

Chapter 4

Error-Robust Design Approach

4.1 Background

Of all modalities desirable for future mobile multimedia systems, high-quality motion video over a reliable transmission is the most demanding. However, the transmitted visual quality may suffer abruptly because the channel deteriorates due to fading, co-channel interference, and signal attenuations [79]. On the other hand, the compressed video is extremely vulnerable against transmission errors, since video coding schemes rely on variable length codes (VLCs) for improving coding efficiency. With VLCs, the channel-induced errors can have a detrimental effect on the received bit stream. This is because a single bit that is received in error can influence the remaining bits in the stream due to unchecked error propagation. This causes the VLC decoder to discard many properly received data bits before synchronization is re-established. To deal with the transmission errors over an error-prone channel, much effort has been invested to improve the error-robustness in the source decoding procedures, such as error-resilient [72][73] tools, error-detection [74]–[76] and error-concealment [76][77] algorithms. Although channel coding can be used for error detection and correction to reduce the impact of channel-induced errors, this is accomplished by adding redundant information to the data stream for transmission and therefore introduces overhead to the transmitted bitstream. In this chapter, we consider the error-robustness in the source decoding side instead of channel coding and focus on the error detection and concealment method for improving the error-robustness under the mobile environment.

4.2 *Design Challenges*

Variable length codes (VLCs), also called Huffman codes [78] are commonly 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 new design of H.264/AVC. However, most of the VLC designs are highly sensitive to error disturbances and suffers from error propagation in the remaining VLC symbols. Hence, a robust transmission of VLC-based bitstream over mobile communication channels presents several challenging problems that remain to be resolved. One of the most important problems is that the success of error concealment techniques relies on the correct detection of erroneous macroblocks (MBs). However, conventional syntax-based error detection methods [76][80] may detect erroneous MBs late or even result in a failure of detection for a corrupted bit-stream. This is because invalid codes may not occur when the input stream is corrupted, and this stream may be decodable even in the presence of errors. Hence, one cannot find the exact location of errors or correctly detect errors within the bit-stream. On the other hand, error concealment is challenging as well due to both high-quality and low-complexity requirements. Specifically, it operates on MB boundaries and performs smoothing procedures to conceal the corrupted videos. In general, one may introduce two hardware building blocks such as deblocking filter and error concealment modules in a video decoding system. But, both modules include interpolating procedures to improve the visual quality, and they will not execute at the same time. Hence, how to integrate both functionalities into a single architecture without degrading visual quality is also a challenging task when developing our error-robust video decoder.

Beyond the aforementioned challenges, mobile video transmission also leads to error propagation on the frame level due to a motion-compensated coding method. They use the previous encoded and reconstructed frame to predict the next frame. Therefore, the loss of

information in one frame has considerable impact on the quality of the subsequent frames and this impact will be enlarged with prevalent frame-recompression methods [81]–[87]. In general, a frame-recompression method is desirable for reducing the memory cost in frame buffers. However, existing video-compression algorithms are not suitable for this kind of frame recompression because their main objective is high coding efficiency rather than error-robustness. On the other hand, existing solutions improve the compression efficiency but sacrifice simplicity and low-latency advantages, and vice versa. Hence, it is of great challenges to consider how to make a better compromise in different performance indexes.

4.2.1 System Highlights

In this work, we highlight three key points of our proposals in this error-roust video decoder. One is the soft computing on the context-adaptive VLC decoder (CAVLD) for improving the error detection capabilities. Another is the integration between error concealment and deblocking filter in order to meet both low cost and high performance requirements. The other is a new frame-recompression method to improve both power awareness and error robustness.

Figure 4.1 summarizes the soft-input video decoding system based on our previous work discussed in Chapters 2 and 3 of this dissertation. To deal with corrupted video streams and improve the subject and objective visual quality, we first introduce soft information based on the multi-level de-quantization process in each demodulated symbol. Because a soft-input stream retains the data reliability and informs the decoder about channel behaviors, it provides a reliable estimator to detect or localize the corrupted video. To apply a soft-stream into the video decoding system, a soft context-adaptive variable length decoder (CAVLD) has been newly designed and is capable of detecting erroneous positions. It notifies error concealment that whether the corrupted macroblock (MB) occurs

or not. Therefore, the corrupted data can be early concealed and the objective and subjective visual quality can be improved. On the other hand, we propose a joint architecture that meets the functionalities of error concealment as well as deblocking filter. We share the interpolating operations in both modules. As a result, this error-concealed deblocking filter can reduce the implementation cost with comparable PSNR performance of existing solutions. As for frame recompression method, an embedded compressor/de-compressor is exploited to compress the pixel data decoded by H.264/AVC and thereby cut the external DRAM power dissipation. This design features a power-aware design that can change the operating condition for different power requirements. Moreover, this compression method is aware of channel behavior by error-concealed deblocking filter. In other words, the compression method can change the operating procedures when the corrupted video is detected and concealed. Therefore, it achieves reduction of not only DRAM space but also computational complexity.

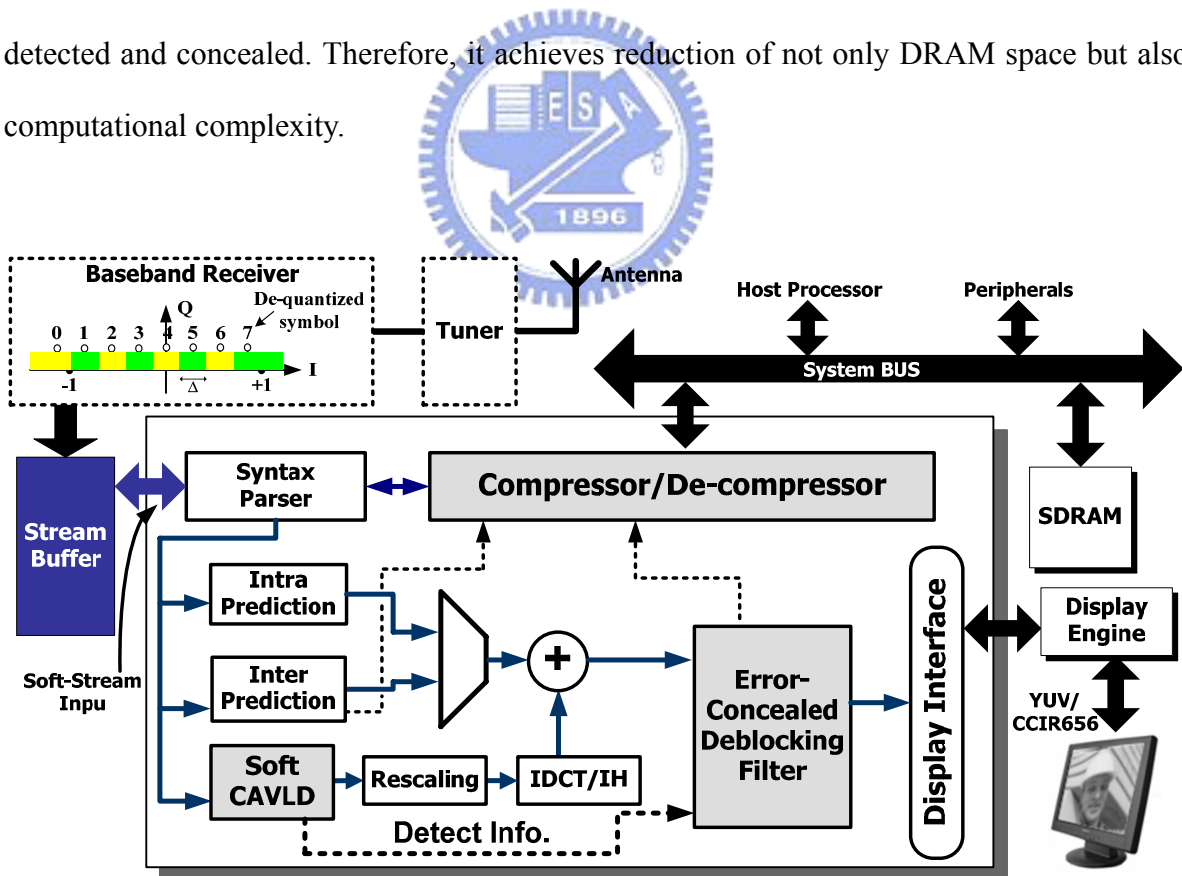


Figure 4.1: Soft-input H.264/AVC decoding block diagram.

4.3 *Soft CAVLC Decoder*

4.3.1 Soft Decoding Concept

In most applications of variable length codes (VLCs) such as MPEG-x and H.26x families, decoding is carried out bit by bit, with the input to the entropy decoder assumed to be a sequence of “hard” bits about which no soft information is available. However, in noisy environments, soft information can be associated with each information bit, either by direct use of channel observations in the case of un-coded transmission [63], or through soft-output channel decoders [64] when channel coding is applied. It is intuitive that this soft information, if it can be exploited, can be used to correct the corrupted symbols and thereby improve the performance of VLC decoding. Therefore, many researchers developed various algorithms [63][65]–[67] to explore the possibility of using soft information. Specifically, these algorithms improved the estimation of transmitted VLC symbol sequence based on reliability (soft) values, and significant gains can be achieved over hard decisions. However, those designs suffer from the large computation as well as memory storage. Furthermore, although several reduced-complexity algorithms [68]–[71] have been proposed to date, they are still unpractical for VLSI implementation. To implement soft decoding into a hardware prototype, we translate the capabilities of soft decoding from error correction to detection. Although this translation degrades the decoding performance, the computational complexity can be greatly reduced and the detected information can be further passed to post-processing procedures for the purpose of error concealments.

4.3.2 Soft VLC Decoding with Error Detection

Many researchers pay lots of attention to conceal the corrupted video content, but these methods seem to have its limitation. They often assume that video errors have been correctly located; otherwise error concealment method cannot be properly applied.

Therefore, detecting or localizing the erroneous position is of great importance, and thereby becomes our major task prior to the error concealment discussed in next sub-section. In this work, we exploit the soft information of channel observations to detect the erroneous position in a macroblock (MB) level. In particular, we modify a SISO algorithm to improve the capabilities of error detection instead of correction. In general, the SISO decoding technique [71][88] is considered as an exhaustive decoding procedure to resist the error disturbance in a noisy channel. It estimates and searches on the tree-like path in the existence of additive white Gaussian noise (AWGN). However, a main drawback of this approach is that many states have to be kept for correcting errors. It needs a great deal of memory storage and computational complexity. Hence, it is inappropriate for VLSI implementation. To avoid the penalty of large computation, we concentrate on the error detection since there is no need to keep many states for correcting errors in trace-back procedures. We only keep present states to detect that whether this video data is correct or not. In the following, we show the modified SISO algorithm to improve the error detection capabilities.

Before we address a general algorithm of the modified SISO, we use a graph representation to help the understanding of soft VLC decoding. We give a simple example to illustrate the behavior of soft VLC algorithm. 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 soft bit-stream may become $\{-0.8, -0.05, -0.2\}$ (i.e. an error occurred in the second bit). Figure 4.2 depicts the graph representation of this example. The path metric $PM(i,j)$ denotes the cumulative square difference of i^{th} symbol and j^{th} bit in each decoded symbol. The branch metric $BM(i,j)$ is the square difference between the received soft-stream and decoded codewords. Moreover, the present PM is the summation of previous PM and present BM .

Each number inside square represents the decoded bit pointer. In Figure 4.2, we have to keep 9-state to trace-back and correctly decode bitstream. However, the number of states will increase exponentially with the increment of decoded symbols. To alleviate this problem, we focus on error detection instead of correcting the errors. That is, there is no need to keep each state for trace-back procedures. Previous *PM* value will be discarded and contributes to the present *PM* value. We only keep the minimal *PM* of survival paths as our detected candidate. Afterward, these *PM* values will be recursively updated until the end of one macroblock (MB).

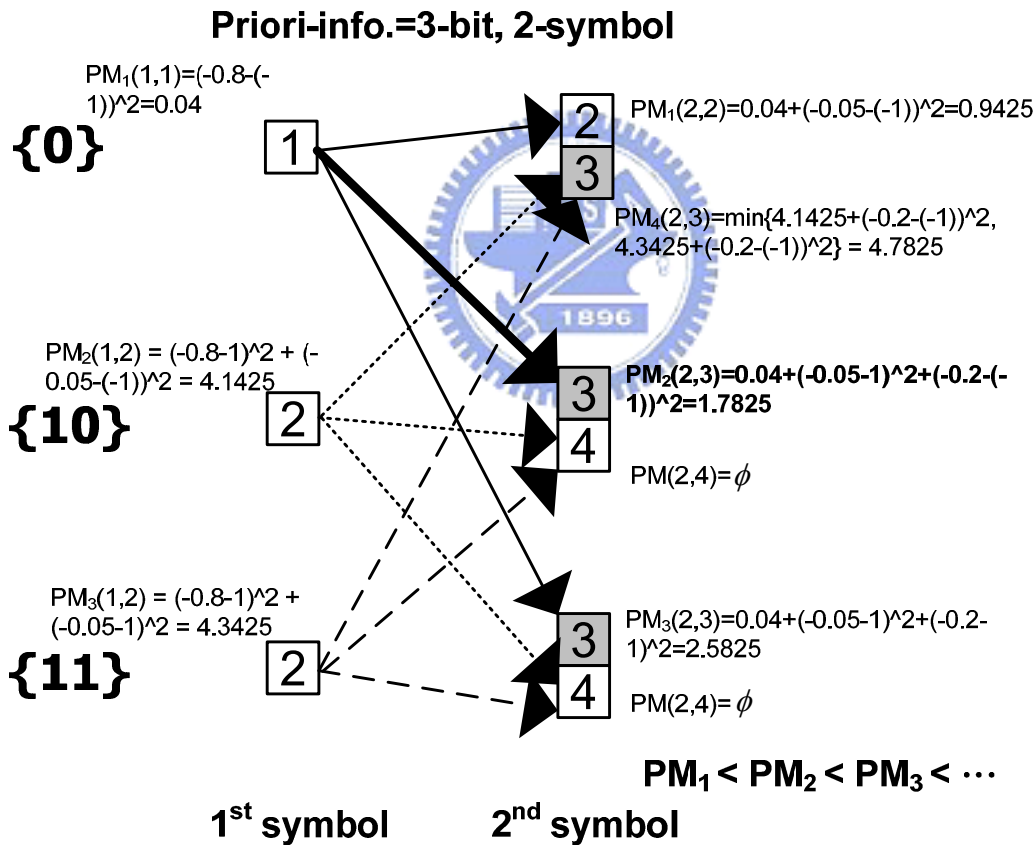


Figure 4.2: Soft VLC algorithm using graph representation.

When errors occur, a key observation is that the minimal *BM* will increase, indicating the large square difference between received soft streams and decoded codewords. Figure

4.3 depicts the behavior of minimal BM in the 1st and 2nd macroblocks of one frame. The minimal BM of 1st MB is very small because the received stream is correct while the 2nd MB is corrupted by the channel disturbance, yielding the large BM value. In addition, to enlarge the erroneous magnitude for detecting errors, we exploit the summation of BM over one MB as a measurement of error detection. A formal statement of the detection algorithm follows:

Soft Detection Algorithm:

Storage:

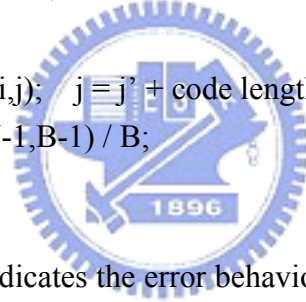
$\{i, j\}$ ({symbol, bit} index)
 $PM(i,j), BM(i,j)$ (path metric, branch metric)

Initialization:

$\{i, j\} = \{0,0\}$, $0 < i < N, 0 < j < B$;
 $PM(i,j) = 0$; $BM(i,j) = 0$;

Recursion: Compute

$PM(i+1, j') = PM(i,j) + BM(i,j)$; $j' = j' + \text{code length}$
Error Measurement = $PM(N-1, B-1) / B$;



Although the PM value indicates the error behavior, each MB has different numbers of coded bits due to the variable length characteristics. Hence, the PM has to be translated and normalized by the number of coded bits. In Figure 4.3, this MB contains B bit and the derived PM is divided by B as Eq. (4.1) listed. The threshold THR can be determined through the simulation or channel behavior [75]. Hence, the detecting information ERR_FLAG can be used as an enable signal to activate the post-processing unit such as error concealment for improving visual quality.

$$ERR_FLAG = \begin{cases} 1, & \text{if } \frac{PM}{B} > THR \\ 0, & \text{if } \frac{PM}{B} < THR \end{cases} \quad (4.1)$$

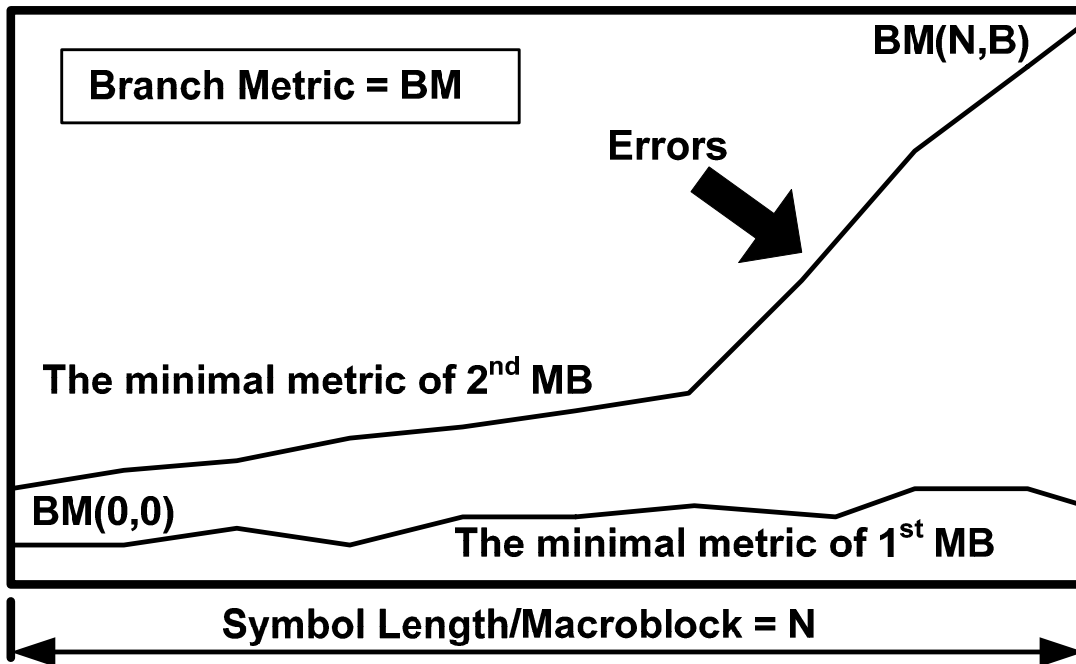


Figure 4.3: Soft VLC decoding path w/t and w/o errors.

4.3.3 Decoding Architecture

Before we address the soft decoding architecture of CAVLC decoder, we explain the CAVLC decoding flow in a system point view. Under the condition of entropy decoder in H.264/AVC, the input bit-stream will not always feed through CAVLC decoder since sufficient processing cycles are required. Therefore, the issue and hold signal is located to reduce the internal buffer and improve the operation throughput. In particular, we exploit single-bit hold signal to indicate the bit-stream alignment instead of multi-bit code-length to achieve low cost implementation. In Figure 4.4(a), a data transfer is accomplished when the receiver doesn't hold data input and transmitter sends issue signal for the notification in advanced. Hence, this handshake mechanism deals with the hold of receiver and issue of transmitter. This scheme is described as follows:

- Normally, the transmitter sends an issue signal to the receiver first. A data transfer is completed when the transmitter disables a hold signal.

- However, if the issue is disabled, the receiver must hold the request and wait the required data.

Instead of input register and barrel shifter, we use issue-hold signal to reduce cost and realize system integration more smoothly by the handshake structure. In Figure 4.4(b), H.264/AVC contains fix length, universal and context-adaptive VLC decoder in Extended profile. UVLC and CAVLC are the critical part in terms of system performance, especially in high bit-rate or low power video applications. The issue signal of entropy decoder will activate when 2×2 , 4×4 , 8×8 or 16×16 coefficient encounters. Further, the hold signal can be realized with an OR operation launched by UVLC, CAVLC, CABAC...etc. The input of OR operation can be connected with other modules in H.264/AVC or the entropy decoder of different video standards. Each modules enable hold signal when the operation module cannot process a great data for a specified cycles. In general, the video stream buffer is realized in most of video devices to support buffering process. After that, the output signal including syntax and residual data will validate when the valid signal ties to high.

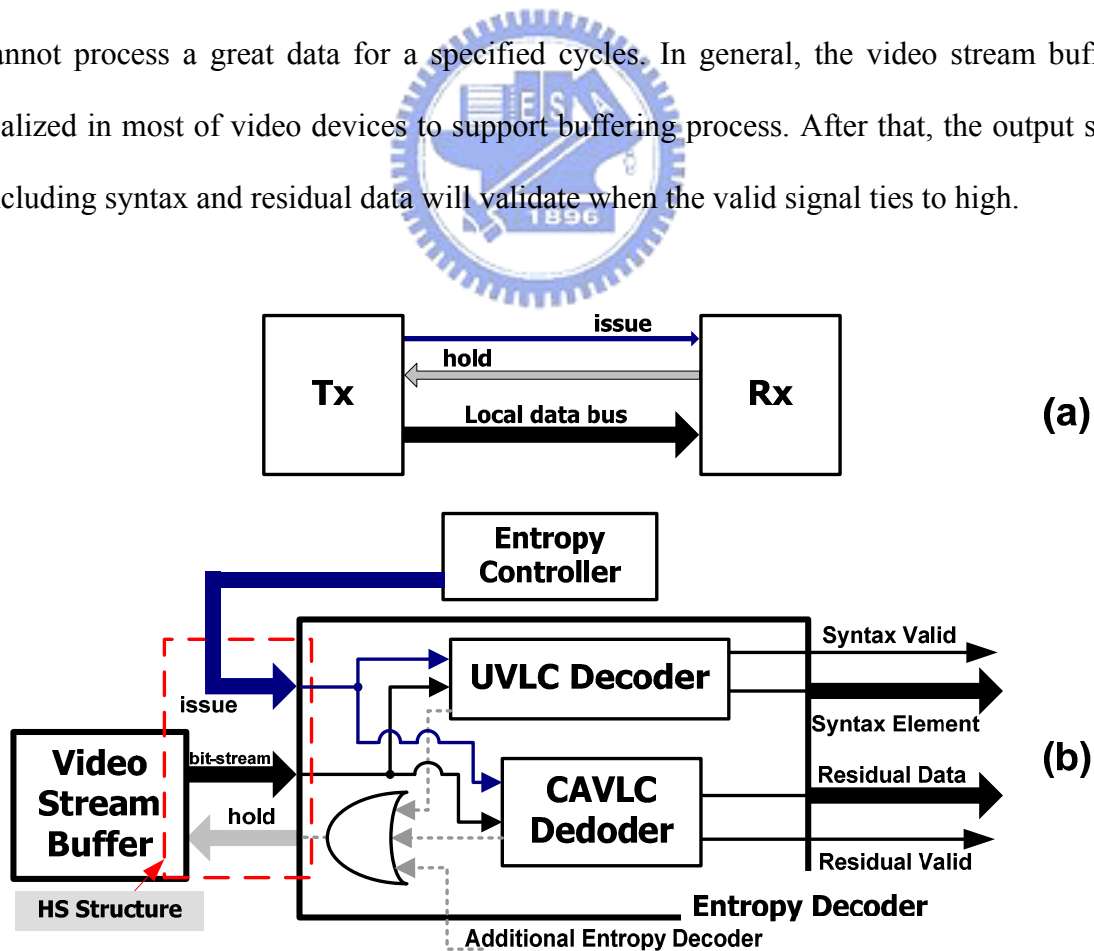


Figure 4.4: The (a) handshake structure and (b) entropy decoder in H.264/AVC.

Figure 4.5 describes the proposed soft CAVLD block diagram with embedded error detection capability. The gray portion indicates the modifications to support the error detection capability. The soft CAVLD consists of four main functional units: codeword partitioning unit, soft VLC decoder, error detection, and output buffer. First, a codeword partition method will be discussed in next sub-section and soft VLC decoder implements the decoding behavior discussed in the previous sub-section. We don't apply soft decoding on the *level* and *Trailing1_sign* since they can be decoded through 1's detection and fixed length decoder respectively. Second, error detection realizes Eq. (4.1) that extracts the *PM* value from soft decoder to decide whether this MB is corrupted or not. Third, the output buffer converts the data in a parallel fashion. In the following, we present two architectural breakthrough involved in this soft CAVLC decoder. One is the codeword partition method and the other is fully-parallel architecture for reducing the soft decoding complexity.

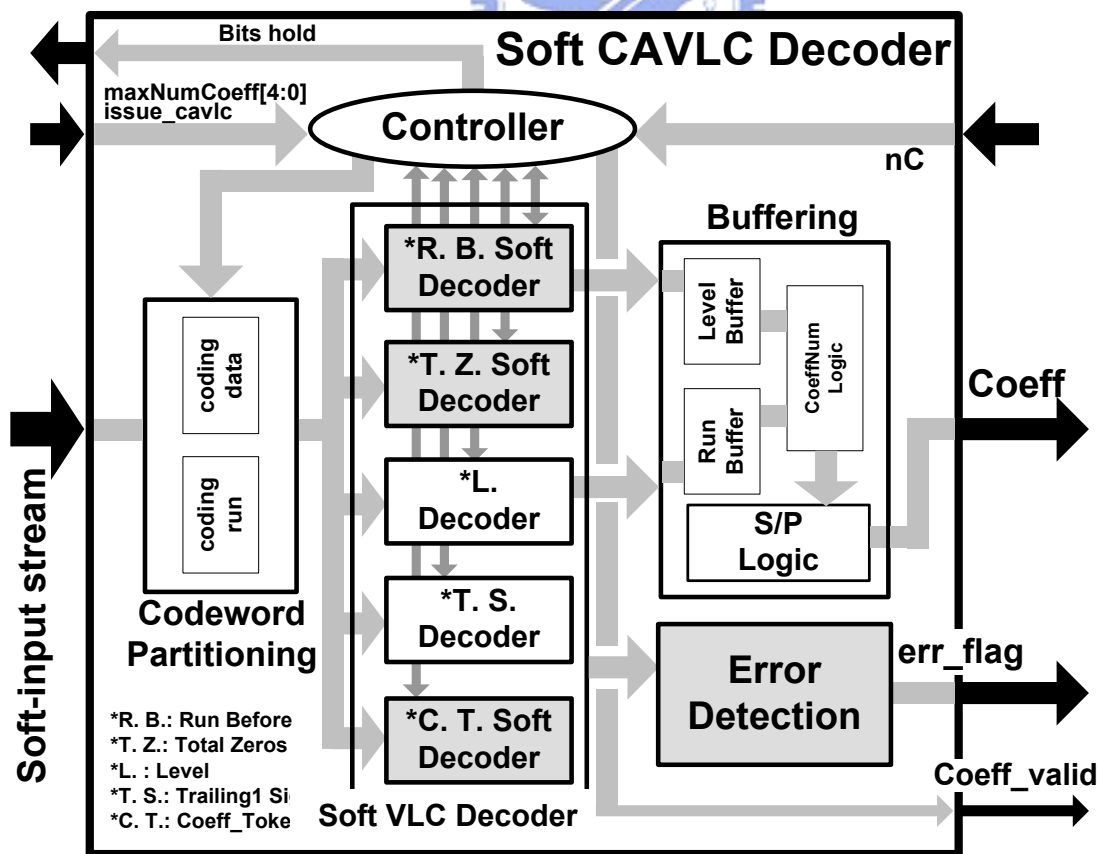
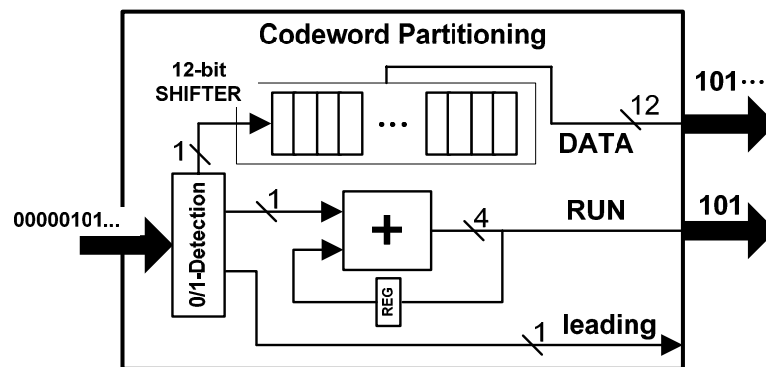


Figure 4.5: Soft CAVLD block diagram.

4.3.3.1 Code-Word Partitioning

The codeword partitioning can be considered as a pre-processing module in front of VLC decoder. Based on the characteristics of leading-0/1 in each VLC codeword, we simply partition bit-stream into the coding *RUN* as well as *DATA*. In Figure 4.6(a), the coding *RUN* extracts the number of zeros or ones that it detects, and assigns the remainder of bit-stream as the coding *DATA*. We use coding *DATA* instead of total codeword to reduce the VLC decoding complexity since only coding data is needed to be uniquely decodable.

We show a simple example in Figure 4.6(b) for the illustration of codeword partitioning. We assumed that the input bit-stream is ‘00000101’ and the number of leading-0 can be obtained easily (i.e. five). After that, we extract the remainder bit-stream and dispatch it to the 12-bit shifter. Furthermore, we hierarchically target the decoded symbol from the given nC and *RUN*. The 4-bit zero counter and 12-bit shifter have to be reset when the decoded symbols are obtained. As a result, we search the codeword in a small group (i.e. gray region of Figure 4.6(b)) instead of large entry such as 62 in a *Coeff-Token* coding table [1]. Therefore, based on the proposed codeword partitioning, we can easily decode bitstream by the partitioned coding *RUN* and *DATA* and thereby reduce the complexity in the module of VLC decoder.



(a)

$0 \leq nC < 2$		
Group Info.	RUN	DATA
Group No. = 4 (Leading-0)	3'd4	11
⋮	⋮	⋮
Group No. = 5 (Leading-0)	3'd5	111
		110
		101
		100
⋮	⋮	⋮

(b)

Figure 4.6: The (a) codeword partitioning and (b) partial *Coeff-Token* coding table [1].

4.3.3.2 Fully-Parallel Design

To meet the real-time transmission over an error-prone environment, we exploit a fully parallel architecture to improve the decoding throughput. Figure 4.7(a) shows the soft VLC decoding block diagram. Specifically, each *BM* unit simultaneously calculates the square difference between received soft streams and decoded codewords. It is performed in a parallel fashion without cycle penalties. After that, the compare unit (CMP) searches the minimal *BM* as the detected candidate. The minimal *BM* is accumulated in each clock cycle and thereby stored into *PM* registers. When the decoding index reaches the end of MB, the *PM* value is sent into error detection block to judge whether this MB is correct. As for the *BM* unit, Figure 4.7(b) describes the *BM* unit in case of a decoded codeword {000101}. The *BM* unit calculates the square difference between received 16-level soft streams and decoded code-words. Hence, it executes the subtraction, multiplication, addition, and constant division to obtain the *BM* value.

Although the hardware complexity of soft CAVLD is considerably high, this is the first

trial to realize the soft decoding using hardware building blocks [89]. It is synthesized using 0.18 μm Artisan cell library by Synopsys Design Compiler. Under the timing constraint of 20ns, the synthesized gate counts approximately reach 80k. In the future, this area penalty can be greatly reduced by compromising the area and processing cycles or merging the identical codewords to reduce the number of *BM* units.

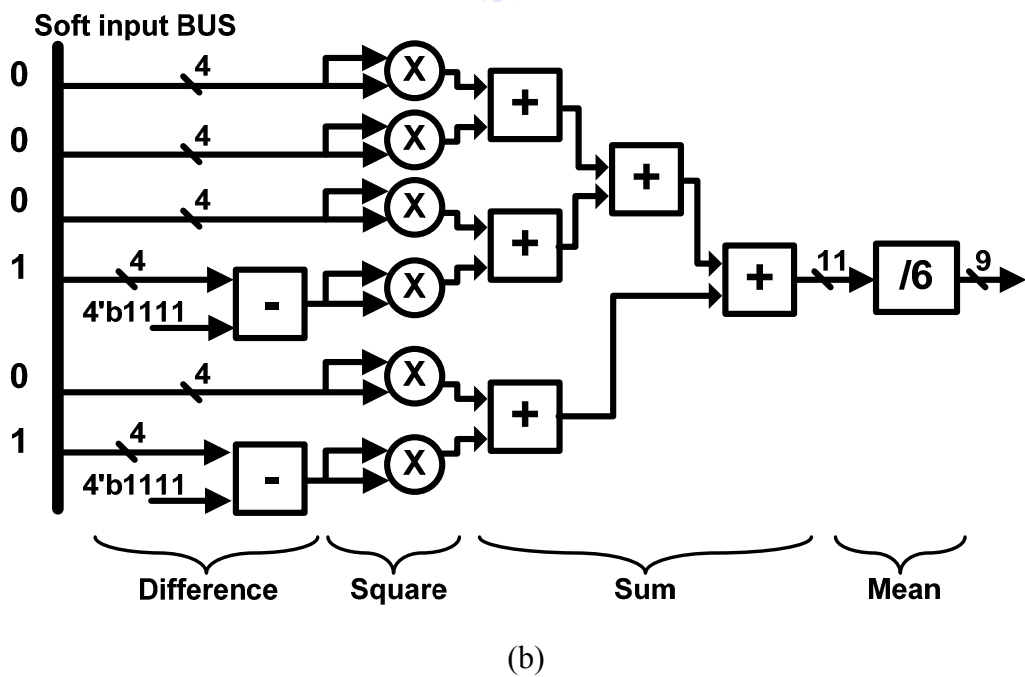
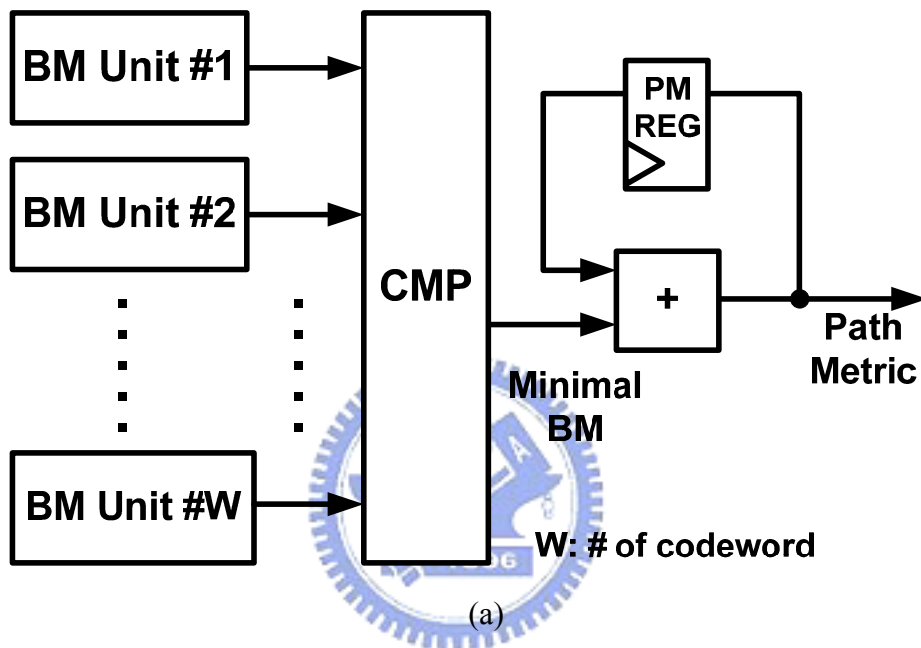


Figure 4.7: (a) Soft VLC decoder block diagram and (b) associated *BM* architecture.

4.3.4 Performance Evaluation

In Figure 4.8, we verify the proposed soft CAVLD coupled with error detection capability over the AWGN channel using BPSK modulation. First, the input sequence is encoded by H.264/AVC Extended profile with a data partition. In the data partition mode, we have assumed that the texture part, composed of a sequence of VLC code-words, is corrupted by AWGN. The other parts are of error-free due to the protection of channel coding. This assumption can be achieved by exploiting the unequal error protection (UEP) and rate-compatible punctured convolutional codes (RCPC codes [90]). Furthermore, the coded stream is transmitted via BPSK modulation. 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} [138]. The bit number q is used to quantize the symbol and determined by channel and application. After the channel deterioration, the corrupted stream is determined by the number of de-quantization levels (hard: 2-level; soft: N-level). Hence, we use the soft output of quantizer as our input bit-stream of soft CAVLD. The soft CAVLD can overlook the bit-errors from the quantizer of physical layers since we assumed that an UDP-Lite protocol [91] is applied to our simulation model. Specifically, we partition all of data transaction into the 5 layers of OSI (Open System Interconnection) model in Figure 4.9. 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 N-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 H.264/UDP-Lite/UEP/AWGN in the next paragraph.

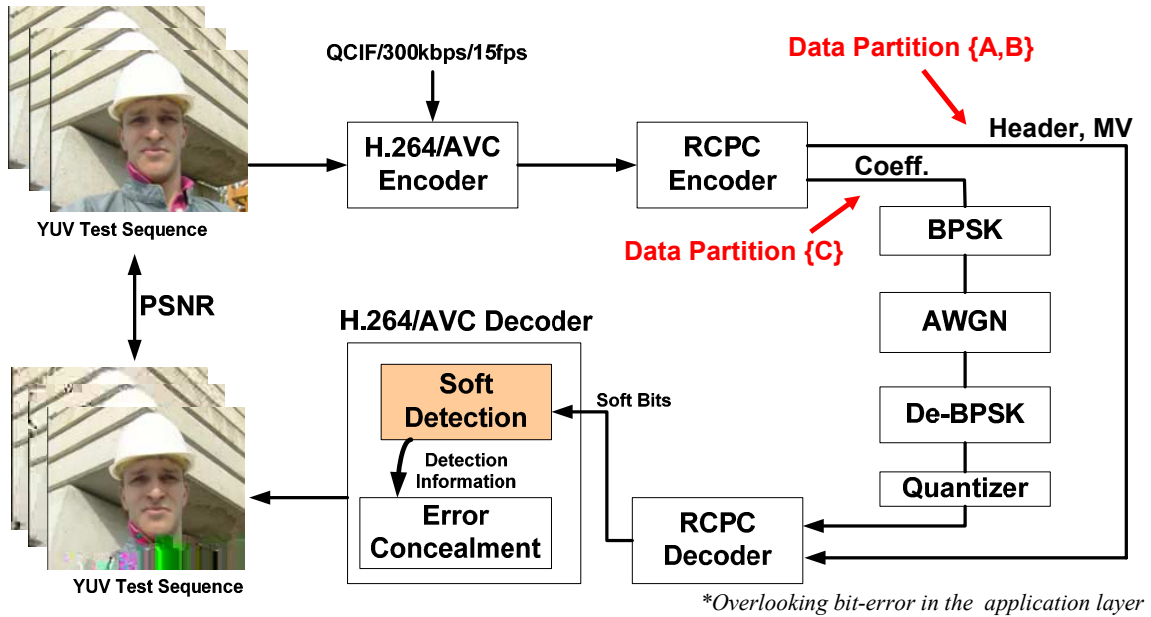


Figure 4.8: The overall simulation environment of the soft detection over H.264/AVC.

Contents	OSI Layers
Application – MPEG	Application Layer
Transport – UDP or UDP-Lite	Transport Layer
Network – IP	Network Layer
UEP – RCPC Codes	Link Layer
Soft Information	Physical Layer

Figure 4.9: Overlooking bit errors in the application layer.

The simulation results exhibit that the errors can be early detected through the proposed soft decoding algorithm. On the other hand, the errors can be detected by the abnormal syntax parsing or unknown codewords. It has been named as hard detection [80] in this dissertation. To clarify the performance between hard and soft detection, we consider the first I-frame of foreman with QCIF resolution (i.e. 99 MBs) as a test benchmark. Figure 4.10 shows the path metric over bits (i.e. PM/B) in terms of MB index. According to Eq. (4.1), large PM/B indicates that errors may occur in this MB, resulting in large PM/B values. In Figure 4.10, there is a peak in MB #83. In other words, we declare that the following

MBs (including MB #83) are corrupted due to the error propagations. However, the detected index is 94 (>83) through the hard detection. That is, the duration between MB #83 and #93 are neglected and cannot be concealed by post-processing units. Due to the improvement on error detection, the associated decoded visual quality has been shown in Table 4.1. We assumed that simple error concealment is applied and it will be activated when the detection algorithm notices that errors occur. The Table 4.1 shows that more than 1dB of PSNR improvement can be gained when the BER and THR equals 2.7×10^{-3} and 18, respectively.

To quantify the performance of error detection, we exploit a metric of the error distance that indicates the distance between the exact error-position and detected one. Table 4.2 shows the error distance of three different video sequences under 10^{-3} and 10^{-4} of BERs. The proposed soft detection has the small distance in average. That is, it can early detect the corrupted MBs as compared to hard detection [80]. Hence, these MBs can be concealed from neighboring pixels before displaying on monitors.

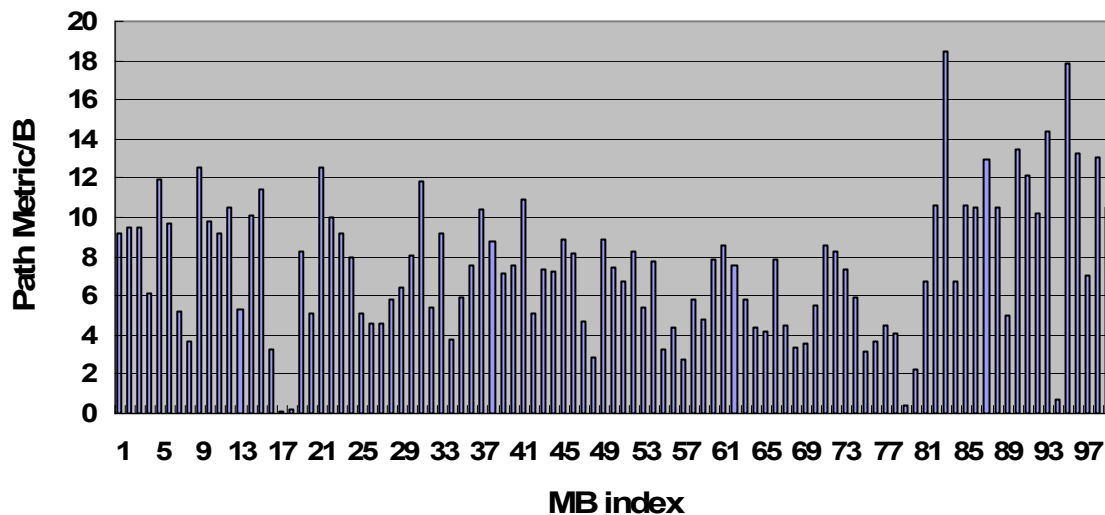


Figure 4.10: A histogram of PM/B in terms of macroblocks (MB).

Table 4.1: Subjective and objective visual comparison.



AWGN+BPSK (BER=2.7×10⁻³, THR=18)		
	Hard detection [80] + EC	Proposed + EC
Subjective Quality		
Objective PSNR	23.93dB	25.34dB

Table 4.2: Performance measurement of detection capabilities.

Error Distance (unit: average no. of MBs)			
QCIF, all I frames		Hard detection [80]	Proposed [89]
BER=10 ⁻³	Foreman	7.1	3.5
	Suzie	14.6	4.8
	Akiyo	13.7	3.3
BER=10 ⁻⁴	Foreman	10.4	7.6
	Suzie	12.8	7.5
	Akiyo	12.5	3.2

4.4 Error-Concealed Deblocking Filter

4.4.1 Design Background

As mentioned in Chapter 4.2, the problem of lost data in block-coded images due to imperfect communication channels needs to be solved. Error concealment (EC) intends to

ameliorate the impact of channel impairments by utilizing neighboring information in order to provide subjectively acceptable restoration of damaged picture regions. Specifically, the EC scheme attempts to recover the corrupted macroblocks (MBs) by exploiting pixel data from spatially [93]–[95] or temporally [96]–[98] adjacent blocks. The temporal schemes conceal the corrupted MBs by exploiting the data in the adjacent frames while spatial schemes hide the corrupted regions by the neighboring pixels in the current decoded frame. In general, a successful spatial interpolation is necessary for hiding the effect of missing blocks in still images and video frames. Temporal interpolation, or replenishment, by itself is not always adequate for concealing errors in video sequences. This is especially true for video sequences with irregular motion, abrupt scene changes, and intra-coded image frames. To the viewer, a poor spatial concealment of erroneous regions leads to the error propagation in the subsequent frames. Hence, compared to temporal EC, spatial EC is of great importance and challenges. Thereafter, we focus on the spatial EC for concealing corrupted region of image frames.



Using a deblocking filter for improving visual quality [92] is more or less helpful in the block-based video transmission over an error-prone environment. The reason is that the functionalities of deblocking filters are similar to that of spatial error concealment since both modules smooth the block boundaries by a pre-defined interpolating procedure. In particular, the error concealment will be disabled when the bit-streams are of error-free. On the contrary, the deblocking filter can be turned off when the error has been detected from soft CAVLC decoder. Therein, deblocking filter can be replaced with error concealment module for improving visual quality. In other words, both deblocking filter and error concealment will not activate simultaneously. Hence, the goal of our proposal is to realize an area-efficient design to combine both deblocking filter and error concealment. In the following, we develop a new error concealment method which can be easily integrated into deblocking filter of H.264/AVC.

4.4.2 Error-Concealed Deblocking Filter (ECDF)

To improve the hardware utilization between deblocking filter and error concealment, we develop an error-concealed deblocking filter (ECDF) in our error-robust video decoder. The detailed block diagram is depicted in Figure 4.11 which consists of three main components as depicted in shaded regions. The detected information, the output of soft CAVLD, plays a key role to switch the functionality and decides that ECDF is operated on either error concealment or deblocking filter mode. Moreover, edge detection, replacement, and smoothing procedures can be considered as add-on modules for changing the operating process when encountering erroneous pixels. Edge detection fetches the neighboring and decoded pixels from slice memory and pixel buffers respectively to predict the edge characteristics. Replacement receives the edge information and determines the replaced pixel. A concealed strength module determines the filtering strength and replaces the bS defined in H.264/AVC [1]. In the following, we will address each module in details.

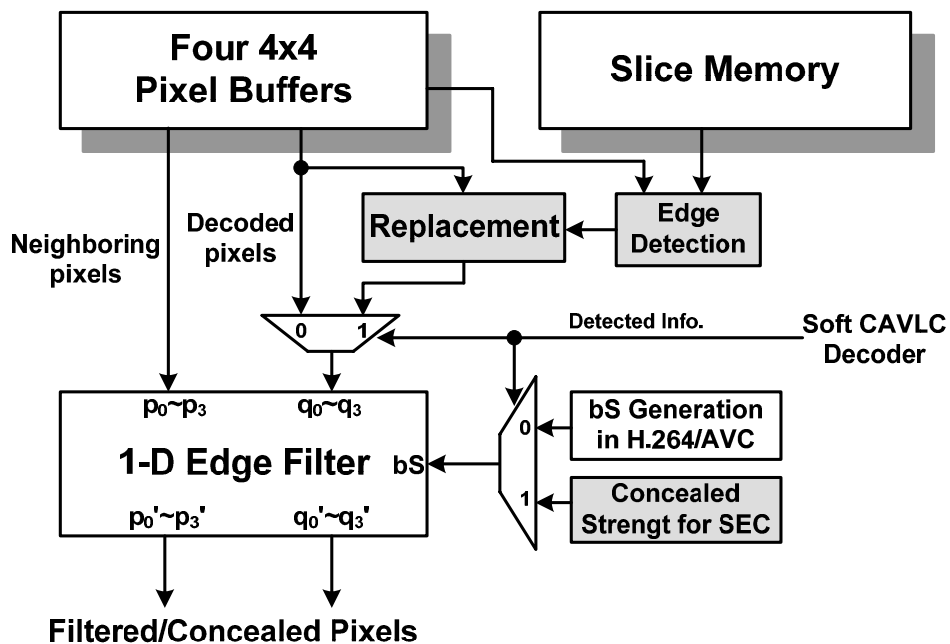


Figure 4.11: The block diagram of the proposed ECDF.

4.4.2.1 Edge Detection

In general, a directional interpolation (DI) [95] is an important process to spatial error concealments. One important step for use of directional interpolation is to correctly find the edge trend that goes through the distorted area. To this end, *Sobel* [107] edge detector has been chosen due to its circularity property, which gives more accurate angle estimates over the standard gradient operator. More accurate but computationally cumbersome operators, such as the canny edge detector [108], cannot be used because of real-time and VLSI implementation constraints. Here, *Sobel* mask is shown in Eqs. (4.4)-(4.6) and used to determine the magnitude of gradient $|G|$ and edge slope of existing edge in the neighboring 4×4 sub-blocks. S_x and S_y indicate the *Sobel* operator in horizontal and vertical directions while F represents a 3×3 pixel mask in one frame. After that, we use the information of gradient to predict the edge trends of corrupted 4×4 sub-blocks.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad F = \begin{bmatrix} p_{i-1,j-1} & p_{i,j-1} & p_{i+1,j-1} \\ p_{i-1,j} & p_{i,j} & p_{i+1,j} \\ p_{i-1,j+1} & p_{i,j+1} & p_{i+1,j+1} \end{bmatrix} \quad (4.4)$$

$$G_x = S_x^T \cdot F, \quad G_y = S_y^T \cdot F \quad (4.5)$$

$$|G| = \sqrt{[G_x]^2 + [G_y]^2}, \quad slope = \frac{G_y}{G_x} \quad (4.6)$$

4.4.2.2 Replacement

Based on the aforementioned edge information, we rank it as four different kinds of edge degrees: 0° , 45° , 90° , and 135° . These degrees determine the corrupted pixels which are replaced by neighboring ones. Figure 4.12 depicts the replacement of corrupted pixels in the gray regions. Because the corrupted pixels have been detected by soft CAVLD, we first replace it with neighboring pixels based on the received edge information. Then, the

replaced pixels feed into the edge filter to smooth the block boundary and improve the visual quality.

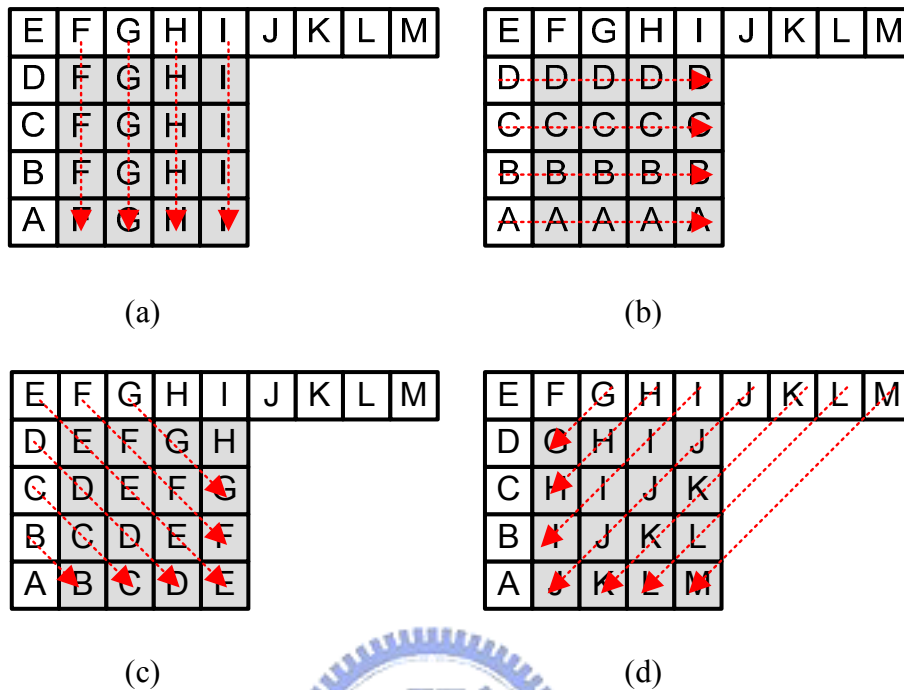
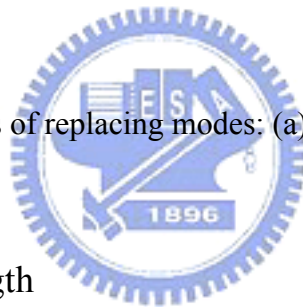


Figure 4.12: Four kinds of replacing modes: (a) 90° , (b) 0° , (c) 45° , (d) 135° .



4.4.2.3 Concealed Strength

When encountering erroneous pixels, the boundary strength is fixed at strong mode prior to the 1-D edge filter. We name the boundary strength of erroneous data as concealed strength. As we know, boundary strength (i.e. bS) defines the filtering strength and impacts the number of filtering taps. There are two filtering modes defined in H.264/AVC: strong and weak modes. In particular, an edge filter in strong and weak mode relates to six and four pixels of each 4×4 sub-block boundary, respectively. In other words, edge filter in strong mode obtain better smoothing results due to large number of filtering taps. Therefore, we let the boundary strength (bS) as a strong mode when the soft CAVLC decoder found that the decoded pixels is corrupted.

4.4.3 Performance Evaluations

We verify the proposed ECDF over the JM9.8 simulated platform with flexible macroblock ordering (FMO) modes. FMO modifies the way how pictures are partitioned into slices and macroblocks by utilizing the concept of slice groups. Each slice group is a set of macroblocks defined by a macroblock to slice group map, which is specified by the content of the picture parameter set and some information from slice headers. Figure 4.13 depicts the corrupted frame using FMO with a checker-board type of mapping. By FMO, a post-processing unit, error concealment, can easily improve the visual quality through the neighboring pixels.

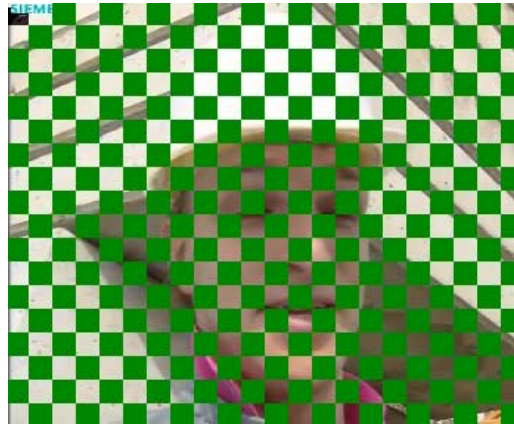


Figure 4.13: A corrupted frame when using FMO with checker-board type.

In Figure 4.14, we consider a “foreman” test sequence with CIF resolution and QP=28 in order to perceive the contrast between traditional bilinear interpolation (BI) and the proposed ECDF. Simulation results show that the proposed ECDF additionally gains 1.34dB against the BI method. Moreover, the ECDF shares the pixel buffers and edge filter in Figure 4.11. Under a 100MHz of working frequency in 0.18 μ m CMOS process, the synthesized gate counts are reduced to 22.74K which is about 70% of original design without exploiting any resource sharing. Overall, the ECDF is not only compatible to prevalent deblocking filter defined by H.264/AVC but also achieves low cost

implementation with comparable PSNR performance.



Figure 4.14: Subjective quality comparison between the (a) proposed algorithm and (b) BI.

4.5 *Embedded Compressor/De-compressor*

4.5.1 Literature Reviews

Recently, with continuing increases in high-quality video playback requirements, the increased memory capacity becomes a primary penalty in the design of mobile multimedia system. The memory stores image frame and is usually implemented by DRAM. However, the introduced memory power consumption is so high that many researchers [81]–[87] pay much attention on the compressor and de-compressor for reducing memory capacity as well as power dissipation. In terms of characteristics, these compressors can be divided into two classes: on-the-fly [81]–[83] and embedded [84]–[87] compression. The on-the-fly compression can be considered as a post processing unit and the embedded compression is involved in a typical video coding/decoding flow. In this sub-section, we aim at an embedded compression design that is heavily involved in the H.264/AVC video decoding system. Although an existing design [86] has realized an embedded compression for H.264/AVC, it introduces two problems in practical implementation. First, it exploits a

complex CAVLC and intra compensation to share the resource, leading to high complexity and the problem of the hardware scheduling. Second, a highly compressed bitstream stored in frame buffers suffers an addressing problem when the modules require the reference macroblock in frame buffers under a given decoding index and motion vector. Moreover this highly-compressed data will adversely impact the quality when channel deteriorates. On the other hand, in terms of compressed algorithm, the aforementioned techniques can be partitioned into two groups again: lossy and lossless compression. The performance of lossless compression is limited while lossy compression achieves higher compression ratio but leads to quality degradation due to the error propagation (i.e. drift effect [84]). However, Bourge *et al.* [100] present a re-compression method with graceful quality degradation by measuring a metric of JND (Just Noticeable Difference [101]). The ideal JND provides each signal being represented with a threshold level of error visibility, below which reconstruction errors are rendered imperceptible. Although a graceful degradation is tolerant for the end-users by JND, it is still hard to convince all viewers of this quality acceptability. Hence, to optimize the performance between lossy and lossless methods, a recent research [87] proposed a multi-mode compression method by adopting a set-partitioning in hierarchical trees (SPIHT [102]) algorithm to support both lossy and lossless compression. Because a SPIHT algorithm features to simply reach lossy/lossless compression, fixed compression ratio, rate and quality control, it has been adopted for a purpose of frame re-compression. However, the challenge of this algorithm is the numerous memory cost and computational power. Although [87] presented an efficient architecture, it cannot reach high-definition real-time compression/decompression due to the extensive processing cycles.

Until now, we first review the existing methods due to a great diversity of existing algorithms. In the following, we focus on the embedded compression to support both lossy and lossless features in order to highlight our contributions. The goal of this design is to

implement low-complexity VLSI architecture for reducing DRAM capacity. Moreover, it provides a power-aware [99] feature which scales DRAM power consumption in response to changing operating conditions. As a result, it adaptively modifies the compression behavior to balance the performance between compression ratio and battery lifetimes. To meet the design goal of this chapter, this design further considers the channel behavior when compressing frame pixels. That is, it is robust to channel disturbance by the received pixels and detected information via the aforementioned ECDF and soft CAVLC decoder respectively.

4.5.2 Power-Aware and Error-Robust Features

Figure 4.15 depicts the block diagram of the proposed compressor/de-compressor embedded in H.264/AVC video decoding system. Two 4MB synchronous DRAM (SDRAM) modules are exploited for writing decoded data and reading reference data reciprocally at the same time. To cut the frequency of data transaction between internal modules and external SDRAM, a compressor receives filtered pixels and thereby encodes them into a reduced form. A de-compressor fetches the compressed data from SDRAM and performs decoding procedures to reconstruct the pixels for motion compensation. Moreover, to enhance the power-awareness on SDRAM modules, the host processor notifies the compressor/de-compressor to change the power modes when the battery monitor unit detects the voltage drop of battery. Specifically, we adopt a truncation and insertion scheme to the least significant bit of frame pixels for changing the compression ratio according to the battery lifetime. On the other hand, the proposed compression receives the erroneous information from the soft CAVLD or ECDF for reducing the compressed complexity. In particular, this information notifies the compressor of the error occurrence. After that, erroneous skipping method skips erroneous portion of a frame to reduce the computational

complexity while erroneous padding method inserts the neighboring pixel to reconstruct the concealed data. In sum, the proposal not only reduces the external memory space requirement but also adapts the memory power consumption to battery status and eliminates the computation in the presence of errors within one frame.

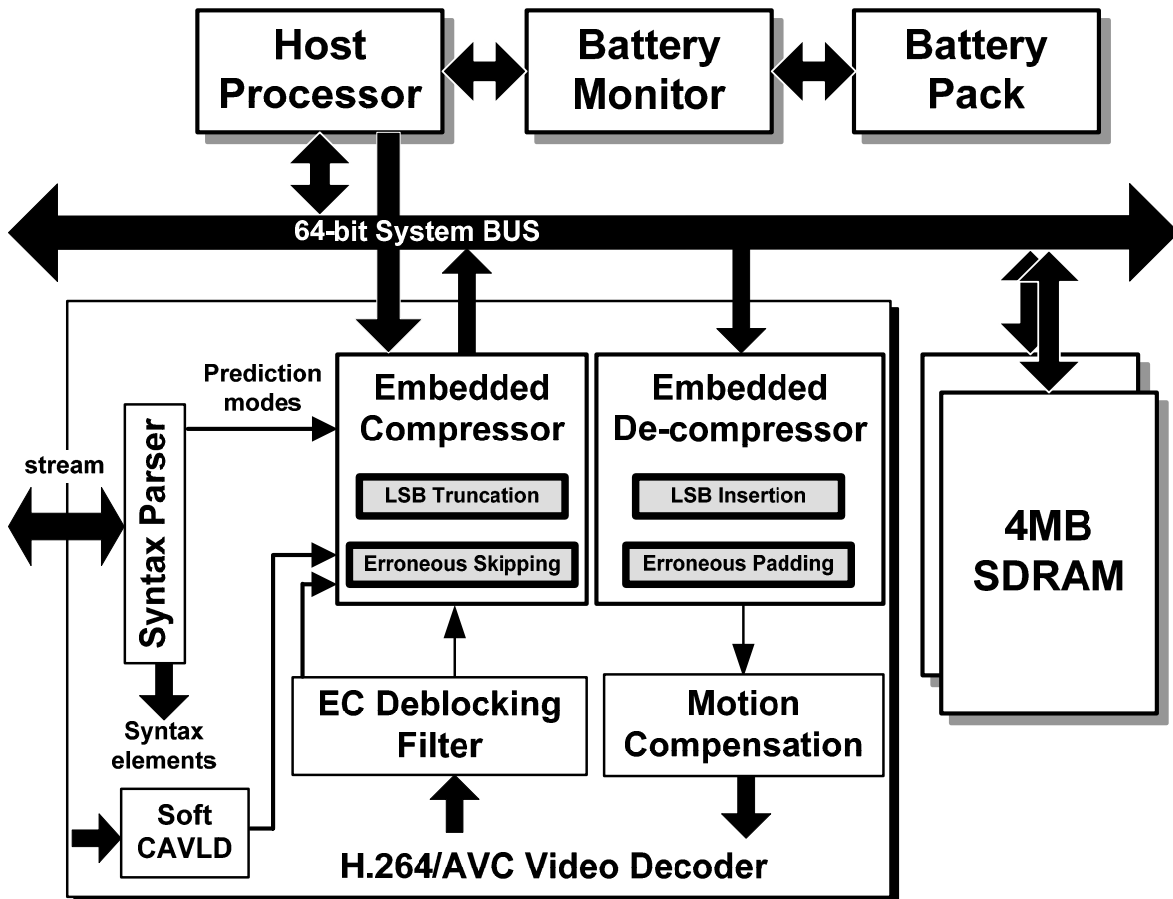


Figure 4.15: System block diagram of compressor/de-compressor.

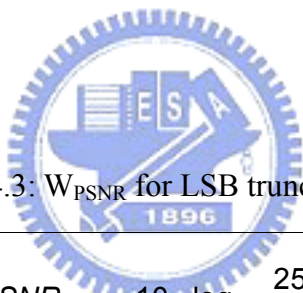
4.5.2.1 Power Awareness by Least Significant Bit (LSB)

Truncation/Insertion

We adopt a simple LSB substitution [103] to not only support lossy/lossless features but also achieve power awareness in the frame re-compression design. In general, the frame re-compression techniques can be divided into lossy and lossless methods. A lossy

compression greatly improves the performance but sacrifices the visual quality while a lossless compression retains the image quality but the performance improvement may be very mild. Hence, the contradiction exists between the compression ratio (bits/pixel) and visual quality. To make a better compromise, we choose an LSB truncation scheme to support both lossless and lossy compression. Specifically, the number of truncated LSB (i.e. variable k in Table 4.3) adversely impacts the image quality. Lossless compression can be considered as a special case when k is fixed at zero. In Table 4.3, Chan *et al.* [103] summarized the performance degradation in the worst case. Note that these W_{PSNR} are different from traditional PSNR metric which is measured by comparing the truncated frames with original raw frames. Here, W_{PSNR} considered the worst case based on the equation in the 1st row of Table 4.3. Therefore, the quality degradation is not less than that in the worst case.

Table 4.3: W_{PSNR} for LSB truncation [103].



$$PSNR_{\text{worst}} = 10 \times \log_{10} \frac{255^2}{(2^k - 1)} \text{ (dB)}$$

k	1	2	3	4	5
W_{PSNR} (dB) ¹⁴	48.13	38.59	31.23	24.61	18.3
bits/pixel	7-bpp	6-bpp	5-bpp	4-bpp	3-bpp

In addition to the lossy/lossless features, the LSB truncation scheme also adapts compression ratio (bits/pixel or bpp) in Table 4.3, resulting in scalable data bandwidth as well as DRAM power consumption when power requirements change. It is advantageous to implement compressor/de-compressor with DRAM power scalability through the variable

¹⁴ W_{PSNR} : PSNR in the worst case

number of truncated bit plane (i.e. variable k). Hence, considering an H.264/AVC decoded video, our design provides memory power-awareness and scales memory power through variable k for graceful quality degradation. On the other hand, because the truncated errors propagate into subsequent frames through motion compensation, we exploit a post-upgrade equation to alleviate the quality degradation, where Eqs. (4.2) and (4.3) represent truncation and insertion procedures in compressor and de-compressor respectively. In Eq. (4.2), the pixel A is defined as filtered raw pixels while pixel a_2 represents the truncated pixel written into SDRAM. Considering the k -bit LSB truncation case, if the truncated errors (i.e. $A \% 2^k$) are larger than one half of 2^k , the pixel A rounds the k^{th} bit in order to decrease the truncation errors. In the de-compression side of Eq. (4.3), we insert k zeros into the LSB of a_2 . The detailed results of truncation errors will be reported in Chapter 4.5.5.

$$\begin{cases} a_1 = ((A \% 2^k) \geq 2^{k-1}) ? A + 2^k : A; \\ a_2 = a_1 \gg k; \end{cases} \quad (4.2)$$

$$a = a_2 \ll k; \quad (4.3)$$



4.5.2.2 Error Robustness by Erroneous Skipping/Padding

We employ an erroneous skipping/padding technique to cope with the corrupted frame in the compression/de-compression module. Figure 4.16 demonstrates the behavior of the erroneous skipping/padding methods. Considering a corrupted frame in Figure 4.16(a), errors occur in MB index k and propagate into the following MBs of this frame. We assumed that the erroneous position has been correctly detected by soft CAVLC decoder in MB levels. Therefore, there is no need to re-compress the corrupted pixels in the compression side. The compressor skips corrupted data for reducing the complexity in the presence of errors as Figure 4.16(b) illustrates. Moreover, to correctly re-construct the pixel

data in the de-compression side, we additionally write the skip information into buffers. On the other hand, the compressed data are read from external SDRAM and thereby dispatched into de-compressor or pixel padding according to the skip information. Figure 4.16(c) depicts the de-compression behavior. The pixel padding module references the concealed algorithm in ECDF and reconstructs the skipped pixel from SDRAM. Both compressor and de-compressor are performed in MB levels for further system integration.

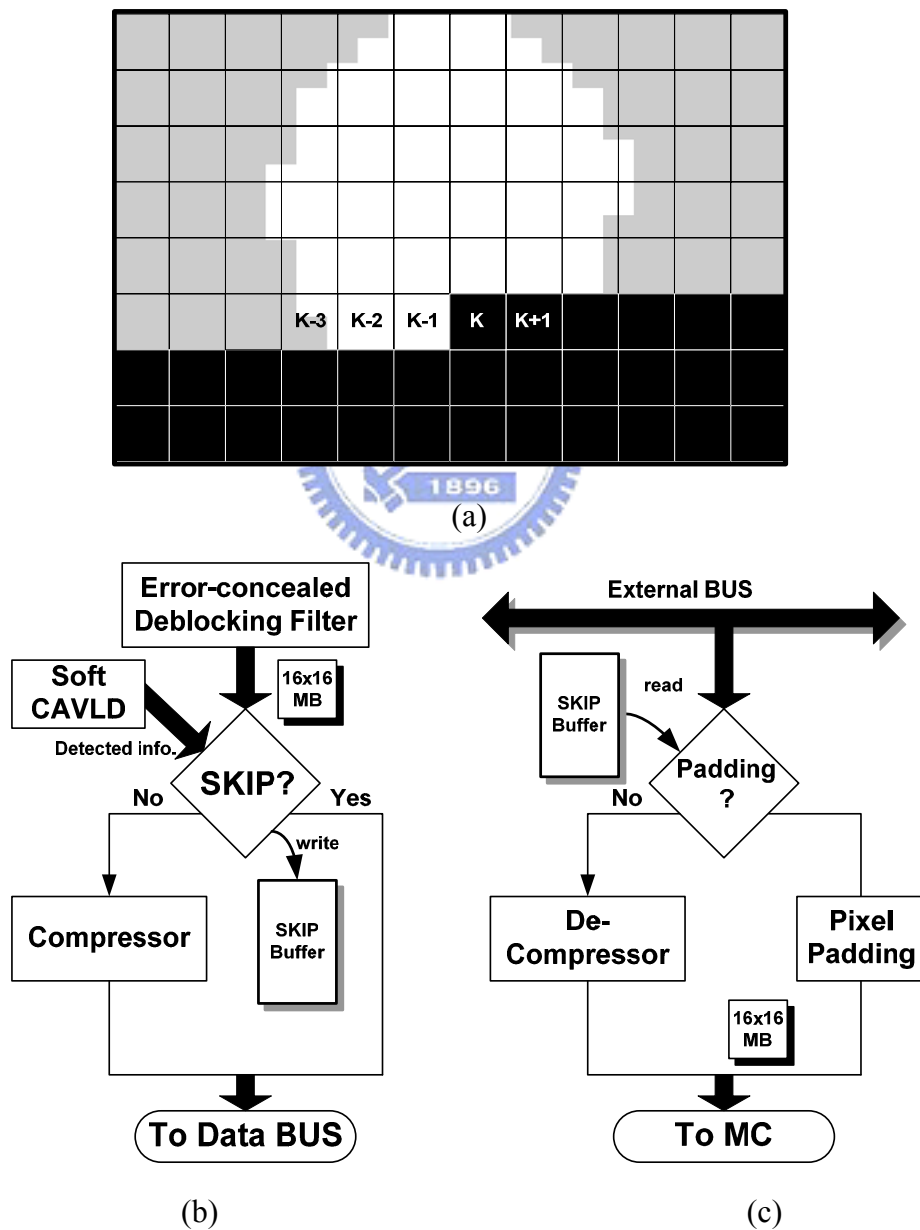
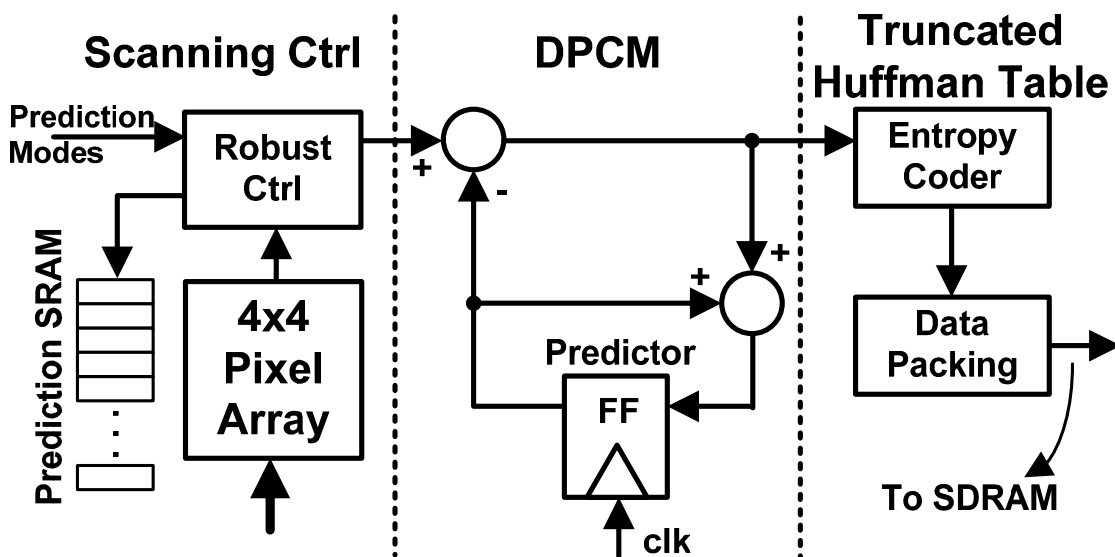


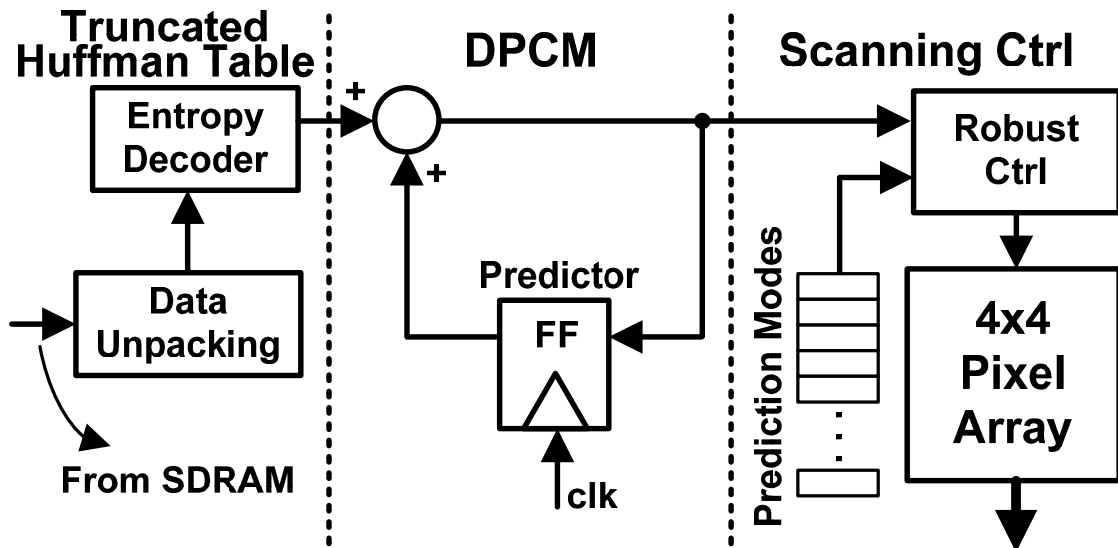
Figure 4.16: (a) a corrupted frame and error-robust (b) compressor and (c) de-compressor.

4.5.3 New Compressor/De-compressor for H.264/AVC

The objective of this design is to compress the pixel data for reducing external memory space based on the visual, intra-pixel, and coding redundancy. First, in the visual redundancy, we adopt the LSB truncation in each bit-plane of pixels. Because there are different contributions to total image appearance, only higher order bits contain visually significant data and the other bit planes contribute the more subtle details. This idea is presented for power-awareness in Chapter 4.5.2.1 and widely used for image watermarking and data hiding [103] applications. Therefore, it not only features power awareness but also reduces the visual redundancy. Second, the intra-pixel redundancy exploits the spatial locality and is implemented by differential pulse coded modulation (DPCM). Third, a truncated Huffman table is applied to remove the coding redundancy. Overall, the detailed block diagrams of the proposed compressor/de-compressor have been re-drawn in Figure 4.17 and the associated illustrations have been made in the following sub-sections.



(a)



(b)

Figure 4.17: Proposed (a) compressor and (b) de-compressor.

4.5.3.1 DPCM and Scanning Pattern Control

A simple and well-known method for redundancy reduction was to predict the pixel values based on the values previously coded, and code the prediction error. This method is called differential pulse code modulation (DPCM). In general, best predictors are those from the neighboring pixels. It is of utmost importance with respect to how to determine the predictors. In this design, we propose a scanning pattern control technique to create a better predictor based on intra prediction modes. In H.264/AVC, there are 9 and 4 intra 4×4 and 16×16 prediction modes respectively. In Table 4.4, we translate all prediction modes into two classes: horizontal and vertical scanning pattern for simplicity. The predictor of horizontal scanning pattern comes from the left neighboring pixels while the vertical scanning pattern predicts the current pixel based on upper neighboring pixels. The detailed block diagram of prediction method is depicted in Figure 4.18. To guarantee the equivalence between compressor and de-compressor, we need an additional buffer with 3.1Kb to keep the scanning control of one frame. Therefore, the scanning control not only comes from the

Table 4.4 for data compression but also writes into SRAM storage for subsequent frame de-compressions.

Table 4.4: Horizontal and vertical scanning patterns.

Prediction types	Prediction modes	Scanning Ctrl
Intra 4x4	Horizontal, Horizontal-down, Horizontal-up	Horizontal
Intra 16x16	Horizontal	
Intra 4x4	Vertical, DC, diagonal down-left, diagonal down-right, vertical-right, vertical-left	Vertical
Intra 16x16	Vertical, DC, plane	

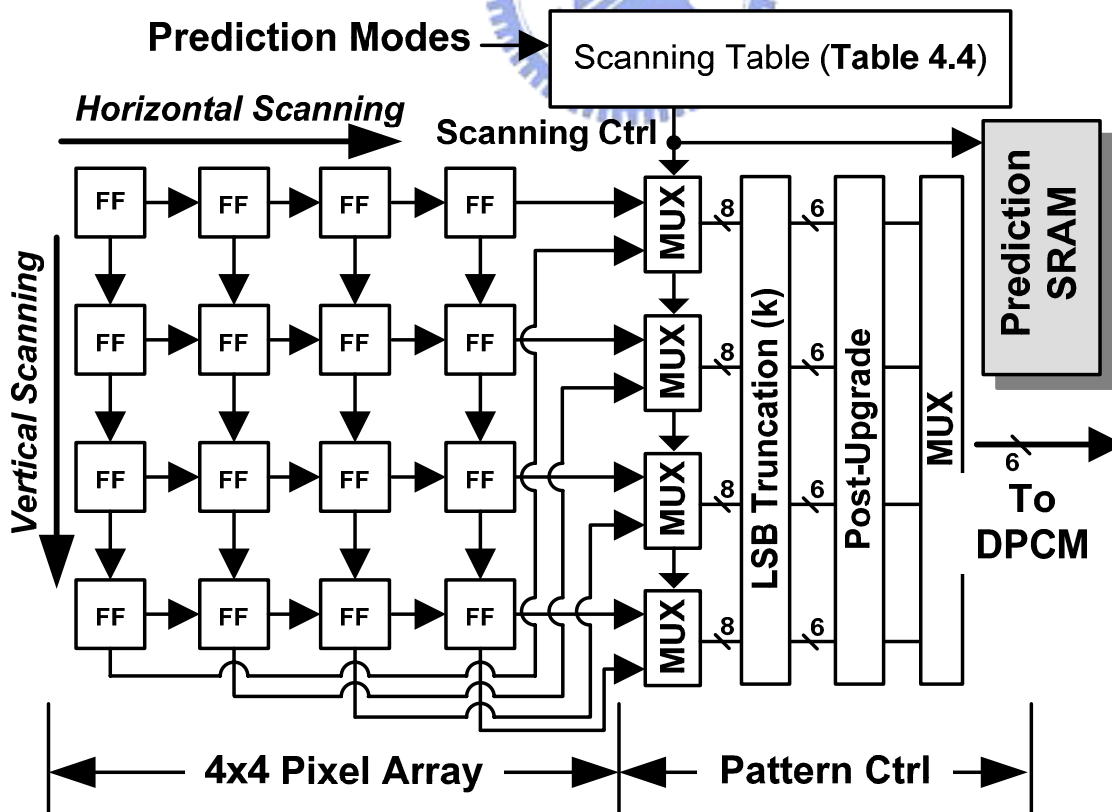


Figure 4.18: The horizontal and vertical scanning control.

4.5.3.2 Truncated Huffman Tables

Although an arithmetic coding is the best-known lossless coding method, it requires expensive iterative coding and decoding procedures and consumes large computing power. On the other hand, because the Huffman code may result in a long codeword and computationally-intensive coding and decoding, we adopt truncated Huffman code to translate the differential pixel into a specified codeword for a low-complexity design approach. The truncated Huffman table is listed in Table 4.5. Specifically, the raw pixel ranges from 0 to 255 while the ranges of differential pixels are -255~255. To suppress the data range and reduce the number of codewords, we apply LSB truncation variable k as 2 without great loss of generality. Hence, the range of differential values becomes -63~63. As for different k , this coded table can also be generated by training numerous video sequences.

Table 4.5: Truncated Huffman tables.

Differential values	Code-word	Code-length
0	0	1
-1,+1	$11x_0$	3
-3,-2,+2,+3	$101x_0x_1$	5
-11,...,-4,+4,...,+11	$1001x_0x_1x_2x_3$	8
-63,...,-12,+12,...,+63	$1000x_0x_1x_2x_3x_4x_5x_6$	11

After receiving the codewords from the truncated Huffman table, we apply a data packing to send the packed results into a data word-line on SDRAM. Figure 4.19(a) depicts the SDRAM organization and each word-length is of size 25-bit where first 1-bit is a TAG signal and the following 24-bit means pixel payloads. We assume LSB truncation k equals 2

and each un-compressed pixel is of size 6-bit. The 1-bit *TAG* indicates whether the following payloads have been compressed or not. Figure 4.19(b) shows three kinds of data formats in one 4×4 sub-block. One is four 4-pixels without compression (i.e. 8-bpp) and another is two 8-pixels with compression (i.e. 3-bpp). The other is two 4-pixels and one 8-pixel with partial compression (i.e. 4.5-bpp). In Figure 4.19(b), TAG_i means the *TAG* signal in the i^{th} entry of SDRAM. We first collect four successive *TAG*s, and thereby pack and send these streams into SDRAM. In the de-compressor side, the four successive *TAG* signals received can be used to unpack the compressed data into the raw pixel data.

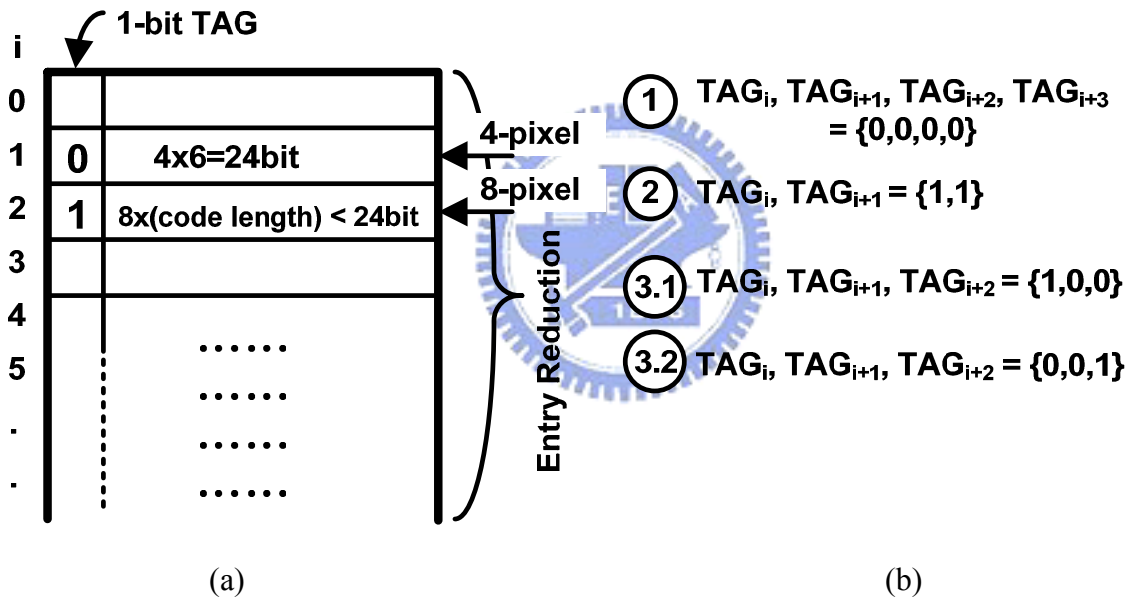


Figure 4.19: (a) SDRAM organization and (b) data packing in 4x4 pixels.

4.5.4 Virtual-to-Physical Address Mapping Technique

Because the motion compensation module requires the reference pixel values in SDRAM based on a given decoding index and motion vector, a highly compressed pixel is difficult for data addressing from SDRAM [100]. To facilitate the SDRAM data addressing, we propose a virtual-to-physical address mapping technique in Figure 4.20. The address

calculation computes the base address under a given virtual address. However, because we store the pixel data into SDRAM in a compressed way, the calculated address will not be equal to the real one. Therefore, we need a translation buffer to look-up the offset address for indicating each physical address on a macro-block level.

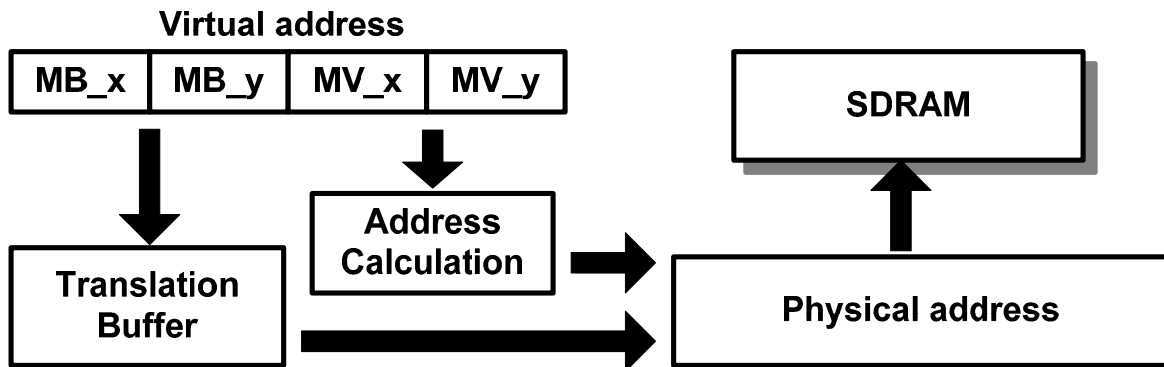


Figure 4.20: A virtual to physical address mapping.

4.5.5 Performance Evaluation

Although the truncated LSB k greatly impacts the visual quality, we can use post-upgrade equation to improve the visual quality. Figure 4.21 shows the performance degradation with and without exploiting post-upgrade method. In the simulated conditions on H.264/AVC main profile, QCIF, mother & daughter sequence, and 15 INTRA PERIOD, the post-upgrade method averagely achieves 4dB of PSNR improvement compared to the preliminary design without exploiting the post-upgrade method. Moreover, a detailed comparison with other leading edge approaches, such as Golomb-Rice [84], ADPCM [104], and DCT-cut [105] based compression, can be made for the further research.

We exploit a metric (bits per pixels, a.k.a. bpp) to analyze the compressed performance. Note that we provide several design alternatives via the variable k . In addition to the lossless performance (i.e. $k=0$), we choose k (i.e. number of truncated bits) as 2 without great loss of

generality and use JM8.2 as our simulated platform. Finally, the detailed results are listed in Table 4.6. The proposed compressor/de-compressor simply exploits the DPCM and table-look-up method. It achieves 3.87-bpp at most when decoding the “Suzie” sequences with QCIF resolution, 100 frames, 15fps, 15 INTRA PERIOD and 2-bit LSB truncation. From the subjective point of view, we capture the 55th frame to give a comparison in Table 4.7.

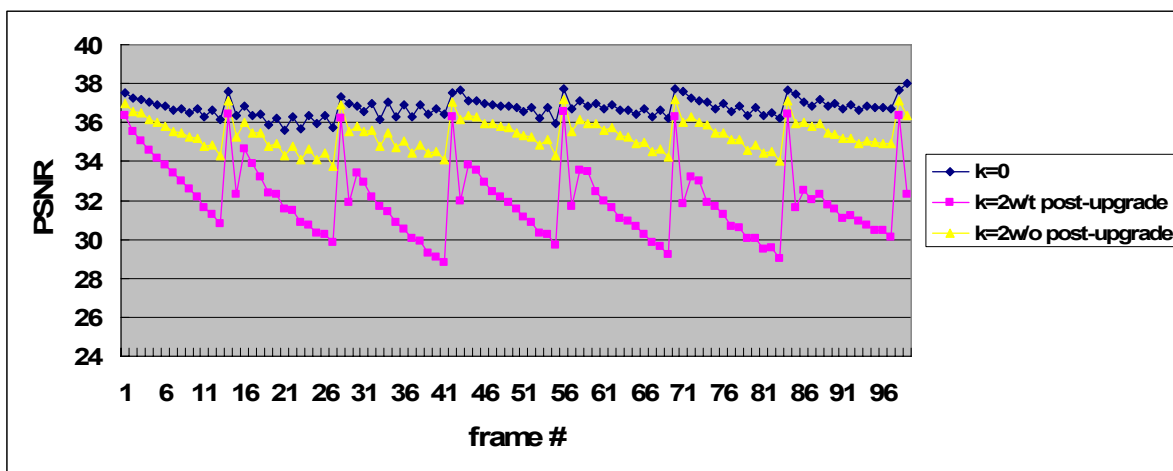


Figure 4.21: Objective visual quality for post-upgrade equation.

Table 4.6: Compression performance for various sequences.

H.264/AVC Main Profile, QCIF, QP=28, 15fps, Intra period = 15					
Sequence		Foreman	Mother & Daughter	Suzie	Akiyo
k = 0	PSNR	35.98	36.76	37.42	37.78
	bpp	6.51	6.25	5.87	5.96
k = 2	PSNR	34.14	35.44	35.52	36.73
	bpp	4.51	4.25	3.87	3.96

Table 4.7: The subjective quality comparison.



Suzie sequence	w/o compression	compressed with k = 0	compressed with k = 2
55 th frame			
bpp	8-bpp	5.87-bpp	3.87-bpp
PSNR	37.42dB	37.42dB	35.52dB

Table 4.8: Hardware Summary.

Item	Specification	
Function	Compressor	De-compressor
Process	0.18 μ m	
Working Frequency	100MHz	
Internal Memory Size	3.1Kb	
Gate Counts	7.4K	6.7K
Processing Cycles/4 \times 4	4-6 cycles	~ 8 cycles
SDRAM	k = 0	16.14mW @ 5.87bpp, 37.42dB
Power	k = 2	10.64mW @ 3.87bpp, 35.52dB

Table 4.8 shows the hardware summary of the proposed compressor/de-compressor. After synthesizing based on UMC 0.18 μ m CMOS technology, the total gate counts are 14.1K excluding the memory. Additionally, the processing cycles are less than 8 on both compressor and de-compressor. Therefore, it achieves both low complexity as well as latency requirements. In addition to the data reduction through compressor/de-compressor,

the proposed LSB truncation scheme can be considered as a power-aware unit that can change the compression behavior to meet the battery lifetime requirements. In general, compressed data will reduce the memory capacity, bandwidth and hence power consumption. In Eq. (4.4), the SDRAM bus bandwidth can be deduced through the $\frac{bits}{pixel}$ (i.e. bpp). Moreover, different bandwidth requirements also impact the memory power consumption. Therefore, the reduction of compressed data can lower the associated DRAM power requirements. We choose CAS latency = 2, BL = 1, $t_{CK} = 7ns$ as our SDRAM model configuration [47]. Specifically, we adopt the system-power calculator [52] as an off-chip power model and use “Suzie” (QCIF) as our test sequence and encode it at 150kbps and 15fps for mobile applications. Table 4.8 exhibits that the power consumption on SDRAM ranges from 10.64mW to 16.14mW. The corresponding compression ratio and visual quality are shown as well. That is, this design can modify its compression behavior based on the current power availability for providing power-awareness features.

$$bandwidth = frame\ width \times frame\ height \times frame\ rate \times \frac{bits}{pixel} \quad (4.4)$$

Compared to the SPIHT-based compression method [87], the proposal achieves low complexity as well as processing cycles with comparable compressed performance. Table 4.9 exhibits the detailed comparison in terms of compression ratio as well as complexity. Although [87] achieves higher compression ratio and lesser PSNR drop, its computational complexity in terms of cost and processing time is considerable. We propose a DPCM-based compression method to reduce the DRAM capacity without extensively adding cost overhead in a system point of view. The proposal simply use DPCM and Huffman table to reduce the processing cycles in the compression and de-compression sides. Moreover, it adapts the DRAM power by changing the number of truncated LSB. Hence,

we present low-complexity VLSI architecture to compress the pixel data for cutting the memory capacity and bandwidth requirements. Although the compression ratio in DPCM-based approach is less than [87], a further research can be studied for making a compromise between compressed ratio and computational complexity.

Table 4.9: Performance comparison.

Item	Proposed	Cheng <i>et al.</i> [87]
Compressed Algorithm	DPCM-based	SPIHT-based
Lossless/Lossy	Supported	Supported
Compression Ratio	3.87~6.51bpp (Variable)	2bpp and 4bpp (Fixed)
PSNR Drop (cf. Foreman)	1.84dB/4.51bpp (H.264 QP=28)	0dB/4bpp and 1.1dB/2bpp (MPEG-4, QP=15)
Process	0.18 μ m	0.18 μ m
Internal Memory Size	3.1Kb	10.24Kb
Gate Counts	14.1K	26.93K
Working Frequency	100MHz	30MHz
Processing	Encode	18 μ s
Time per MB	Decode	
		0.8 μ s
		1.28 μ s

4.6 Summary

Many transmission channels cause severe challenges for streaming or broadcasting video due to bit errors or packet loss. To combating transmission errors, this chapter thoroughly addresses three techniques for improving visual quality and power awareness.

First of all, we point out the importance of error detection. Specifically, inaccurate detections lead to poor performance in error concealment module. We propose a novel CAVLC decoder with soft computing method. In particular, we introduce the soft-decision information to localize the erroneous position at macroblock (MB) levels. This method compares and selects the minimal square difference between the received soft streams and decoded codewords. The corrupted MBs can be early detected and thereby concealed from neighboring pixels. After presenting the soft CAVLC decoder with error detection capabilities, we further introduce a simple error concealment which is tightly combined into the deblocking filter module. The key idea is that both error concealment and deblocking filter will not be activated at the same time. Hence, we develop an error-concealed deblocking filter (ECDF) to improve both hardware utilization and visual quality. On the other hand, we develop a frame re-compression method to cut the bus bandwidth and DRAM capacity. Specifically, we simply use DPCM and table-look-up method to improve the compression ratio. We exploit H.264/AVC's intra prediction modes to find a better predictor. Additionally, this compression technique not only considers the power awareness but also error robustness features for a robust transmission of video data. To summarize, the aforementioned techniques not only improve the error-robustness of this video decoder but also feature simple architectures for further VLSI integration.