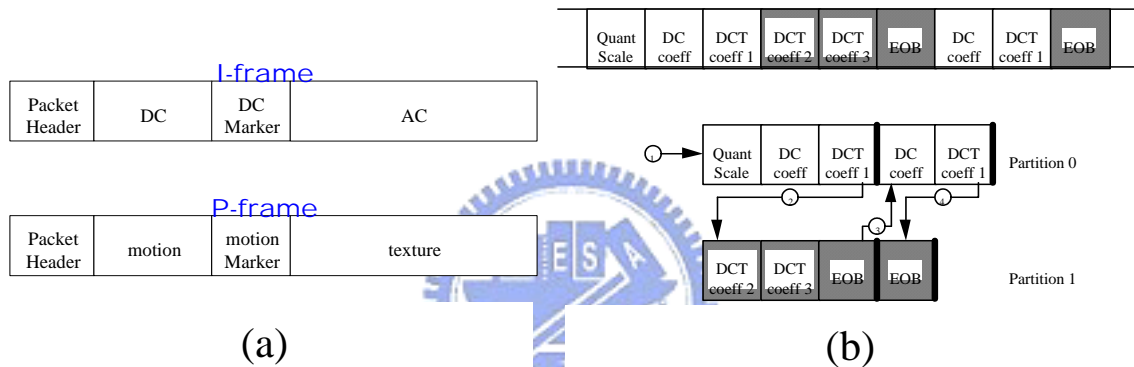


Figure 6.2 : The data partition mode in MPEG-4 (a) and MPEG-2 (b).

## ESCAPE code Handler

The existence of an ESCAPE mode in the MPEG syntax for texture encoding prevents the direct application of soft decoding algorithms to the extracted texture partition. In the MPEG-2 video standard, the ESCAPE code is encoded with “000001” followed by a fixed length code of 6-bit ‘run’ and 12-bit ‘signed\_level’. We easily use table look-up hard decoding with fixed length codes when encountering this specific codeword. However, it becomes more complicated for the ESCAPE code in MPEG-4 video standard. They utilize multiple tables to look-up the ‘Run’ or ‘Level’ of symbol information because of the improvement of coding efficiency. To deal with this complicated coding behavior, we just artificially include ESCAPE mode codeword and adapt the algorithm to automatically treat this fixed length code extension [29].

```

ESCAPE_Code_Handler ( ) // for MPEG-2 ESCAPE code
{
    // Step1.1 : Find ESCAPE code.

    // Step1.2 : Look-up TB-16 for 6-bit RUN and 12-bit Signed_Level.

    // Step1.3 : Fixed length decoding is performed easily.
}

ESCAPE_Code_Handler ( ) // for MPEG-4 ESCAPE code
{
    // Step2.1 : Find ESCAPE code.

    // Step2.2 : Look-up different table in different types of ESCAPE code.

    // Step2.3 : much more difficult to handle, just artificially included.
}

```

Figure 6.3 : The high-level description of ESCAPE code handler on MPEG-2 and MPEG-4.

## 6.1.2 Channel Model

### UDP-Lite Protocol

UDP is a simple best effort transport protocol. Unlike TCP, UDP does not provide reliability, in-order delivery or congestion control, which made it especially popular among delay-sensitive real-time applications. Further, audio/video applications often prefer damaged packets over lost packets. One way for an application to allow delivery of damaged packets is to disable the UDP check sum. These applications could benefit from using UDP Lite instead of UDP.

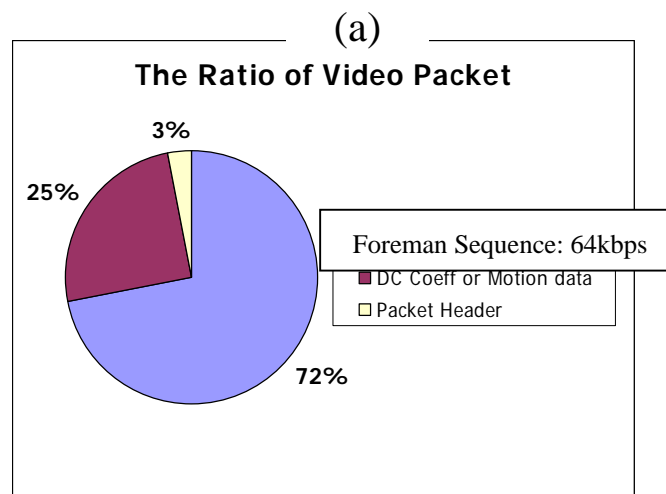
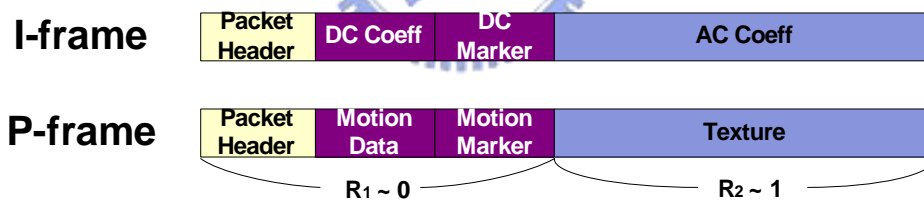
UDP-Lite [33] is a lightweight version of UDP with increased flexibility in the form of a partial checksum. UDP-Lite provides a check sum with an optional partial coverage. When enabling this option, a packet is partitioned into a sensitive and an insensitive part. Errors in the insensitive part will not cause the packet to be discarded by the transport layer at the receiving end-user. When the check sum covers the entire packet, which should be the default and UDP-Lite is semantically identical to UDP.

Based on the UDP-Lite protocol, we can easily apply our algorithm to the delay-sensitive real-time transmission on the MPEG-4/UDP-Lite. However, to perform the UDP-Lite effectively, the MPEG-4 has to enable on the “Resynchronization Marker” and “Data Partition” mode. Besides, the RCPC codes are exploited to cooperate with the

MPEG-4/UDP-Lite to construct our simulation model.

## Rate Compatible Punctured Convolutional Codes (RCPC Codes)

RCPC Codes is used to implement the unequal error protection (i.e. UEP) and connected with the UDP-Lite and data partition of MPEG-2/4. The overall simulation model is performed packet by packet. In our channel model, we assumed that the modified RCPC has been employed on our simulation chain. This modified RCPC can be described in Figure 6.4(a). Particularly, the first three data partition (i.e. header, DC coefficient and DC marker for I frames; header, motion data and motion marker for P frames) play an important role in the decoding procedure of source decoder. The loss of these parts will introduce the loss of synchronization and corrupt the whole frames. Therefore, we paid lots of efforts to protect this part (i.e.  $R_1 \sim 0$ ). However, about 72% of video packet is less important (i.e. Figure 6.4(b)), there is no reason to protect them as the small coding rate (i.e.  $R_2 \sim 1$ ). We assume that there is no any channel coding performed on this part, and recovered by the source decoder (e.g. RVLC decoder) or joint source and channel decoder (e.g. Soft VLC Decoder).



(b)

Figure 6.4 : The content (a) and ratio (b) of one video packet in MPEG-4.

## Quantizer

The quantizer is used to provide the soft bit-stream or soft information on the certain channel condition and application. In the BPSK modulation, the modulated symbol is either +1 or -1. After the channel corruption, the demodulated signal will become 0 or 1 in the hard decision scheme. But, if we exploit the quantizer to implement the soft decision method, the demodulated symbol will range from 0 to  $2^q-1$  [34]. The bit number  $q$  is used to quantize the symbol and determined by channel and application.

For example of  $q=3$ , Figure 6.5 shows the uniform quantization after the BPSK demodulation.  $\Delta$  is the step size and the quantized symbol can be formulated as Equation 6.1. To optimize the system performance using the quantization step, we simulate the relation between different  $q$  value and the system performance. Thus, in Figure 6.6, we can achieve the optimal trade-off between system cost and performance when  $q$  is equal to 4. In Figure 6.6(b), it shows that 16-level quantizer is a good choice because it obtains the complexity reduction with the price of minor performance loss. Therefore, based on the above statements, we use the 4-bit (i.e. 16-level) quantizer to construct the overall simulation on MPEG-4/UDP-Lite/UEP/AWGN.

$$S_q = \lfloor I \times (1/\Delta) + 4.5 \rfloor \quad (6.1)$$

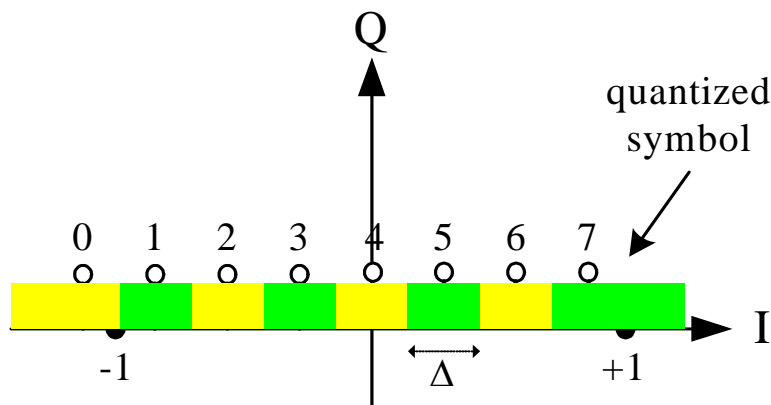
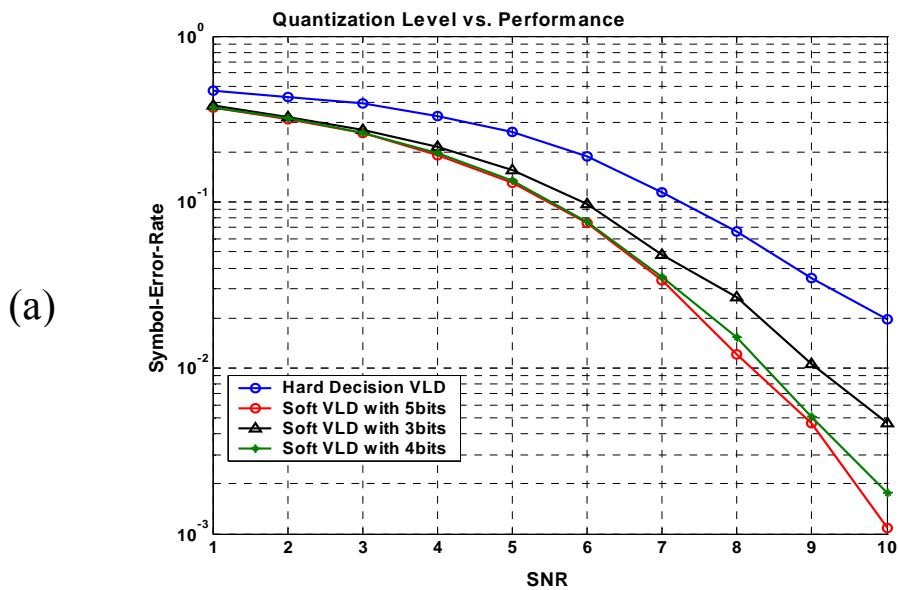


Figure 6.5 : The soft input of VLC decoder.





(b)

Improvement of the Soft VLD in the SER of  $10^{-1}$

	32-level	16-level	8-level
<b>BPSK</b>	1.765dB	1.72dB	1.284dB

Figure 6.6 : The performance improvement (a)(b) with different quantization level.

In summary, we have addressed the detailed function block of source and channel coder. In addition, we can also partition all of them into the 5 layers of OSI (Open System Interconnection) model in Figure 6.7. In the wireless network or mobile transmission, we assume that UDP-Lite of transport layer and UEP of link layer are provided. Further, we exploit the soft bit-stream after the 16-level quantizer and overlook the soft bit-error of physical layer into the application layer. Based on the above statements, we are going to evaluate the proposed design on the MPEG-4/UDP-Lite/UEP/AWGN in next section.

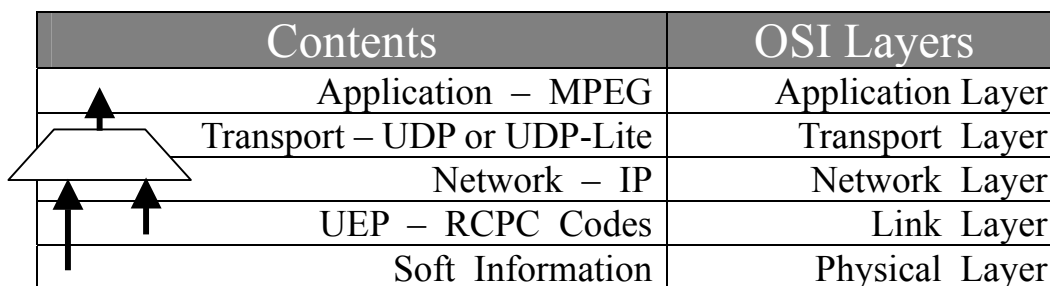


Figure 6.7 : Overlooking bit errors in application layer.

## 6.2 Performance Evaluation on the MPEG-4/UDP-Lite/UEP/AWGN

The proposed soft VLC decoder can be evaluated on any VLC-based video compression. We use MPEG-4 as the test model in the source coding part. It is also suitable for any MPEG-x and the H.26x series of video coding standard. The only difference among them is the source VLC symbol tables. The proposed soft VLC decoder has been parameterized in chapter 5, justified and proved the practicability in our simulation chain.

In our simulation of Figure 6.8, we use foreman (QCIF resolution, 50frames) as our test sequence and encode the sequence at 64kbps and 15fps for the wireless or mobile communication. Each video packet contains 300bits and the intra interval is 1 (i.e. no P-frame, only intra-coded). Further, there is no any side information to be transmitted, thus, the proposed soft VLC decoder is bandwidth-efficient. In this comparison, we assume that the anchor (i.e. TLU VLD) has no any error concealment scheme performed. Therefore, we can obtain a fair and neat comparison in the objective quality.

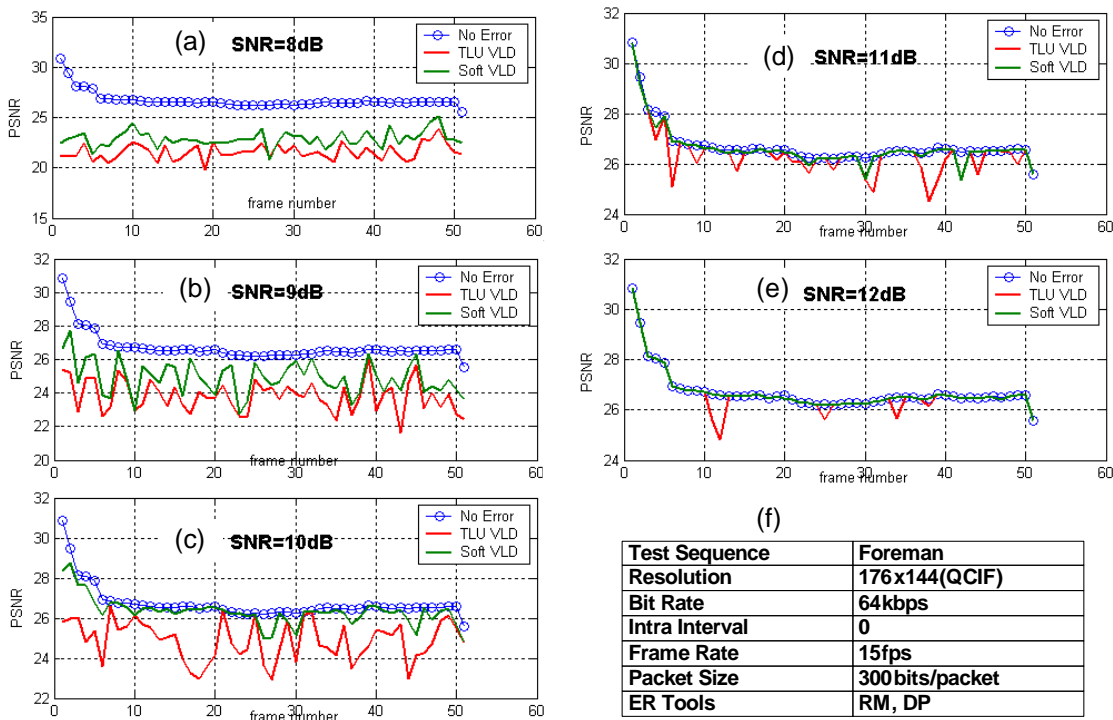


Figure 6.8 : Average PSNR of Y component for proposed soft VLD and TLU VLD.

To apply the proposed soft VLC decoder, we enable the error resilient tools in MPEG-4 (i.e. RM and DP). In addition, we also enable the mechanism of rate control to achieve the given bit-rate. Therefore, there are several frames at the beginning have more higher PSNR quality in Figure 6.8. To summarize the above simulation, Figure 6.9 shows the PSNR versus the different channel condition. Our proposed scalable soft VLC decoder is standard compliant and provides more than 1dB PSNR gains as compared with the straightforward table look-up decoding (i.e. TLU VLD) when the SNR=10 (i.e. BER= $10^{-3}$ ). Further, we also show that the improvement on the burst errors of channel environment versus the different burst length ranging from 3 to 20 bits. In Figure 6.10, the simulation result shows that the improvements of 1dB are gained as compared with the traditional TLU decoding method in terms of Y-component or average PSNR. At the same condition of channel model, we apply the standard-support RVLD to our simulation chain. Figure 6.11 depicts that the improvement of 0.5dB has been found between RVLD and traditional VLD.

Finally, the proposed design is also compared with the existing methods with error recovery capability (see Table 6.1) such as the RVLD, SSVLD of source recovery or the Viterbi, Turbo decoder of channel recovery. Based on the advantage of joint design (joint source and channel), the proposed scalable soft VLD can achieve a compromise between the coding performance (i.e. capability of error recovery) and the channel bandwidth. In the source coding side, MPEG-4 supports the Reversible VLC table method to improve the error resilient video transmission, but it may induced the coding overhead of 2.2% than the traditional VLC table [35]. In addition, [10] proposed a self-synchronization VLC decoding algorithm to improve the coding overhead at the same performance compared with RVLD. In the other way, forward error correction codes provides the high capability of error recovery, but it has to pay the great penalty of coding overhead (e.g. code rate=1/2, a.k.a. coding overhead=200%). As shown in Table 6.1, the proposed scalable soft VLD is standard compliant and there is no any side information to be required. It is highly advantageous to the band-limited video transmission (e.g. wireless or mobile communication). Further, it provides high performance in terms of PSNR with objective measurement.

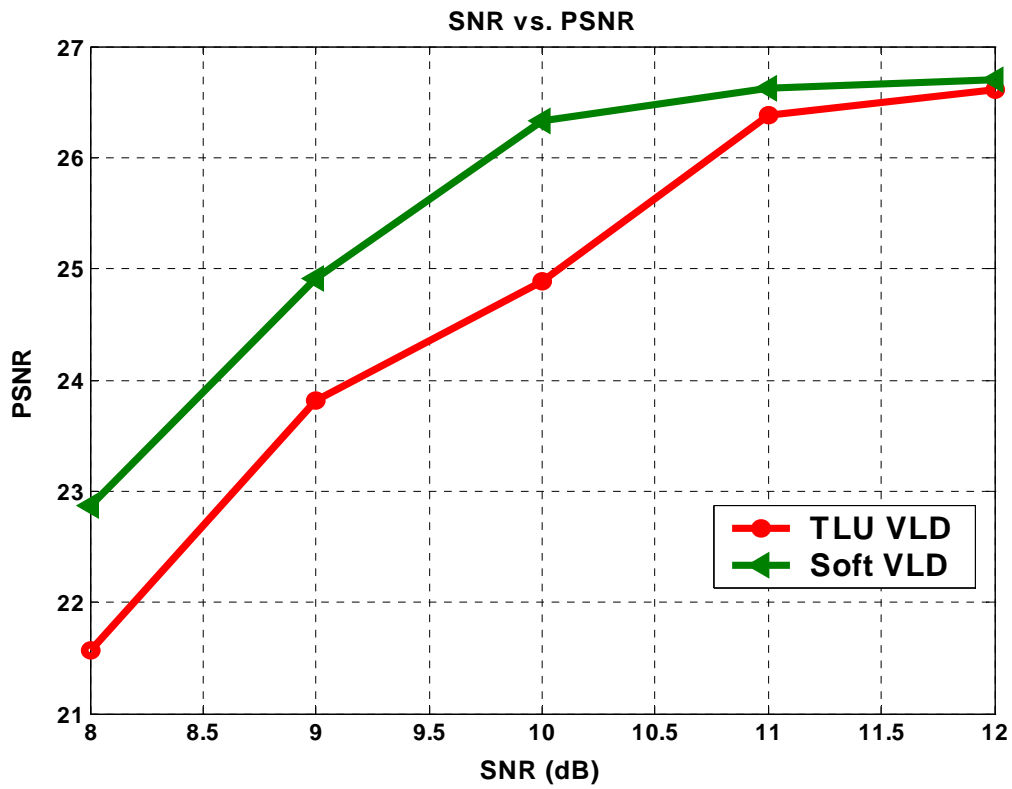


Figure 6.9 : PSNR vs. AWGN channel performance

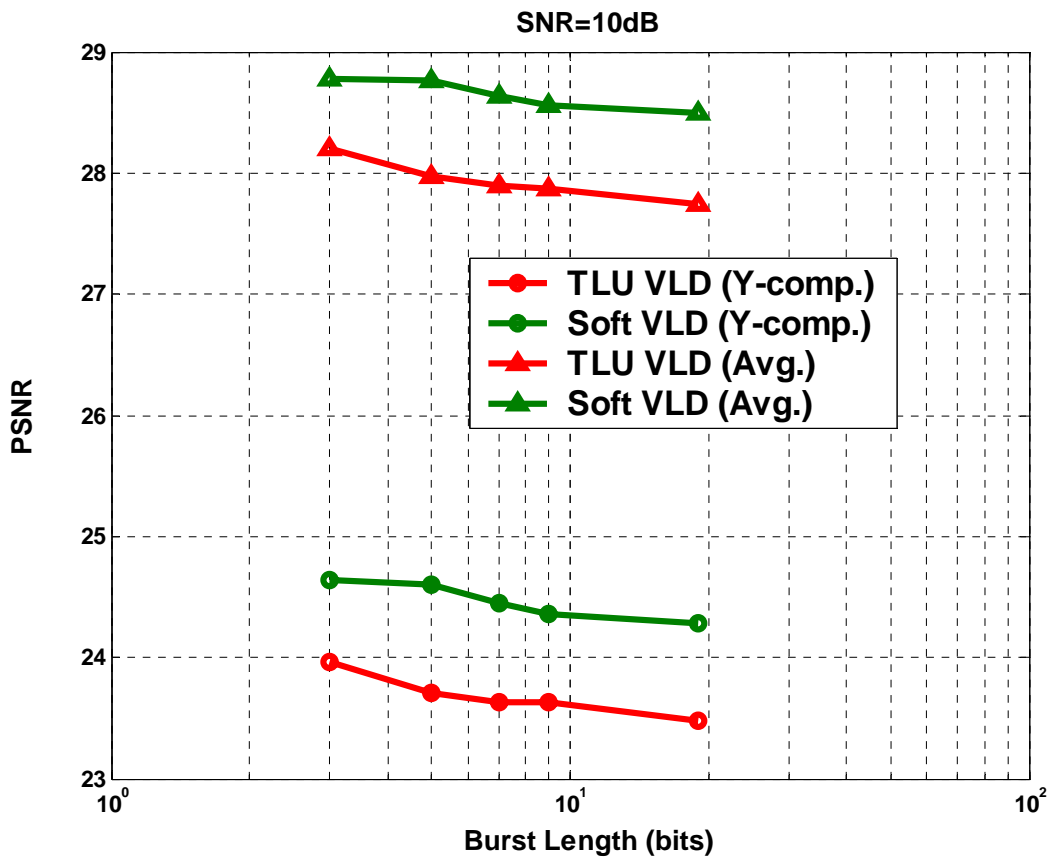


Figure 6.10 : PSNR vs. Burst error performance

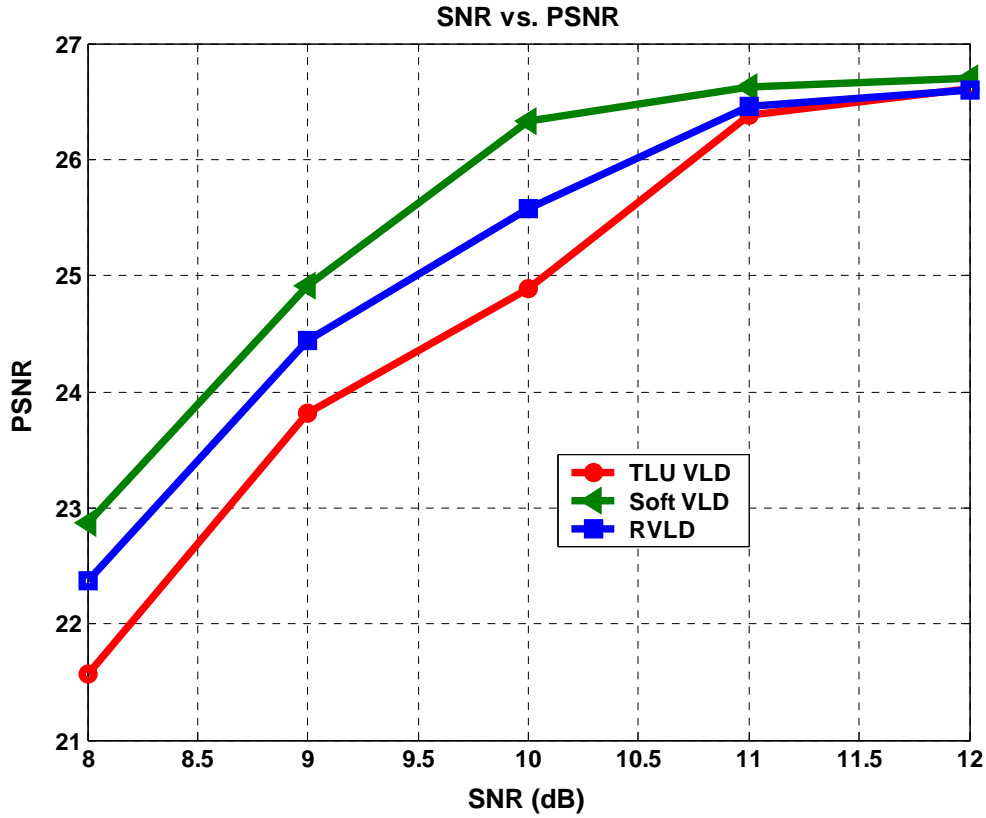


Figure 6.11 : The comparison between the proposed soft VLD and the RVLD.

Table 6.1 : The trade-off between error correction and channel bandwidth.

	<b>Coding Method</b>	<b>Performance</b>	<b>Channel Bandwidth/ Coding Efficiency</b>
<b>1</b>	Table Look-Up VLD <sub>[DP+RM]</sub>	Anchor	Anchor
<b>2</b>	Reversible VLD <sub>[30]</sub>	~ 0.5dB	2.2%
<b>3</b>	Self-Sync. VLD <sub>[10]</sub>	~ 0.5dB	1.6% (0.6% overhead reduction)
<b>4</b>	Scalable Soft VLD <sub>[proposed]</sub>	~ 1.2dB	0%
<b>5</b>	Viterbi, Turbo Decoder	~ Shannon Bound	200% for Viterbi 300%~500% for Turbo

The above simulation is based on all I-frame assumption. We address the results in I-P-P structure of encoded bit-stream below. If the P frame is involved in our encoded sequence, the VLC decoder has to switch the table to deal with the different frame configuration. It means that not only intra but also non-intra VLC coding table should be exploited on the decoding procedure of the proposed soft VLD. Based on the proposed table-merging algorithm, we merge two tables into one super-set VLC coding

table. Therefore, the proposed VLC decoder can easily switch table on the decoding procedure instead of the duplicated structure [31]. Figure 6.12 shows that the improvement of 1.5dB has been found at the SNR=10dB. Further, at the different source characteristics (i.e. Foreman, Suzie and Silent), we show that a PSNR improvement ratio of 40~60% can be achieved at the condition of 64kbps, 15fps, 300bits/packet and 12frames/intra interval.

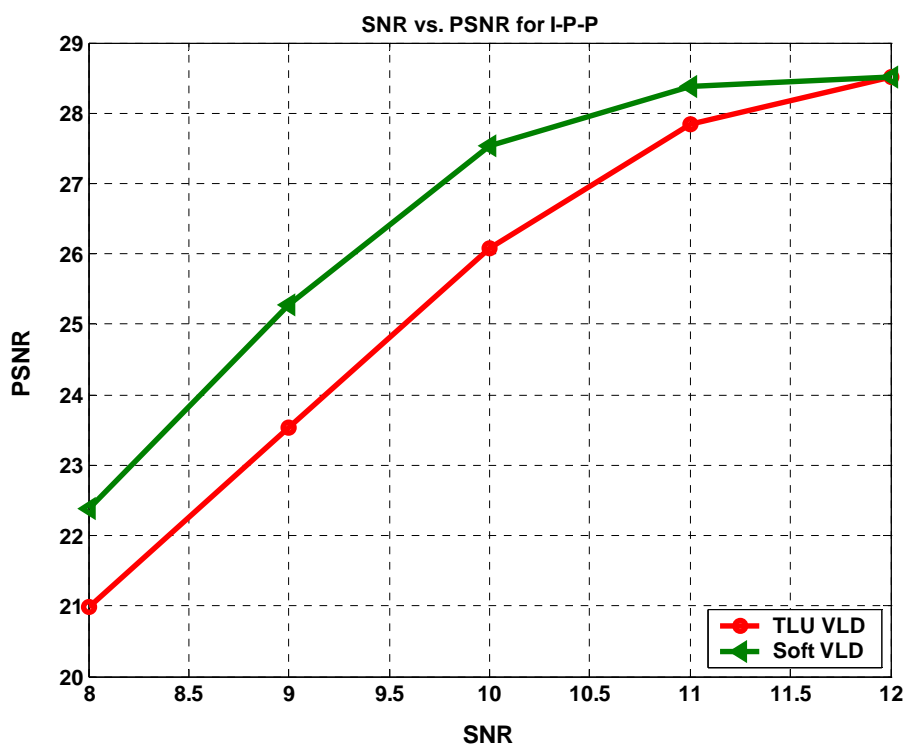


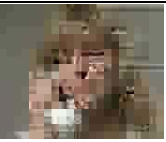


Figure 6.12 : The comparison between the proposed soft VLC decoder and the standardized VLC decoder for Multi-Table-Merging.

Table 6.2 : The PSNR improvement within different video characteristics.

Video pattern			
QCIF resolution, 64kbps, 15fps, 300bit/packet, 12frames/intra interval, 10dB/channel condition			
Total frame number	Foreman, 400 frames	Suzie, 150 frames	Silent, 300 frames
PSNR improvement	0.417dB	2.896dB	0.7752dB
Improved ratio	41.90%	85.377%	52.22%

# Chapter 7

## Conclusions and Future Work

In this dissertation, we propose an efficient and scalable soft VLC decoder to significantly reduce the memory utilization and decoding complexity. The proposed performance modeling reaches the optimal trade-off between performance and complexity for the multimedia communication.

Generally, the soft VLC decoder needs to maintain many states for the correct decoding when the sequence length or table size grows. Our approach reduces the complexity by simplifying the algorithm and reducing the table size. Specifically, we simplify the algorithm by adaptively selecting the survival states to reduce the number of memory access. Further, we reduce the table size by using a symbol-merging scheme. We merge two symbols with the same prefix into one. To share the same soft VLC decoder on the different VLC table, we propose a novel soft VLC decoder with table merging algorithm to reduce the implementation cost. Particularly, we utilize the codeword merging and prefix merging method to realize the table merging scheme. In order to obtain optimal performance and complexity, we propose a Black-Box model. In this proposed model, we present a novel measurement of “symbol-alias” to improve the accuracy of performance estimation.

Experimental results show that our proposed adaptive scheme can averagely save 15% of memory access as compared to the state-of-the-art algorithms. Furthermore, our proposed scalable soft VLC decoder has more than 0.4~2.9dB PSNR gain and offers better subjective quality compared with the table look-up decoding method and the standard-support RVLD.

There are still several improvements can be done in this research in the near future. First, only software implementation and complexity analysis have been completed in this work. It is essential to pay more attentions on the real-time implementation and low-latency transmission. In addition, the proposed soft VLC decoder is performed only on the AC transform coefficients. It's an in-significant and partial part of the whole MPEG video bit-stream. Therefore, designing a soft VLC decoder that can cope with the whole VLC symbol bit-stream becomes a great challenge in the near future. In this thesis, the proposed soft VLC decoder can be applied not only MPEG-4 but also MPEG-2 AC transform coefficients. However, the newly video standard (i.e. H.264/AVC) is created and very different from the former standards. It is an interesting research to extend the proposed design to this novel video standard.





# Bibliography

- [1] D. A. Huffman, "A method for the construction of minimum-redundancy codes", in *Proc. IRE*, vol. 40, pp. 1098-1101, Sept. 1952.
- [2] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Final Draft International Standard (FDIS), Mar. 2003.
- [3] S. Shirani, F. Kossentini, and R. Ward, "An adaptive markov random field based error concealment method for video communication in an error prone environment", in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, (ICASSP'99)* March 1999.
- [4] C. E. Shannon, "A mathematical theory of communication", *Bell Syst. Tech. J.*, vol. 27, pp. 379-423-623-656, 1948.
- [5] S. Vembu, S. Verdu, and Y. Steinberg, "The source channel theorem revisited", *IEEE Trans. Inform. Theory*, vol. 41, pp. 44-54, Jan. 1995.
- [6] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques", *IEEE Signal Processing Magazine*, vol. 17, pp. 61-82, July 2000.
- [7] S. Shirani, F. Kossentini, and R. Ward. "Error Concealment Methods, A Comparative Study." *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 835-840, 1999.
- [8] Y. Takishima, M. Wada, H. Murakami, "Reversible Variable Length Codes," *IEEE Trans. Comm.*, vol. 43, No. 2/3/4, pp158-162, 1995.
- [9] David W. Redmill and Nick G. Kingsbury, "The EREC: An error-resilient technique for

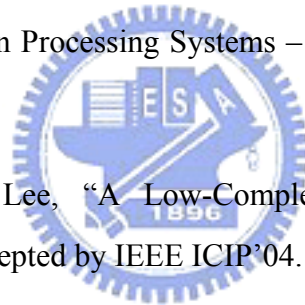
- coding variable-length blocks of data", *IEEE Trans. on Image Processing*, vol. 5, no. 4, pp. 565-574, April 1996.
- [10] G. Y. Hong, B. Fong, and A. C. M. Fong, "Error Localization for Robust Video Transmission", *IEEE Trans. on Consumer Elec.*, Vol. 48, Issue: 3, pp. 463-469, Aug. 2002.
- [11] European Telecommunications Standards Inst. (ETSI), ETS 300 744 (1997): *Digital Broadcasting Systems for Television, Sound and Data Services; Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television*, 1997.
- [12] Jeanne, M.; Carlach, J.C.; Siohan, P.; Guivarch, L.; "Source and Joint Source-Channel Decoding of Variable Length Codes" *IEEE International Conference on Communications*, vol. 2, pp.768-772, May 2002.
- [13] Demir, N.; Sayood, K., "Joint source/channel coding for variable length codes", *Data Compression Conference '98, Proceedings*, pp. 139-148, April 1998.
- [14] V. B. Balakirsky, "Joint source-channel coding with variable length codes" in Proceedings of *IEEE ISIT*, Ulm, Germany, 1997.
- [15] R. Bauer and J. Hagenauer, "On variable length codes for iterative source/channel-decoding", in *Proc. IEEE Data Compression Conf.*, pp. 273-282, 2001.
- [16] Bystrom, M.; Kaiser, S.; Kopansky, A., "Soft source decoding with applications", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, Issue. 10, pp. 1108 - 1120, Oct. 2001.
- [17] Viterbi A. J., "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Trans. on Information Theory*, Vol. IT-13, pp. 260-269, 1967.
- [18] Buttigieg V. & Farrell P.G., "A maximum likelihood decoding algorithm for variable-length error-correcting codes", *Proc. 5th Bangor Symposium on Communications*, Bangor, Wales, pp. 56-59, 2-3 Jun. 1993.
- [19] Jiangtao Wen; Villasenor, J., "Soft-input soft-output decoding of variable length codes", *IEEE Transactions on Communications*, Vol. 50, Issue: 5, pp. 689-692, May 2002
- [20] J. Hagenauer and P. Poehrer, "A Viterbi algorithm with soft-decision outputs and its

applications”, *Proc. IEEE GLOBECOM*, Dallas, TX, Nov. 1989.

- [21] J. Hagenauer, “Source-controlled channel decoding”, *IEEE Trans. on Communication*, Vol. 43, pp. 2449-2457, Sept. 1995.
- [22] Buttigieg V. & Farrell P.G., "A maximum a-posteriori (MAP) decoding algorithm for variable-length error-correcting codes", *Codes and cyphers: Cryptography and coding IV*, Essex, England, The Institute of Mathematics and its Applications, pp. 103-119, 1995.
- [23] M. Park and D. J. Miller, “Joint source-channel decoding for variable length encoded data by exact and approximated MAP sequence estimation”, *ICASSP’99*, Phoenix, Arizona, USA, pp. 2451-2454, March 15-19, 1999.
- [24] Lamy, C.; Pothier O., “Reduced complexity Maximum A Posteriori decoding of variable-length codes”, *GLOBECOM ‘01*, vol.2, pp.25-29, Nov. 2001.
- [25] M. Park and D. J. Miller, “Joint source-channel decoding for variable length encoded data by exact and approximate MAP sequence estimation”, *IEEE Transactions on Communications*, pp. 1-6, vol. 48, no. 1, Jan. 2000.
- [26] F. Jelinek, “A fast sequential decoding algorithm using a stack”, *IBM J. Res. and Dev.*, pp. 675-685, Nov. 1969.
- [27] Buttigieg V. & Farrell P.G., "Sequential decoding of variable-length error-correcting codes", *Eurocode 94*, Côte d'Or, France, 24-28 Oct. 1994.
- [28] R. Bauer and J. Hagenauer, Turbo-FEC/VLC-decoding and its application to text compression“, *Proceedings of the Conference on Information Sciences and System (CISS’00)*, Princeton University, USA, pp. WA6-6 – WA-11, Mar. 15-17, 2000.
- [29] L. Perros-Meilhac and C. Lamy, “Huffman tree based metric derivation for a low-complexity sequential soft VLC decoding”, *in Proc. of ICC’02*, pp. 783-787, vol. 2, New York, USA, Apr. 2002.
- [30] ISO/IEC 14496-2. Information technology – Coding of audio-visual objects – Part 2: Visual, December 1999.
- [31] Q. Chen and K. P. Subbalakshmi, “Joint Source-Channel Decoding for MPEG-4 Video

Transmission over Wireless Channels”, *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 1780-1789, Dec. 2003.

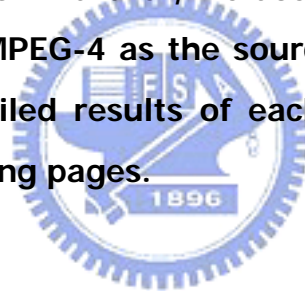
- [32] J. Hagenauer, “Rate-Compatible Puncture Convolutional Codes (RCPC Codes) and their applications”, *IEEE Trans. on Communications*, vol. 36, no. 4, pp. 389-400, April 1988.
- [33] Larzon, et al., “The UDP Lite Protocol”, Internet Draft (work in progress), draft-ietf-tsvwg-udp-lite-02.txt, Lulea University of Technology, Aug. 2003.
- [34] Onyszchuk, I.M.; Cheung, K.-M.; Collins, O. “Quantization Loss in Convolutional Decoding”, *IEEE Trans. on Communications*, Vol. 41, Issue. 2, Feb. 1993.
- [35] Chien-Wu Tsai, To-Ju Huang, Kuo-Lin Fang, and Ja-Ling Wu, “A Hybrid and Flexible H.263-based Error Resilient and Testing System”, *IEEE-TENCON*, vol. 1, pp. 19- 22, Aug. 2001.
- [36] ISO/IEC 7498-1. Information Processing Systems – OSI Reference Model – The Basic Model 1994.
- [37] Tsu-Ming Liu, Chen-Yi Lee, “A Low-Complexity Soft VLC Decoder Using Performance Modeling”, accepted by IEEE ICIP’04.



# Appendix A

## Symbol Merging Algorithm

The symbol merging algorithm is used to reduce the table size. We merge two symbols with the same "LAST" and "SIGN" field into single merged-symbol. We show the pseudo-code and a simple example on the following description. Further, we use the INTRA and NON-INTRA table of MPEG-2 and MPEG-4 as the source table. After the proposed merging scheme, detailed results of each merging process are also depicted on the following pages.



### High Level Description of Symbol-Merging Algorithm

---

#### Symbol Merging Algorithm ( merge-i )

```
{ inc = 1;
  Merging_Condition(x,y) = ((x XOR y) == CL'b1) && (LASTx == LASTy) &&
  (SIGNx == SIGNy);
  While (inc == i)
    For (whole symbol combinations in one coding table)
      { Find( CodeLength(symbolm) == CodeLength(symboln))
        { If(Merging_Condition(symbolm, symboln))
          { Merged_Symbol = symbolm(CodeLengthm-2:0) >> 1;
            Merged_SIGN = SIGNm;
```

```

        Merged_LAST = LASTm;
    }
}
}
inc = inc + 1;
end while
}

```

**Example:**

index	symbol	LAST	SIGN	Merged_Symbol	Merged_SIGN	Merged_LAST
1(=m)	0010	0	1	001	0	1
2(=n)	0011	0	1			



NOTE1 : This code shall be used for the first (DC) coefficient in the block.

NOTE2 : ESCAPE code

MPEG-2 Table B-14				MPEG-2 Table B-14 [Merge-1]				MPEG-2 Table B-14 [Merge-2]			
0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST
1	10	0	1	1	10	0	1	1	10	0	1
2	1 (NOTE1)	1	0	2	1(NOTE1)	1	0	2	1(NOTE1)	1	0
3	11	1	0	3	11	1	0	3	11	1	0
4	011	1	0	4	011	1	0	4	011	1	0
5	0100	1	0	5	010	2	0	5	010	2	0
6	0101	1	0	6	00101	1	0	6	00101	1	0
7	00101	1	0	7	0011	2	0	7	0011	2	0
8	00110	1	0	8	00010	2	0	8	0001	3	0
9	00111	1	0	9	00011	2	0	9	00001	3	0
10	000100	1	0	10	000010	2	0	10	000001(NOTE2)	0	0
11	000101	1	0	11	000011	2	0	11	001000	3	0
12	000110	1	0	12	000001(NOTE2)	0	0	12	001001	3	0
13	000111	1	0	13	0010000	2	0	13	0000010	3	0
14	0000100	1	0	14	0010001	2	0	14			
15	0000101	1	0	15	0010010	2	0	15			
16	0000110	1	0	16	0010011	2	0	16			
17	0000111	1	0	17	00000100	2	0	17			
18	000001(NOTE2)	0	0								
19	00100000	1	0								
20	00100001	1	0								
21	00100010	1	0								
22	00100011	1	0								
23	00100100	1	0								
24	00100101	1	0								
25	00100110	1	0								
26	00100111	1	0								
27	0000001000	1	0								

28	0000001001	1	0								
29	0000001010	1	0	18	000000101	2	0				
30	0000001011	1	0								
31	0000001100	1	0	19	000000110	2	0	14	00000011	3	0
32	0000001101	1	0								
33	0000001110	1	0	20	000000111	2	0				
34	0000001111	1	0								
35	000000010000	1	0	21	00000001000	2	0	15	0000000100	3	0
36	000000010001	1	0								
37	000000010010	1	0	22	00000001001	2	0				
38	000000010011	1	0								
39	000000010100	1	0	23	00000001010	2	0	16	0000000101	3	0
40	000000010101	1	0								
41	000000010110	1	0	24	00000001011	2	0				
42	000000010111	1	0								
43	000000011000	1	0	25	00000001100	2	0	17	0000000110	3	0
44	000000011001	1	0								
45	000000011010	1	0	26	00000001101	2	0				
46	000000011011	1	0								
47	000000011100	1	0	27	00000001110	2	0	18	0000000111	3	0
48	000000011101	1	0								
49	000000011110	1	0	28	00000001111	2	0				
50	000000011111	1	0								
51	0000000010000	1	0	29	000000001000	2	0	19	00000000100	3	0
52	0000000010001	1	0								
53	0000000010010	1	0	30	000000001001	2	0				
54	0000000010011	1	0								
55	0000000010100	1	0	31	000000001010	2	0	20	00000000101	3	0
56	0000000010101	1	0								



57	0000000010110	1	0	32	000000001011	2	0				
58	0000000010111	1	0								
59	0000000011000	1	0	33	000000001100	2	0	21	00000000110	3	0
60	0000000011001	1	0								
61	0000000011010	1	0	34	000000001101	2	0				
62	0000000011011	1	0								
63	0000000011100	1	0	35	000000001110	2	0	22	00000000111	3	0
64	0000000011101	1	0								
65	0000000011110	1	0	36	000000001111	2	0				
66	0000000011111	1	0								
67	00000000010000	1	0	37	0000000001000	2	0	23	000000000100	3	0
68	00000000010001	1	0								
69	00000000010010	1	0	38	0000000001001	2	0				
70	00000000010011	1	0								
71	00000000010100	1	0	39	0000000001010	2	0	24	000000000101	3	0
72	00000000010101	1	0								
73	00000000010110	1	0	40	0000000001011	2	0				
74	00000000010111	1	0								
75	00000000011000	1	0	41	0000000001100	2	0	25	000000000110	3	0
76	00000000011001	1	0								
77	00000000011010	1	0	42	0000000001101	2	0				
78	00000000011011	1	0								
79	00000000011100	1	0	43	0000000001110	2	0	26	000000000111	3	0
80	00000000011101	1	0								
81	00000000011110	1	0	44	0000000001111	2	0				
82	00000000011111	1	0								
83	000000000010000	1	0	45	00000000001000	2	0	27	0000000000100	3	0
84	000000000010001	1	0								
85	000000000010010	1	0	46	00000000001001	2	0				

86	000000000010011	1	0								
87	000000000010100	1	0	47	00000000001010	2	0	28	0000000000101	3	0
88	000000000010101	1	0								
89	000000000010110	1	0	48	00000000001011	2	0				
90	000000000010111	1	0					29	0000000000110	3	0
91	000000000011000	1	0	49	00000000001100	2	0				
92	000000000011001	1	0								
93	000000000011010	1	0	50	00000000001101	2	0				
94	000000000011011	1	0					30	0000000000111	3	0
95	000000000011100	1	0	51	00000000001110	2	0				
96	000000000011101	1	0								
97	000000000011110	1	0	52	00000000001111	2	0				
98	000000000011111	1	0					31	0000000000100	3	0
99	000000000010000	1	0	53	00000000001000	2	0				
100	000000000010001	1	0								
101	000000000010010	1	0	54	00000000001001	2	0				
102	000000000010011	1	0					32	0000000000101	3	0
103	000000000010100	1	0	55	00000000001010	2	0				
104	000000000010101	1	0								
105	000000000010110	1	0	56	00000000001011	2	0				
106	000000000010111	1	0					33	0000000000110	3	0
107	000000000011000	1	0	57	00000000001100	2	0				
108	000000000011001	1	0								
109	000000000011010	1	0	58	00000000001101	2	0				
110	000000000011011	1	0					34	0000000000111	3	0
111	000000000011100	1	0	59	00000000001110	2	0				
112	000000000011101	1	0								
113	000000000011110	1	0	60	00000000001111	2	0				
114	000000000011111	1	0								

MPEG-2 Table B-15				MPEG-2 Table B-15 [Merge-1]				MPEG-2 Table B-15 [Merge-2]			
0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST
1	0110	0	1	1	0110	1	1	1	0110	1	1
2	10	1	0	2	10	1	0	2	10	1	0
3	010	1	0	3	010	1	0	3	010	1	0
4	110	1	0	4	110	1	0	4	110	1	0
5	0111	1	0	5	0111	1	0	5	0111	1	0
6	00101	1	0	6	00101	1	0	6	00101	1	0
7	00111	1	0	7	0011	2	0	7	0011	2	0
8	00110	1	0								
9	11100	1	0	8	1110	2	0	8	1110	2	0
10	11101	1	0								
11	000001(NOTE2)	0	0	9	000001(NOTE2)	0	0	9	000001(NOTE2)	0	0
12	000110	1	0	10	00011	2	0	10	0001	3	0
13	000111	1	0								
14	000100	1	0	11	00010	2	0				
15	000101	1	0								
16	0000110	1	0	12	000011	2	0	11	00001	3	0
17	0000111	1	0								
18	0000100	1	0	13	000010	2	0				
19	0000101	1	0								
20	1111000	1	0	14	111100	2	0	12	11110	3	0
21	1111001	1	0								
22	1111010	1	0	15	111101	2	0				
23	1111011	1	0								
24	00100000	1	0	16	00100000	2	0	13	001000	3	0
25	00100001	1	0								
26	00100010	1	0	17	0010001	2	0				
27	00100011	1	0								

28	00100100	1	0	18	0010010	2	0	14	001001	3	0
29	00100101	1	0								
30	00100110	1	0	19	0010011	2	0				
31	00100111	1	0								
32	11111010	1	0	20	1111101	2	0	15	1111101	2	0
33	11111011	1	0								
34	11111100	1	0	21	1111110	2	0	16	111111	3	0
35	11111101	1	0								
36	11111110	1	0	22	1111111	2	0				
37	11111111	1	0								
38	000000100	1	0	23	00000010	2	0	17	00000010	2	0
39	000000101	1	0								
40	000000111	1	0	24	000000111	1	0	18	000000111	1	0
41	0000001100	1	0	25	000000110	2	0	19	000000110	2	0
42	0000001101	1	0								
43	000000010001	1	0	26	000000010001	1	0	20	000000010001	1	0
44	000000010010	1	0	27	000000010010	1	0	21	000000010010	1	0
45	000000010101	1	0	28	000000010101	1	0	22	000000010101	1	0
46	000000011100	1	0	29	000000011100	1	0	23	000000011100	1	0
47	000000011010	1	0	30	000000011010	1	0	24	000000011010	1	0
48	000000011001	1	0	31	000000011001	1	0	25	000000011001	1	0
49	000000010110	1	0	32	00000001011	2	0	26	00000001011	2	0
50	000000010111	1	0								
51	000000011110	1	0	33	00000001111	2	0	27	00000001111	2	0
52	000000011111	1	0								
53	0000000010000	1	0	34	000000001000	2	0	28	00000000100	3	0
54	0000000010001	1	0								
55	0000000010010	1	0	35	000000001001	2	0				
56	0000000010011	1	0								

57	0000000010100	1	0	36	000000001010	2	0	29	000000001010	2	0
58	0000000010101	1	0								
59	0000000010110	1	0	37	0000000010110	1	0	30	0000000010110	1	0
60	0000000011100	1	0	38	000000001110	2	0	31	00000000111	3	0
61	0000000011101	1	0								
62	0000000011110	1	0	39	000000001111	2	0				
63	0000000011111	1	0								
64	0000000011011	1	0	40	0000000011011	1	0	32	0000000011011	1	0
65	00000000010000	1	0	41	0000000001000	2	0	33	000000000100	3	0
66	00000000010001	1	0								
67	00000000010010	1	0	42	0000000001001	2	0				
68	00000000010011	1	0								
69	00000000010100	1	0	43	0000000001010	2	0	34	000000000101	3	0
70	00000000010101	1	0								
71	00000000010110	1	0	44	0000000001011	2	0				
72	00000000010111	1	0								
73	00000000011000	1	0	45	0000000001100	2	0	35	000000000110	3	0
74	00000000011001	1	0								
75	00000000011010	1	0	46	0000000001101	2	0				
76	00000000011011	1	0								
77	00000000011100	1	0	47	0000000001110	2	0	36	000000000111	3	0
78	00000000011101	1	0								
79	00000000011110	1	0	48	0000000001111	2	0				
80	00000000011111	1	0								
81	000000000010000	1	0	49	00000000001000	2	0	37	0000000000100	3	0
82	000000000010001	1	0								
83	000000000010010	1	0	50	00000000001001	2	0				
84	000000000010011	1	0								
85	1111100	1	0	51	1111100	1	0	38	1111100	1	0

86	000000000010100	1	0	52	00000000001010	2	0	39	0000000000101	3	0
87	000000000010101	1	0								
88	000000000010110	1	0	53	00000000001011	2	0				
89	000000000010111	1	0								
90	000000000011000	1	0	54	00000000001100	2	0	40	0000000000110	3	0
91	000000000011001	1	0								
92	000000000011010	1	0	55	00000000001101	2	0				
93	000000000011011	1	0								
94	000000000011100	1	0	56	00000000001110	2	0	41	0000000000111	3	0
95	000000000011101	1	0								
96	000000000011110	1	0	57	00000000001111	2	0				
97	000000000011111	1	0								
98	000000000010000	1	0	58	00000000001000	2	0	42	0000000000100	3	0
99	000000000010001	1	0								
100	000000000010010	1	0	59	00000000001001	2	0				
101	000000000010011	1	0								
102	000000000010100	1	0	60	00000000001010	2	0	43	0000000000101	3	0
103	000000000010101	1	0								
104	000000000010110	1	0	61	00000000001011	2	0				
105	000000000010111	1	0								
106	000000000011000	1	0	62	00000000001100	2	0	44	0000000000110	3	0
107	000000000011001	1	0								
108	000000000011010	1	0	63	00000000001101	2	0				
109	000000000011011	1	0								
110	000000000011100	1	0	64	00000000001110	2	0	45	0000000000111	3	0
111	000000000011101	1	0								
112	000000000011110	1	0	65	00000000001111	2	0				
113	000000000011111	1	0								

MPEG-4 Table B-16				MPEG-4 Table B-16 [Merge-1]				MPEG-4 Table B-16 [Merge-2]			
0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST
1	10	1	0	1	10	1	0	1	10	1	0
2	110	1	0	2	110	1	0	2	110	1	0
3	1110	1	0	3	111	2	0	3	111	2	0
4	1111	1	0	4	0110	2	0	4	0110	2	0
5	01100	1	0	5	01011	1	0	5	01011	1	0
6	01101	1	0	6	01000	2	0	6	0100	3	0
7	01011	1	0	7	01001	2	0	7	01010	2	0
8	010000	1	0	8	01010	2	0	8	00101	3	0
9	010001	1	0	9	001010	2	0	9	00011011	1	0
10	010010	1	0	10	001011	2	0	10	000111	3	0
11	010011	1	0	11	00011011	1	0	11	00001101	2	0
12	010100	1	0	12	0001110	2	0	12	0000111	3	0
13	010101	1	0	13	0001111	2	0	13	00001101	2	0
14	0010100	1	0	14	00001101	2	0	14	0000111	3	0
15	0010101	1	0	15	00001110	2	0	15	00001110	2	0
16	0010110	1	0	16	00001111	2	0	16	00001111	2	0
17	0010111	1	0								
18	00011011	1	0								
19	00011100	1	0								
20	00011101	1	0								
21	00011110	1	0								
22	00011111	1	0								
23	000011010	1	0								
24	000011011	1	0								
25	000011100	1	0								
26	000011101	1	0								
27	000011110	1	0								

28	000011111	1	0								
29	000100000	1	0	17	00010000	2	0	13	0001000	3	0
30	000100001	1	0								
31	000100010	1	0	18	00010001	2	0				
32	000100011	1	0								
33	000100100	1	0	19	00010010	2	0	14	00010010	2	0
34	000100101	1	0								
35	0000001000	1	0	20	000000100	2	0	15	00000010	3	0
36	0000001001	1	0								
37	0000001010	1	0	21	000000101	2	0				
38	0000001011	1	0								
39	0000001100	1	0	22	000000110	2	0	16	00000011	3	0
40	0000001101	1	0								
41	0000001110	1	0	23	000000111	2	0				
42	0000001111	1	0								
43	0000100000	1	0	24	000010000	2	0	17	000010000	2	0
44	0000100001	1	0								
45	00000000110	1	0	25	0000000011	2	0	18	0000000011	2	0
46	00000000111	1	0								
47	00000100000	1	0	26	0000010000	2	0	19	000001000	3	0
48	00000100001	1	0								
49	00000100010	1	0	27	0000010001	2	0				
50	00000100011	1	0								
51	000001010000	1	0	28	00000101000	2	0	20	0000010100	3	0
52	000001010001	1	0								
53	000001010010	1	0	29	00000101001	2	0				
54	000001010011	1	0								
55	000001010100	1	0	30	00000101010	2	0	21	0000010101	3	0
56	000001010101	1	0								



57	000001010110	1	0	31	00000101011	2	0				
58	000001010111	1	0								
59	001110	1	1	32	00111	2	1	22	00111	2	1
60	001111	1	1								
61	0111	1	1	33	0111	1	1	23	0111	1	1
62	001100	1	1	34	001100	1	1	24	001100	1	1
63	001101	1	0	35	001101	1	0	25	001101	1	0
64	0010010	1	0	36	0010010	1	0	26	0010010	1	0
65	0010011	1	1	37	0010011	1	1	27	0010011	1	1
66	0010000	1	1	38	001000	2	1	28	001000	2	1
67	0010001	1	1								
68	00011010	1	1	39	00011010	1	1	29	00011010	1	1
69	00011000	1	0	40	0001100	2	0	30	0001100	2	0
70	00011001	1	0								
71	00010110	1	1	41	00010110	1	1	31	00010110	1	1
72	00010111	1	0	42	00010111	1	0	32	00010111	1	0
73	00010100	1	1	43	0001010	1	1	33	0001010	1	1
74	00010101	1	1								
75	00010011	1	1	44	00010011	1	1	34	00010011	1	1
76	000010010	1	1	45	00001001	2	1	35	00001001	2	1
77	000010011	1	1								
78	000010100	1	1	46	00001010	2	1	36	0000101	3	1
79	000010101	1	1								
80	000010110	1	1	47	00001011	2	1				
81	000010111	1	1								
82	000011000	1	0	48	00001100	2	0	37	00001100	2	0
83	000011001	1	0								
84	000010001	1	1	49	000010001	1	1	38	000010001	1	1
85	0000000110	1	1	50	0000000110	1	1	39	0000000110	1	1

86	0000000111	1	0	51	0000000111	1	0	40	0000000111	1	0
87	0000000100	1	1	52	000000010	2	1	41	000000010	2	1
88	0000000101	1	1								
89	00000000100	1	1	53	0000000010	2	1	42	0000000010	2	1
90	00000000101	1	1								
91	00000100100	1	0	54	0000010010	2	1	43	000001001	3	1
92	00000100101	1	0								
93	00000100110	1	0	55	0000010011	2	1				
94	00000100111	1	0								
95	000001011000	1	1	56	000001011000	1	1	44	000001011000	1	1
96	000001011001	1	0	57	000001011001	1	0	45	000001011001	1	0
97	000001011010	1	1	58	00000101101	2	1	46	00000101101	2	1
98	000001011011	1	1								
99	000001011100	1	1	59	00000101110	2	1	47	0000010111	3	1
100	000001011101	1	1								
101	000001011110	1	1	60	00000101111	2	1				
102	000001011111	1	1								
103	0000011(NOTE2)	0	0	61	0000011(NOTE2)	0	0	48	0000011(NOTE2)	0	0

$$B.D.(1) = \frac{1 - \frac{61}{103}}{1 \times 0.5} = 81.53\%$$

$$B.D.(2) = \frac{\left(1 - \frac{61}{103}\right) + \left(1 - \frac{48}{61}\right)}{2 \times 0.5} = 62.09\%$$

MPEG-4 Table B-17				MPEG-4 Table B-17 [Merge-1]				MPEG-4 Table B-17 [Merge-2]			
0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST
1	10	1	0	1	10	1	0	1	10	1	0
2	110	1	0	2	110	1	0	2	110	1	0
3	1110	1	0	3	111	2	0	3	111	2	0
4	1111	1	0	4	0110	2	0	4	0110	2	0
5	01100	1	0	5	01011	1	0	5	01011	1	0
6	01101	1	0	6	01000	2	0	6	0100	3	0
7	01011	1	0	7	01001	2	0	7	01010	2	0
8	010000	1	0	8	01010	2	0	8	00101	3	0
9	010001	1	0	9		2	0	9	00011011	1	0
10	010010	1	0	10	001011	2	0	10	000111	3	0
11	010011	1	0	11	00011011	1	0	11	00001101	2	0
12	010100	1	0	12	0001110	2	0	12	0000111	3	0
13	010101	1	0	13	0001111	2	0	13			
14	0010100	1	0	14	00001101	2	0	14			
15	0010101	1	0	15	00001110	2	0	15			
16	0010110	1	0	16	00001111	2	0	16			
17	0010111	1	0								
18	00011011	1	0								
19	00011100	1	0								
20	00011101	1	0								
21	00011110	1	0								
22	00011111	1	0								
23	000011010	1	0								
24	000011011	1	0								
25	000011100	1	0								
26	000011101	1	0								
27	000011110	1	0								

28	000011111	1	0								
29	000100000	1	0	17	00010000	2	0	13	0001000	3	0
30	000100001	1	0								
31	000100010	1	0	18	00010001	2	0				
32	000100011	1	0								
33	000100100	1	0	19	00010010	2	0	14	00010010	2	0
34	000100101	1	0								
35	0000001000	1	0	20	000000100	2	0	15	00000010	3	0
36	0000001001	1	0								
37	0000001010	1	0	21	000000101	2	0				
38	0000001011	1	0								
39	0000001100	1	0	22	000000110	2	0	16	00000011	3	0
40	0000001101	1	0								
41	0000001110	1	0	23	000000111	2	0				
42	0000001111	1	0								
43	0000100000	1	0	24	000010000	2	0	17	000010000	2	0
44	0000100001	1	0								
45	00000000110	1	0	25	0000000011	2	0	18	0000000011	2	0
46	00000000111	1	0								
47	00000100000	1	0	26	0000010000	2	0	19	000001000	3	0
48	00000100001	1	0								
49	00000100010	1	0	27	0000010001	2	0				
50	00000100011	1	0								
51	000001010000	1	0	28	00000101000	2	0	20	0000010100	3	0
52	000001010001	1	0								
53	000001010010	1	0	29	00000101001	2	0				
54	000001010011	1	0								
55	000001010100	1	0	30	00000101010	2	0	21	0000010101	3	0
56	000001010101	1	0								

57	000001010110	1	0	31	00000101011	2	0				
58	000001010111	1	0								
59	0111	1	1	32	0111	1	1	22	0111	1	1
60	001100	1	1	33	00110	2	1	23	0011	3	1
61	001101	1	1								
62	001110	1	1	34	00111	2	1				
63	001111	1	1								
64	0010000	1	1	35	001000	2	1	24	00100	3	1
65	0010001	1	1								
66	0010010	1	1	36	001001	2	1				
67	0010011	1	1								
68	00011010	1	1	37	00011010	1	1	25	00011010	1	1
69	00010100	1	1	38	0001010	2	1	26	000101	3	1
70	00010101	1	1								
71	00010110	1	1	39	0001011	2	1				
72	00010111	1	1								
73	00011000	1	1	40	0001100	2	1	27	0001100	2	1
74	00011001	1	1								
75	00010011	1	1	41	00010011	1	1	28	00010011	1	1
76	000010010	1	1	42	00001001	2	1	29	00001001	2	1
77	000010011	1	1								
78	000010100	1	1	43	00001010	2	1	30	0000101	3	1
79	000010101	1	1								
80	000010110	1	1	44	00001011	2	1				
81	000010110	1	1								
82	000011000	1	1	45	00001100	2	1	31	00001100	2	1
83	000011001	1	1								
84	000010001	1	1	46	000010001	1	1	32	000010001	1	1
85	0000000100	1	1	47	000000010	2	1	33	00000001	3	1

86	0000000101	1	1								
87	0000000110	1	1	48	000000011	2	1				
88	0000000111	1	1								
89	0000000100	1	1	49	000000010	2	1	34	000000010	2	1
90	0000000101	1	1								
91	00000100100	1	1	50	0000010010	2	1	35	000001001	3	1
92	00000100101	1	1								
93	00000100110	1	1	51	0000010011	2	1				
94	00000100111	1	1								
95	000001011000	1	1	52	00000101100	2	1	36	0000010110	3	1
96	000001011001	1	1								
97	000001011010	1	1	53	00000101101	2	1				
98	000001011011	1	1								
99	000001011100	1	1	54	00000101110	2	1	37	0000010111	3	1
100	000001011101	1	1								
101	000001011110	1	1	55	00000101111	2	1				
102	000001011111	1	1								
103	0000011(NOTE2)	0	0	56	0000011(NOTE2)	0	0	38	0000011(NOTE2)	0	0

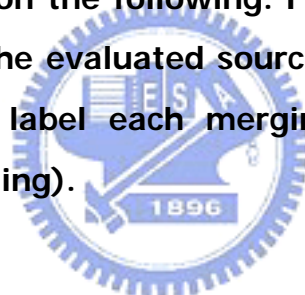
$$B.D.(1) = \frac{1 - \frac{56}{103}}{1 \times 0.5} = 91.26\%$$

$$B.D.(2) = \frac{\left(1 - \frac{56}{103}\right) + \left(1 - \frac{38}{56}\right)}{2 \times 0.5} = 77.77\%$$

# Appendix B

## Table Merging Algorithm

Table switching is essential for the VLC decoding in the practical system. To share the decoder, we use the table-merging algorithm to reduce the implementation cost and the memory access. We show a high-level description on the following. Further, we use the practical tables on MPEG-4 as the evaluated source table. We list the detailed merging process and label each merging method (i.e. code-word merging or prefix merging).

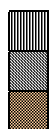


### High Level Description of Table Merging Algorithm

---

#### Table Merging Algorithm ( Table<sub>i</sub>, Table<sub>j</sub> )

{



: stands for the "code-word merging" is performed.

: stands for the "prefix merging" is performed.

: stands for the overhead after table-merging algorithm.

**For** (whole symbol combinations in different coding table)

{

**If**( Table<sub>i</sub>(code<sub>x</sub>) == Table<sub>j</sub>(code<sub>y</sub>) )

MTM\_Merging\_Condition = CodeWord\_Merging;

**Elseif**( Table<sub>i</sub>(code<sub>x</sub>) == prefix of "Table<sub>j</sub>(code<sub>y</sub>)" )

MTM\_Merging\_Condition = Prefix\_Merging;

**Else**

MTM\_Merging\_Condition = zero\_pending

**Case** (MTM\_Merging\_Condition)

CodeWord\_Merging: Code-Word merging is performed;

Prefix\_Merging: Prefix merging is performed;

**Default:** table overhead due this MTM process;

**Endcase**

}

}





MPEG-2 Table B-14 T2				MPEG-2 Table B-15 T2				MTM MPEG-2 Table (B-14-15)					
0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST	0	VLC CODE	TB-14		TB-15	
										VALID	CL	VALID	CL
1	10	0	1	1	0110	1	1	1	0110	1	3	1	4
2	1(NOTE1)	1	0	2	10	1	0	2	10	1	2	1	2
3	11	1	0	3	010	1	0	3	010	1	3	1	3
4	011	1	0	4	110	1	0	4	110	1	2	1	3
5	010	2	0	5	0111	1	0	5	0111	0	-	1	4
6	00101	1	0	6	00101	1	0	6	00101	1	5	1	5
7	0011	2	0	7	0011	2	0	7	0011	1	4	1	4
8	0001	3	0	8	1110	2	0	8	1110	0	-	1	4
9	00001	3	0	9	00001(NOTE2)	0	0	9	00001(NOTE2)	1	6	1	6
10	00001(NOTE2)	0	0	10	0001	3	0	10	0001	1	4	1	4
11	001000	3	0	11	00001	3	0	11	00001	1	5	1	5
12	001001	3	0	12	11110	3	0	12	11110	0	-	1	5
13	00000010	3	0	13	001000	3	0	13	001000	1	6	1	6
14	00000011	3	0	14	001001	3	0	14	001001	1	6	1	6
15	0000000100	3	0	15	1111101	2	0	15	1111101	0	-	1	7
16	0000000101	3	0	16	111111	3	0	16	111111	0	-	1	6
17	0000000110	3	0	17	00000010	2	0	17	00000010	1	8	1	8
18	0000000111	3	0	18	000000111	1	0	18	000000111	1	8	1	9
19	00000000100	3	0	19	000000110	2	0	19	000000110	0	-	1	9
20	00000000101	3	0	20	000000010001	1	0	20	000000010001	1	10	1	12
21	00000000110	3	0	21	000000010010	1	0	21	000000010010	0	-	1	12
23	00000000111	3	0	22	000000010101	1	0	22	000000010101	1	10	1	12
22	000000000100	3	0	23	000000011100	1	0	23	000000011100	1	10	1	12
24	000000000101	3	0	24	000000011010	1	0	24	000000011010	1	10	1	12
25	000000000110	3	0	25	000000011001	1	0	25	000000011001	0	-	1	12
26	000000000111	3	0	26	00000001011	2	0	26	00000001011	0	-	1	11

27	0000000000100	3	0	27	00000001111	2	0	27	00000001111	0	-	1	11
28	0000000000101	3	0	28	00000000100	3	0	28	00000000100	1	11	1	11
29	0000000000110	3	0	29	000000001010	2	0	29	000000001010	1	11	1	12
30	0000000000111	3	0	30	0000000010110	1	0	30	0000000010110	0	-	1	13
31	00000000000100	3	0	31	000000000111	3	0	31	000000000111	1	11	1	11
32	00000000000101	3	0	32	00000000011011	1	0	32	00000000011011	1	11	1	13
33	000000000000110	3	0	33	0000000000100	3	0	33	0000000000100	1	12	1	12
34	000000000000111	3	0	34	0000000000101	3	0	34	0000000000101	1	12	1	12
35				35	0000000000110	3	0	35	0000000000110	1	12	1	12
36				36	0000000000111	3	0	36	0000000000111	1	12	1	12
37				37	00000000000100	3	0	37	00000000000100	1	13	1	13
38				38	1111100	1	0	38	1111100	0	-	1	7
39				39	0000000000101	3	0	39	0000000000101	1	13	1	13
40				40	0000000000110	3	0	40	0000000000110	1	13	1	13
41				41	0000000000111	3	0	41	0000000000111	1	13	1	13
42				42	00000000000100	3	0	42	00000000000100	1	14	1	14
43				43	00000000000101	3	0	43	00000000000101	1	14	1	14
44				44	000000000000110	3	0	44	000000000000110	1	14	1	14
45				45	000000000000111	3	0	45	000000000000111	1	14	1	14

MPEG-4 Table B-16 T2				MPEG-4 Table B-17 T2				MTM MPEG-4 Table (B-16-17)					
0	VLC CODE	SIGN	LAST	0	VLC CODE	SIGN	LAST	0	VLC CODE	TB-16		TB-17	
										VALID	CL	VALID	CL
1	10	1	0	1	10	1	0	1	10	1	2	1	2
2	110	1	0	2	110	1	0	2	110	1	3	1	3
3	111	2	0	3	111	2	0	3	111	1	3	1	3
4	0110	2	0	4	0110	2	0	4	0110	1	4	1	4
5	01011	1	0	5	01011	1	0	5	01011	1	5	1	5
6	0100	3	0	6	0100	3	0	6	0100	1	4	1	4
7	01010	2	0	7	01010	2	0	7	01010	1	5	1	5
8	00101	3	0	8	00101	3	0	8	00101	1	5	1	5
9	00011011	1	0	9	00011011	1	0	9	00011011	1	8	1	8
10	000111	3	0	10	000111	3	0	10	000111	1	6	1	6
11	00001101	2	0	11	00001101	2	0	11	00001101	1	8	1	8
12	0000111	3	0	12	0000111	3	0	12	0000111	1	7	1	7
13	0001000	3	0	13	0001000	3	0	13	0001000	1	7	1	7
14	00010010	2	0	14	00010010	2	0	14	00010010	1	8	1	8
15	00000010	3	0	15	00000010	3	0	15	00000010	1	8	1	8
16	00000011	3	0	16	00000011	3	0	16	00000011	1	8	1	8
17	000010000	2	0	17	000010000	2	0	17	000010000	1	9	1	9
18	0000000011	2	0	18	0000000011	2	0	18	0000000011	1	10	1	10
19	000001000	3	0	19	000001000	3	0	19	000001000	1	9	1	9
20	0000010100	3	0	20	0000010100	3	0	20	0000010100	1	10	1	10
21	0000010101	3	0	21	0000010101	3	0	21	0000010101	1	10	1	10
23	0111	1	1	22	0111	1	1	22	0111	1	4	1	4
22	00111	2	1	23	0011	3	1	23	00111	1	5	1	4
24	001100	1	1	24	00100	3	1	24	001100	1	6	0	-
25	001101	1	0	25	00011010	1	1	25	001101	1	6	0	-
26	0010010	1	0	26	000101	3	1	26	0010010	1	7	1	5

27	0010011	1	1	27	0001100	2	1	27	0010011	1	7	0	-
28	001000	2	1	28	00010011	1	1	28	001000	1	6	0	-
29	00011010	1	1	29	00001001	2	1	29	00011010	1	8	1	8
30	0001100	2	0	30	0000101	3	1	30	0001100	1	7	1	7
31	00010110	1	1	31	00001100	2	1	31	00010110	1	8	1	6
32	00010111	1	0	32	000010001	1	1	32	00010111	1	8	0	-
33	0001010	1	1	33	00000001	3	1	33	0001010	1	7	0	-
34	00010011	1	1	34	0000000010	2	1	34	00010011	1	8	1	8
35	00001001	2	1	35	000001001	3	1	35	00001001	1	8	1	8
36	0000101	3	1	36	0000010110	3	1	36	0000101	1	7	1	7
37	00001100	2	0	37	0000010111	3	1	37	00001100	1	8	1	8
38	000010001	1	1	38	<b>0000011</b>	0	0	38	000010001	1	9	1	9
39	0000000110	1	1		<b>(NOTE2)</b>			39	0000000110	1	10	1	8
40	0000000111	1	0					40	0000000111	1	10	0	-
41	000000010	2	1					41	000000010	1	9	0	-
42	0000000010	2	1					42	0000000010	1	10	1	10
43	000001001	3	1					43	000001001	1	9	1	9
44	000001011000	1	0					44	000001011000	1	12	1	10
45	000001011001	1	1					45	000001011001	1	12	0	-
46	00000101101	2	1					46	00000101101	1	11	0	-
47	0000010111	3	1					47	0000010111	1	10	1	10
48	<b>0000011</b>	0	0					48	<b>0000011</b>	1	7	1	7
	<b>(NOTE2)</b>								<b>(NOTE2)</b>				

# 自傳簡歷

姓名：劉子明

性別：男

國籍：中華民國

籍貫：陝西省府谷縣

出生日期：民國 69 年 8 月 24 日

地址：新竹市 新莊街 148 巷 6 號 6 樓

電話：(03)5712121-54238, 0933977393

學歷：

民國 87 年 9 月至民國 91 年 6 月 國立交通大學電子工程系 學士

民國 91 年 9 月迄今 國立交通大學電子研究所 碩士



經歷：

1998	✓ 就讀國立交通大學電子物理系
1999	✓ 參加電子物理研習營，擔任器材組組員 ✓ 申請校內轉系，進入交大電子系
2000	✓ 參加北區大專盃團體組國樂比賽，榮獲優等第二名 ✓ 擔任 2000 千禧國樂營總召
2001	✓ 參加北區大專盃團體組國樂比賽，榮獲優等第四名 ✓ 免口試直升交大電子所系統組碩士班
2002	✓ 以大學部的專題，投上 ISCAS'02 會議論文 ✓ 此大學專題榮獲殷之同講學金
2003	✓ 碩一期間榮獲上下學期兩次電子所書卷 ✓ 擔任國科會微電子學門的網頁管理者 ✓ 榮獲九十二年度黎明文化基金會研究生獎學金及獎狀乙幀 ✓ 獲頒九十三年度斐陶斐榮譽會員

# 著作目錄

## 1. 會議論文

- **Tzu-Ming Liu**, Bai-Jue Shieh, Chen-Yi Lee, “An Efficient Modeling Codec Architecture for Binary Shape Coding”, in *Proceedings of the 2002 IEEE International Symposium on Circuit and System*, vol. 2, pp. II 316- II 319, May 2002.
- **Tsu-Ming Liu**, Chen-Yi Lee, “A Low-Complexity Soft VLC Decoder Using Performance Modeling”, accepted by IEEE ICIP’2004.
- **Tsu-Ming Liu**, Sheng-Zen Wang, Weng-Hsiao Peng, Chen-Yi Lee, “Memory-Efficient and Low-Complexity Scalable Soft VLC Decoding for the Video Transmission“, Submitting to IEEE APCCAS’2004.

