國立交通大學

電機學院　電子與光電學程

碩 士 論 文

超越 100Mpixels 之即時 JPEG XR 影像編碼器設計

Over 100Mpixels Real Time JPEG-XR Encoder Design

研 究 生：丁建杉

指導教授：張添烜　教授

中 華 民 國 一 〇 一 年 七 月

超越 100Mpixels 之即時 JPEG XR 影像編碼器設計
Over 100Mpixels Real Time JPEG-XR Encoder Design

研 究 生：丁建杉　　Student：Chien-Shan Ding

指導教授：張添烜　　Advisor：Tian-Sheuan Chang

國 立 交 通 大 學

電機學院　電子與光電學程

碩 士 論 文

A Thesis

Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electronics and Electro-Optical Engineering

July 2012

Hsinchu, Taiwan, Republic of China

中華民國 一〇一 年 七 月

超越 100Mpixels 之即時 JPEG-XR 影像編碼器設計

學生：丁建杉　　指導教授：張添烜 博士

國立交通大學　電機學院　電子與光電學程　碩士班

摘　要

在現在資訊快速發展的時代，許多手持式裝置與照相機都需要使用到高解析度與高精細度的影像處理，於是對於影像儲存的壓縮率也顯得越來越重要。由 Microsoft 所開發的最新影像壓縮技術 JPEG XR 不論在影像壓縮率或是影像的支援度也比傳統的 JPEG 與 JPEG2000 都擁有更好的演算法。

在 JPEG XR 的實踐電路中，有三個最主要的路徑會影響到編碼器的性能。一是適應性的量化器運算(Adaptive Normalization)，二是須不斷更新新的係數掃描順序(Adaptive Scan)，第三個則是判斷最佳的 Huffman Tabel 並得到最佳的壓縮率。因此，如何設計最佳化的管線設計讓編碼運算進行順暢或是使用並行方式讓編碼器增加輸出量則成為此篇論文最大的挑戰。此篇論文中，我們研究了適應性量化器的特色並利用它對係數數目具有有效減少其數量的能力。進而改善了每個管線中的控制器使其能夠有效的減少多餘的掃描、快速推進每一階的處理器，更有效的減少晶片的耗電與增加編碼器的每個單位時間的輸出量。

經過最佳化的硬體設計後，我們量測了晶片的效能。經量測，四張具有相同大小但擁有不同畫面細節的測試圖片在經過編碼器的處理之後，皆得到近乎相同的運算成果。其運算能力的輸出量達到了超過 1 Pixel/Cycle。根據 Synposys Design Compiler 0.18 um CMOS 合成的結果與設定 100MHz 的條件之下，晶片的 Gate Count 為 235,377 而需使用的 SRAMs 大小則為 992 X 3 channels。編碼器的即時運算能力達到了 100Mpixels。

Over 100Mpixels Real Time JPEG-XR Encoder Design

student：Chien-Shan Ding　　Advisors：Dr. Tian-Sheuan Chang

Degree Program of Electrical and Computer Engineering
National Chiao Tung University

ABSTRACT

In modern industry, although high resolution and wide dynamic range images had be used with in several applications like digital camera sensors, web display devices, so the compression of visual information becomes more and more important. JPEG XR [1] is an new image coding standard, based on high definition (HD) Photo developed by Microsoft [3]. It supports high compression performance higher than JPEG and JPEG2000.

Entropy coding was the throughput bottleneck in previous architectures. There are three feedback loops in entropy coding stage; (1) Control of ModelBits, (2) Updating of the scanning order, and (3) Decision of the Huffman table to be used. Therefore, how to design a pipelined module in a straightforward implementation or processing Macroblocks in parallel structure becomes main design challenge. We generalized the characteristic of Normalization (Update ModelBits) and took adventage of reduction of Levels. Our propose pipeline controller can optimal the encoding forward steps to decrease un-necessary data processing. We could safely pipeline all the encoding processes including the entropy coding and achieves higher throughput than those of related works.

After our optimization, estimation of the encoding speed in our implementation is measured. The four images with same size but different manner are tested and represent quite similar results. The calculated throughput in terms of pixel/cycle shows that our implementation can achieve more than 1 pixel/cycle. The architecture is synthesized by Synposys Design Compiler with 0.18 um CMOS standard cell library. The result shows that the gate count of the designed JPEG XR encoder with 100MHz as target frequency is 235,377, number of used SRAMs is required by 992

× 3 channels. An over 100Mpixels real time JPEG XR encoder is designed.

誌　　謝

首先，要感謝我的指導教授—張添烜博士這幾年的指導與提攜。老師在研究上給予我自由的發展空間並適時地對於我的困惑提供完善的建議與方向，讓我學習使用不同的思維去處理所面臨的困境。此外，教授提供多項的資源讓我可以無慮的進行論文研究，待人親切的實驗室夥伴們更是開拓了我的視野並讓我學到更多待人處事的道理。在此由衷的感謝張添烜教授。

感謝我的口試委員們，交通大學郭峻因教授與黃聖傑教授，感謝您們在百忙之中抽空前來指導，因為您們淵博的見識與指導，讓本篇論文更加豐富與完備。

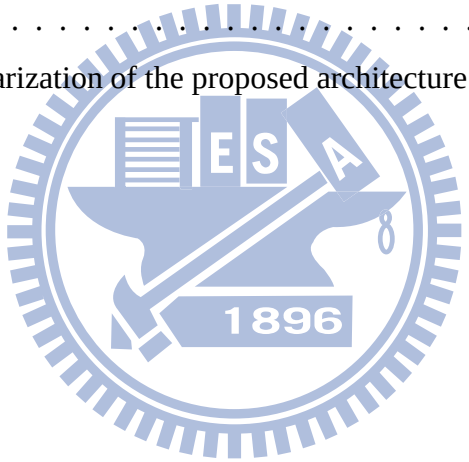
感謝在我研究生涯中提供我許多經驗與幫助的朋友與同事們，因為妳們的協助，使我可以有效的解決研究中遇到的疑惑與困難，即使工作的繁忙讓大家都筋疲力盡，大家卻願意互相幫助分擔彼此的工作，始終相信我的能力，給予我肯定去突破一切的困難。

最後感謝我的父母親，從小到大對我用心的栽培、無怨無悔地照顧關心我，讓我在求學的道路上無後顧之憂；也感謝我的兄弟姊妹這段日子對我的支持與鼓勵，在我研究遇到瓶頸時，總是適時地為我加油打氣，讓我恢復動力繼續向前邁進；最後感謝這些年陪伴在我身邊的 May，在我工作、讀書忙得焦頭爛額、心情沮喪的時候，總是默默在我身邊陪伴，成為我精神上最大支柱。謝謝他們對我的包容與愛，讓我可以毫無後顧之憂完成自己的理想與抱負，勇敢地追求自己的夢想，希望我的努力可以讓你們感到欣慰與驕傲。

Contents

List of Figures

List of Tables



Chapter 1

Introduction

In modern industry, although high resolution and wide dynamic range images had be used with in several applications like digital camera sensors, web display devices, and video games, so the compression of audio-visual information becomes more and more important, especially for applications on mobile devices. Due to advances in very large scale integration (VLSI) technology, many image processing applications has become popular in our daily life. For satisfaction of the high quality image compression, the new compression standard,namely JPEG XR, is discussed and designed with the VLSI architecture.

1.1 Motivation

JPEG XR [1] is an new image coding standard, based on high definition (HD) Photo developed by Microsoft [3]. It supports high compression performance higher than JPEG, and also has an advantage over JPEG 2000 [4] in terms of computational cost. Besides, JPEG XR is expected to be widespread for many devices including embedded systems in the near future. In this paper, we propose a novel architecture for JPEG XR encoding. In previous architectures, entropy coding was the throughput bottleneck because it was implemented as a sequential algorithm to handle data with dependency.

We found that normalization of JPEG XR is help to reduce the number of coefficients, and we could safely and effectively process all pipelined encoding stages including the

entropy coding. For the test in our JPEG XR implementation, we can encode the image data over 1 pixel pre clock which could only be achieved 0.5 pixel/pre clock by previous works.

1.2 Thesis Organization

This thesis is organized as follows. Chapter 2 is the overview of JPEG XR standards and the algorithm analysis. Chapter 3 introduces the pipeline architecture of JPEG XR and it's optimization . Chapter 4 is our experiment results and performance compared to other JPEG XR encoder. Finally, we will give some conclusions in chapter 5, and the future works are listed as well.

Chapter 2

Overview of JPEG XR

The coding flow of JPEG XR is shown in Figure 2.1. First step is to convert one input RGB image to YUV format by using color space converter. Then transform YUV space domain into frequency domain in order to reduce the data size of image. The frequency transform is also play an important role like discrete cosine transform (DCT) of JPEG. In JPEG XR , we call this transform is photo core transform (PCT). The PCT is applied to a rectangular area called a macroblock (MB). When lossy compression, one transform called photo overlap transform (POT) is applied to reduce block noise , which offer occurs at MB boundaries. The PCT and POT are based on lapped biorthogonal transforms [5], [6].

After frequency transformer, the transformed coefficients will be quantized. At lossless mode of JPEG XR, the POT is no required to perform and Quantization Factor will be 1. Following block coefficient prediction process, the quantized coefficients of DC bands, AD bands, AC bands are replaced by prediction engine which detect for horizontal or vertical way to enhance compression rate. Adaptive scan unit scan the predicted coefficients of different bands with distinct adaptive scan order. The scanned data are rearranged to two bitstreams, one is run-level form and another is flexbits format. Finally, entropy coding unit encode the run-level components to JPEG XR bitstreams by using adaptive Huffman tables. Hereafter, the details of each process are described.

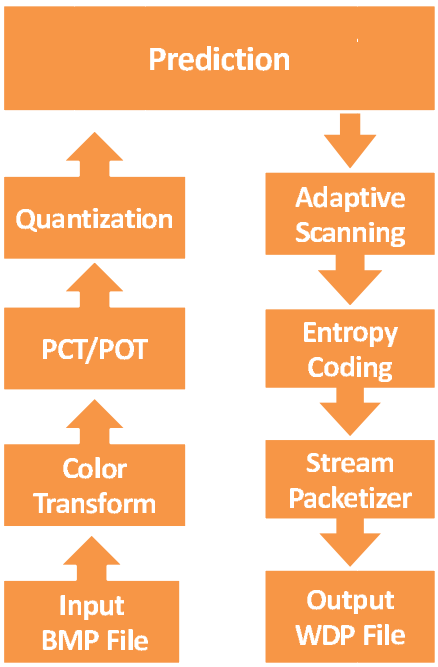


Figure 2.1: Encoding Flow Of JPEG XR

2.1 Color Space Convert

The RGB color space is the most prevalent choice for computer graphics because color display use red, green, and blue to create the desired color. Therefore, the choice of the RGB color space simplifies the architecture and design of the system. Also, a system that is designed using the RGB color space can take advantage of a large number of existing software routines, since this color space has been around for a number of years. However, RGB is not very efficient when dealing with ”real-world” images. Because the human eye has different characteristics of sensitivity for chroma and luminance. For these reasons, many video standards use luma and two color difference signals. The purpose of RGB to YUV conversion is to reflect the color sensitivity. When color space is converted from RGB to YUV in this JPEG XR encoder. Then, the most important and detailed information of image like texture are concentrated on Y channel. The color conversion is reversible, in other words, it is lossless conversion.

$$V = B - R$$
$$U = R - G + \left\lceil \frac{V}{2} \right\rceil$$
$$Y = G + \left\lfloor \frac{U}{2} \right\rfloor$$

Figure 2.2: Color Space Convert Of JPEG XR

2.2 PCT : Photo Core Transform

All the operator of JPEG XR compute with a MB, which consists of 16×16 coefficients. In PCT/POT process stage, the MB is partitioned into 16 4×4 blocks. The PCT/POT is applied to these blocks and convert input image from space domain into frequency domain. The JPEG XR includes two PCT stages and two optional POT stages. Our proposed architecture had no support POT stages because it is non-essential and this paper focus on lossless compression speed.

The PCT is inspired by the 4×4 DCT, yet it is fundamentally different. The first key difference is that the DCT is linear whereas the PCT is nonlinear. The second key difference is that due to the fact that it is defined on real numbers, the DCT is not a lossless operation in the integer to integer space. The PCT is defined on integers and is lossless in this space. The third key difference is that the 2D DCT is a separable operation. The PCT is non-separable by design.

The PCT is composed of three kinds of basic operations namely T_H, T_{ODD}, T_{ODDODD} transform. These transforms are designed with 4 input and 4 output Hardmart transforms [6].

2.2.1 T_H Transform

The Hadamard transform is a 2-point operator that can mathematically be described by equation (2.1). The 2x2 Hadamard transform is developed by taking the Kronecker product of the 2-point Hadamard with itself as seen in equation (2.2).

$$T_H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.1}$$

$$T_H = Kron(T_H, T_H) = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{2.2}$$

The T_H Transform is possible to implement this 2-D Hadamard transform by using only trivial lifting steps [6] as shown indescribed in Figure 2.3 [6].

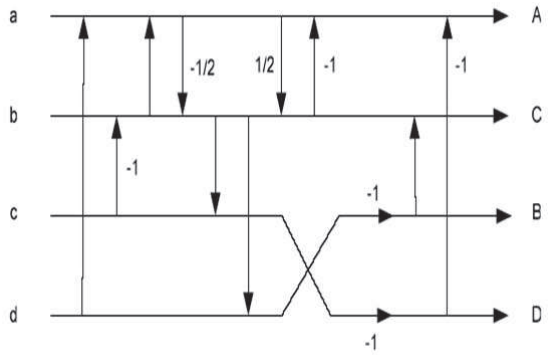


Figure 2.3: T_H transform flow Of JPEG XR.

2.2.2 T_{odd} Transform

The odd transform was developed by taking the Kronecker product of a 2-point rotation operator described in equation (2.3) and the 2-point Hadamard operator .

$$T_R = \begin{bmatrix} -cos\frac{\pi}{8} & sin\frac{\pi}{8} \\ sin\frac{\pi}{8} & cos\frac{\pi}{8} \end{bmatrix} \tag{2.3}$$

The transform can be reduced to a set of equations including four that are non-trivial as described in Figure 2.4 [6] .

2.2.3 T_{oddodd} Transform

The odd-odd transform was developed by taking the Kronecker product of a 2-point rotation operator with itself. The transform can be reduced to a set of equations including three that are non-trivial as described in Figure 2.5 [6].

In the PCT stage1, all 16 blocks is applied to 3 kind of filter operation: T_H, T_{ODD}, T_{ODDODD} transform. First step, the filter operation is applied to 4×4 areas evenly straddling blocks in two dimensions, and each block is decomposed into one direct current (DC) coefficient and 15 alternating current (AC) coefficients. After completion of the first transform stage, the 16 DC coefficients are concentrated to one new 16×16 block. Applie this new block into transforms as same as pct stage1 again. As a result of these two stages of PCT, each MB is decomposed into 240 high-pass (HP) coefficients (AC coefficients of PCT stage 1), 15 low-pass (LP) coefficients (AC coefficients of PCT stage 2), and one DC coefficient (DC coefficient of PCT stage 2). These operations are repeated for all color planes. This process about the relation among HP, LP, and DC coefficients is shown in Figure 2.6 .

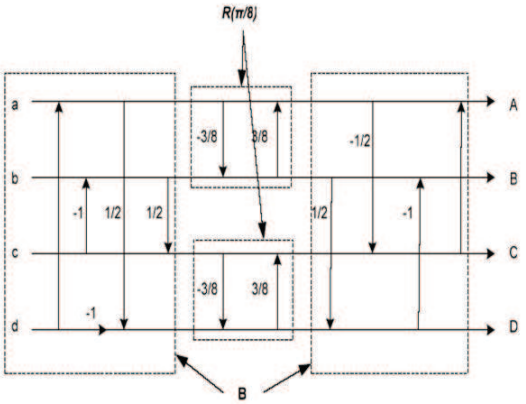


Figure 2.4: T_{odd} transform flow Of JPEG XR.

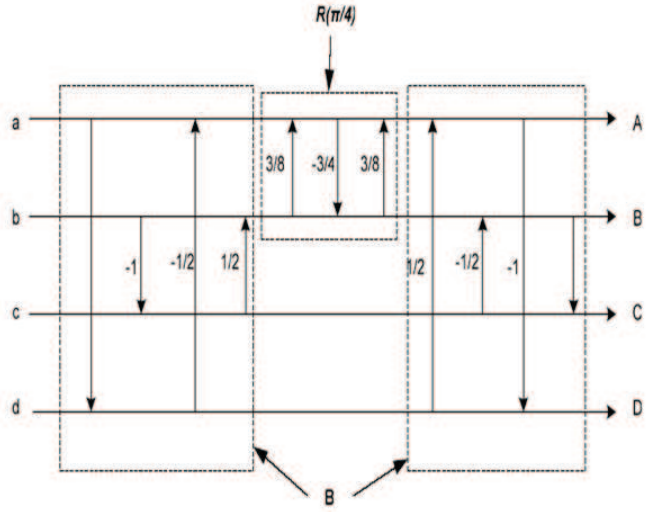


Figure 2.5: T_{oddodd} transform flow Of JPEG XR.

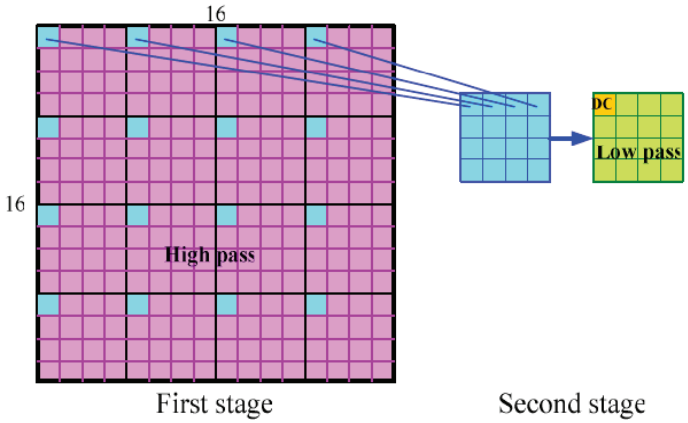


Figure 2.6: Photo Core Transform Flow Of JPEG XR.

2.3 Quantization

Quantization is a process whereby the transform coefficients are essentially divided to reducing the number of bits and rounded to an integer value, called the quantized value. De-quantization is the process where essentially coefficients are reconstructed from their quantized values by multiplying them. The divisor in quantization and multiplier in de-quantization are usually identical and referred to as the quantization parameter or QP. In the lossless coding mode of JPEG XR, QP = 1. For lossy coding, QP > 1.

QP is chosen to be an integer chosen from a harmonic scale. This is determined by another integer QPIndex as per the rule: When QP is smaller than 16, QUANT is the same as QP. In the other cases, QUANT is given by the following equation,

$$QP = QP_{Index}\ for\ QP_{Index}{<}16$$
$$QP = ((QP_{Index}\%16) + 16) \ll ((QP_{Index} \gg 4){-}1)\ otherwise$$

The symbol "%" denotes remainder of integer division or the mod function.

$$When\ QP_{Index} = 255,$$
$$QP\ is\ given\ by\ ((255\%16) + 16) \ll (15{-}1) = 507904.$$

On the decoder side, each entropy decoded transform coefficient is de-quantized by multiplying the coefficient with QP. On the encoder side, the specific rounding factor in the process of quantization is implementation specific, and is not covered in this document. In the lossless mode, coefficients are passed through (i.e. multiplied or divided by 1) on both the encoder and decoder.

Each MB and each frequency component can be quantized with each quantization parameter. In other words, QP is allowed to differ across AC, lowpass and DC bands. The DC QP within a tile is fixed. The DC QP across tiles may vary. The AC and lowpass QPs within a tile may take on either the same value, or one of a multiple value set. This may changed at every MB and is signaled in the bitstream.

2.4 Prediction

When coding an intra block, the DC coefficients, lowpass and highpass bands coefficients are coded by intra prediction. Intra prediction is an operation used in JPEG XR standards to reduce the spatial redundancy between 16×16 blocks. It also be efficient to enhance the compress rate. When tiling is used, each tile is deemed to be a separate image for the purpose of DCAC prediction, to ensure independent decoding of tiles. In order to determine valid predict orientation for current marcoblock ,JPXR XR should record one DC and three AD coefficients from individual neighboring top and left macroblock first. Through the calculation with the analysis of algorithms, the optimal predition data obtained from TOP or LEFT macroblock compute target macroblock smallest coefficients.

There are three levels of prediction: DC prediction, LP prediction, and HP prediction used in JPEG XR standard . DC prediction is illustrated in Figure 2.7. The quantized coefficients are predicted with four alternative modes: Predict from Left ,Predict from Top, Predict from TOP and LEFT, or Null Predict. Which mode to pick is determined from the position of the macroblock, as well as the DC values to the left, top and top-left of the macroblock. Further, if the image has color channels, the corresponding values of the chroma channels are also used.

2.4.1 DC Prediction

The DC prediction mode (dc_mode) pseudo_code is showing in Figure 2.8, [MBx,MBy] is the macroblock index of the current macroblock in the X and Y directions in an image, starting from [0,0].

For example, the DC coefficients of block X is predicted from the DC coefficients of

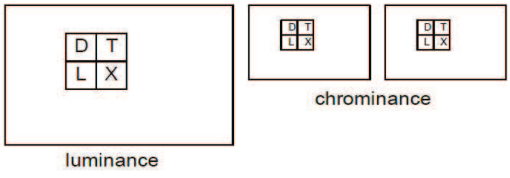


Figure 2.7: Prediction of DC coefficients of blocks.

blocks TOP, LEFT or TOP_LEFT. In computing the prediction of block X. If the absolute value of a horizontal gradient is less than the absolute value of a vertical gradient, then the quantized DC (QDC) of block TOP is used as the prediction, else if the absolute value of a horizontal gradient is larger, then the QDC of block Left is used. If the above two are all no established, then predict current QDC with average of TOP and LEFT.

2.4.2 LP Band Prediction

The LP prediction depends on DC prediction, as shown in Figure 2.9. The LP coefficients in the first row or in the first column are predicted with three previous LP coefficients from TOP, LEFT, or TOP_LEFT macroblock. The direction of LP prediction is the same as DC prediction.

```
If (mx == 0 and my == 0)
   dc_mode = Null predict
Else if (mx == 0)
   dc_mode = Predict from top
Else if (my == 0)
   dc_mode = Predict from left
Else {
   diff_h = abs (D - L)    // luminance
   diff_v = abs (D - T)    // luminance
   If (chrominance channels are available) {
      diff_h = diff_h * scale + sum_over_chrominance_channels { abs (D - L) }
      diff_v = diff_v * scale + sum_over_chrominance_channels { abs (D - T) }
   }

   If (diff_h * orient_weight < diff_v) {
      dc_mode = Predict from top
   }
   Else if (diff_v * orient_weight < diff_h) {
      dc_mode = Predict from left
   }
   Else {
      dc_mode = Predict from left and top
   }
}

Where
scale = 8 for YUV 420, 4 for YUV 422, 2 otherwise
orient_weight = 4
```

Figure 2.8: Prediction pseudo code for DC coefficients.

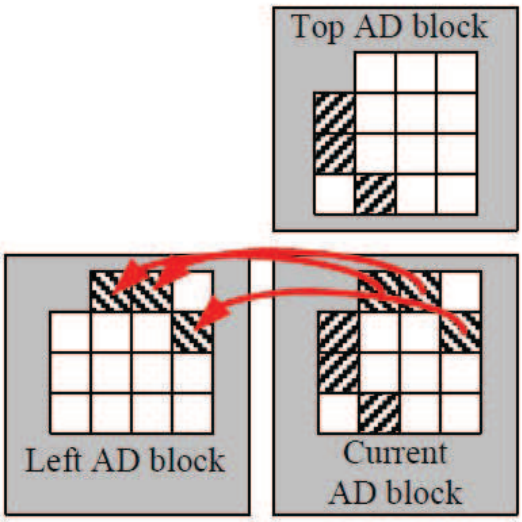


Figure 2.9: Prediction of AD coefficients of blocks.

2.4.3 HP Band Prediction

The highpass prediction mode is determined by the pseudo_code shown as below Figure 2.10. There are 6 coefficients in the low pass_band is used to determine the orientation of AC prediction, based on a simple metric associated with each macroblock.

Three modes are allowed for all blocks within a macroblock for which in-macroblock prediction is possible. For blocks that have no valid reference within the macroblock, null prediction is used. The three modes are:

1. Predict from left (predictor = DCAC [left_block within macroblock])
2. Predict from top (predictor = DCAC [top_block within macroblock])
3. Null predict (predictor = 0)

Prediction from left is shown in Figure 2.11. Prediction from top(Figure 2.12) is similar, with the pattern of arrows transposed to point downwards. The first column of blocks is not predicted in highpass band prediction.

In the implementation of a JPEG XR codec (encoder or decoder), the total information that needs to be available for future use is 1 DC + 6 LP = 7 coefficients per macroblock channel(chanel Y,U,V). Therefore, at most 21 coefficients need to be cached per macroblock for YUV444 encode mode. Further, the coefficients used for ”prediction from

```
diff_h = abs(lowpass(4)) + abs(lowpass(8)) + abs(lowpass(12))
diff_v = abs(lowpass(1)) + abs(lowpass(2)) + abs(lowpass(3))

If (chrominance is present) {
        diff_h = diff_h     + abs(lowpass_U(4)) + abs(lowpass_V(4))
        diff_v = diff_v     + abs(lowpass_U(1)) + abs(lowpass_V(1))
}

If (diff_h * orient_weight < diff_v) {
    highpass_DCAC_mode = Predict from top
}
Else if (diff_v * orient_weight < diff_h) {
    highpass_DCAC_mode = Predict from left
}
Else {
    highpass_DCAC_mode = Null predict
}
```

Figure 2.10: Prediction pseudo code for AC coefficients. [1]

left" can be replaced after the next macroblock is coded. For YUV 444, therefore, it is necessary to only cache 12 coefficients per macroblock for use in the next row of macroblocks.

2.5 Adaptive Scan

Coefficient scanning is the process of reordering the array of transform coefficients into a linear array. This is also commonly referred to as zig-zag scan in JPEG standard. Within a block of data, coefficients are scanned in a deterministic pattern as shown in the example in Figure 2.13. This is similar to the traditional zig-zag scan applied to a 4×4 block. In this example, the first coefficient in the linear array is the top left entry (DC) of the transform coefficient matrix. The second coefficient in the linear array is the transform coefficient marked 1 and so on. Thus, the representation of Figure 2.13 uniquely determines a scan pattern. Scan patterns in JPEG XR are not required to be continuous as in the above example. Further, scan patterns in JPEG XR are allowed to change scan order

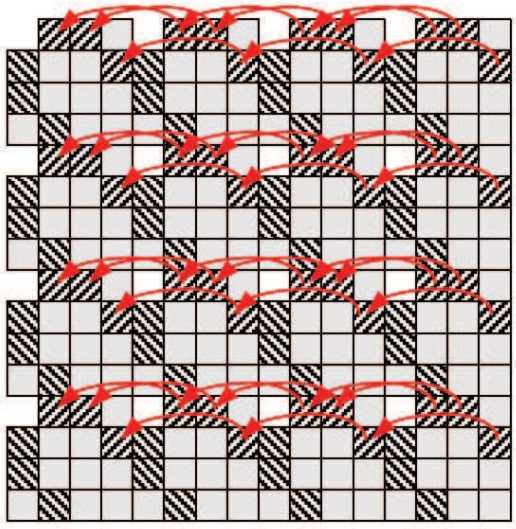


Figure 2.11: Prediction from left for AC coefficients in an intra MB.

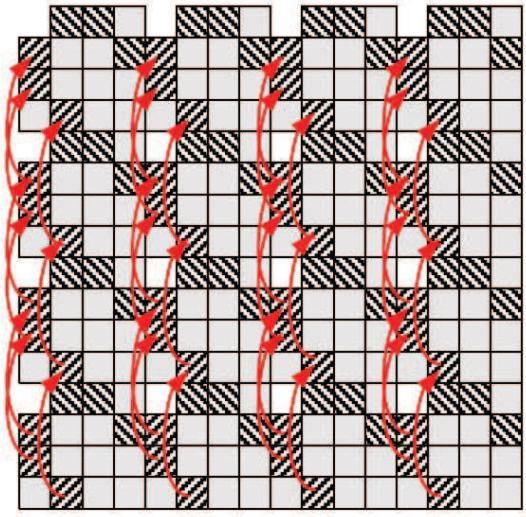


Figure 2.12: Prediction from top for AC coefficients in an intra MB.

with adaptive rules which are defined below.

Three scan patterns are used in JPEG XR. These are referred to as the lowpass, highpass horizontal and highpass vertical scan patterns. The lowpass scan pattern is used to encode/decode the lowpass transform coefficients in a macroblock. The highpass horizontal and vertical scan patterns are used to encode/decode the highpass transform coefficients in a macroblock. Macroblocks that are signaled as dominant horizontal use the highpass horizontal scan pattern, and likewise for vertical. Macroblocks showing no dominance of orientation also use the highpass horizontal scan pattern. The scan pattern is derived from the macroblock DC prediction pattern as defined in 2.8. The three scan patterns are initialized to a specific ordering at the start (top left macroblock) of each tile. The lowpass scan pattern and the highpass horizontal scan pattern are initialized to the pattern shown in Figure 2.14(a)[1], and the highpass vertical scan pattern is initialized to the pattern shown in Figure 2.14(b)[1]. All color channels within a macroblock use the same corresponding scan pattern.

In the adaptive scanning, only 15 coefficients in the block(4×4 matrix) are define as a 1D array. The array elements refer to the transform coefficient index (from 0 through 15, in raster order for the 4×4 transform block), in order of their scan. Thus, the scan pattern shown in Figure 2.14(a) can be written as:

Order[ ] = 1,4,5, 2,8,6,9, 3,12,10,7, 13,11,14,15

For each of the three scan patterns, another 1D array which is called ”scan weight” is created. This is initialized with descending values as shown:

Weight[ ] = 28,26,24, 22,20,18,16, 14,12,10, 6,4,2,0

The array ”Weight” tallies the incidence of occurrence of the particular coefficient. The scan weight is updated while current processing coefficients is not zero. When a coefficient with non-zero value is scanned, the weight corresponding to this coefficient order is incremented by one. Then the scan order is sorted again according to the values of weight. If Weight Array is found that Weight[n] $>$ Weight[n - 1], an exchange operation is applied to the scan order. During the exchange step, the scan orders and corresponding Weight of n and n - 1 are switched. Then the new updated order is obtained

Figure 2.15(a) shows a situation where Totals[n] $>$ Totals[n - 1], indicated by shaded

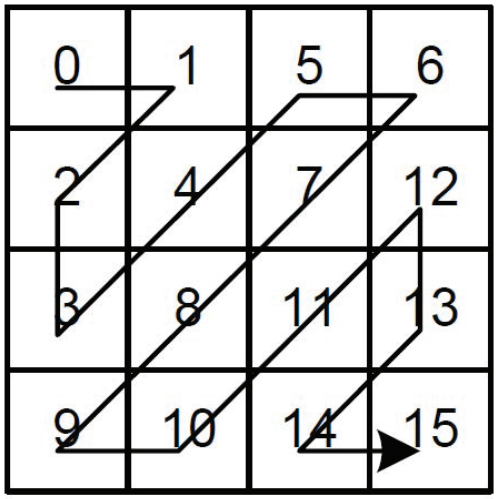


Figure 2.13: Traditional Zig-Zag scan pattern.

X	0	3	7
1	2	5	10
4	6	9	12
8	11	13	14

(a)

X	3	7	9
0	2	6	11
1	5	12	14
4	8	13	10

(b)

Figure 2.14: Initial DC scan patterns (a) lowpass and highpass horizontal scan pattern, and (b) highpass vertical scan pattern.[1]

elements. The arrows show the elements that need to be exchanged. The table of Figure 2.15(a) also shows the Order and Totals arrays subsequent to the exchange, and Figure 2.15(b) shows the corresponding scan order indices on the 4×4 block. Exchange, when triggered, occurs subsequent to encoding or decoding. Therefore, the adaptation is causal.

2.6 Update Modelbits

Wide dynamic range input data resulting in wider dynamic range transform coefficients during the process of encoding an image. It also large the range of quantized transform coefficients when encode image with small or unity quantization factors. Adaptive coefficient normalization is a step in the JPEG XR encoder which processes the transform coefficients to render it suitable for efficient entropy coding.

JPEG XR tracking a statistical measure of variance of transform coefficients in adaptive coefficient normalization process. Based on this measure, the current transform coefficients are regrouped into two data arrays. One is normalized coefficients,the second is

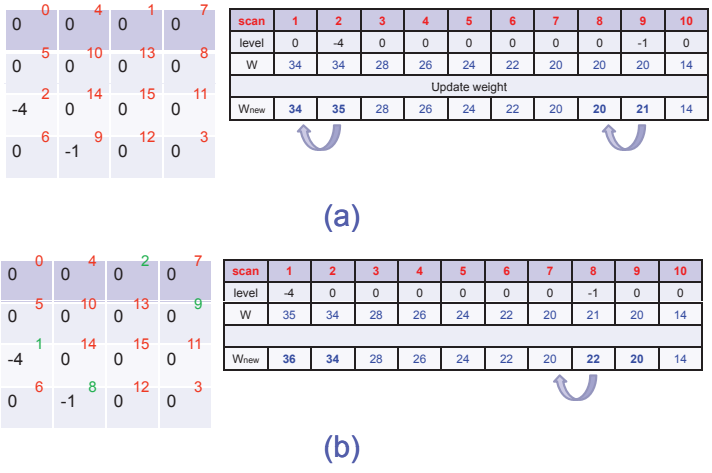


Figure 2.15: Inverse scan order (a)just prior to an exchange operation occurring between shaded coefficients, (b) subsequent to exchange operation.

dyadic bins which called Flexbits. Instead of encoding the transform coefficient, its bin id is sent using an efficient entropy code. Subsequently, an in-bin address locating the transform coefficient within its respective bin is sent. This index is sent with a fixed length code called Modelbits. Sign information is also sent if necessary. Adaptive coefficient normalization is used for all frequency bands including DC, lowpass and highpass.

The data flow of entropy coding in JPEG XR is shown in Figure 2.16. In the entropy coding, first, scanned coefficients are shifted according to ModelBits. Bits of each coefficient are separated into FlexBits and normal data. Shifted data which are non-zero are concentrated to compute next one ModelBits with different weight when coding in different bands.

The FlexBits are considered as the shift-out data in normalization, while the normal data means the normalized coefficients after shift . The normalization process is exemplified in Figure 2.16. If ModelBits is equal to 4. The least significant four bits are output as FlexBits, and the other bits are output as normal data. While FlexBits are output without any entropy coding, normal data is sent to the run-level coder, which encodes the normal data into stream with period interleaved runs and levels.

2.7 Run-Level Coding

JPEG XR Run-Level Encode(RLE) is one compress method which similar to the Run-Length-Encode algorithm in JPEG. The purpose is in order to reduce repeatability of zero coefficients between any two non-zero coefficients. The arranged coefficients are given two different symbols: RUN and LEVEL. RUN means the numbers of successive zeros before non-zero coefficient, and LEVEL is orignal non-zero coefficients. After processing of RLE algorithm, the RLE results are coded with independent Huffman tables. Figure 2.16 describes the process of RLE.

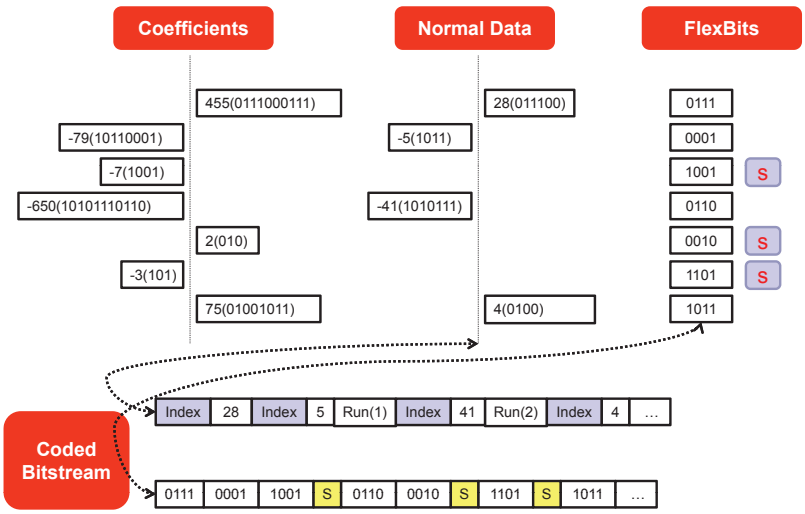


Figure 2.16: Normalization and Flexbits (If Modelbits = 4).

2.8 Coded Block Pattern

The JPEG XR supports to using a coded block pattern(CBP) to signal ALL-ZERO blocks. CBP for one macroblock consists of 18 bits pre channel(each of YUV444). In addition 16 bits to marked HP coefficients of each 16 blocks, extern other 2 bits labeled DC coefficient and LP block. By using this CBP, which be decoded from bitstream, the run-level decoding , Huffman decoding can be skipped when CBP flag shows that is all-zero block.

Table 2.1: Structure of CBP

Y Channel CBP		
DC	LP	HP
1bit	1bit	16bits

2.9 Huffman Coding

Finally, these runs, levels, and CBPs are coded using adaptive Huffman coding. The coefficient levels and signs are encoded according to their frequency band and use different Huffman Tables to generate the corresponding resulting bitstream. All used adaptable predefined Huffman tables are listed as Figure 2.17 [7]. Huffman coding is the most common coding using in image compress. It also be one lossless compress method, in other words, it is reversible. The Huffman coding of JPEG XR is some different to other standards. The Huffman table using in a general coding scheme is one fixed table. But in JPEG XR, several tables but smaller are prepared, and current table on use is adaptively selected based on benchmark calculated by Adaptive Huffman Coding engine. The use of small predefined Huffman Tables helps maintain a small memory footprint and the estimation algorithm is computationally light. Many but small separated Huffman tables is more efficient for Huffman Coder to search Symbol corresponding to the coding coefficient.

The effectiveness is given for a given symbol sequence of each table. Owing to this scheme, an effective coding with shorter code length is achieved. In this adaptive Huffman coding, the weight called Delta of current coding Symbol are calculated and the differences are accumulated. If the accumulated value runs up to the threshold, the current table (Table X) is changed to the most effective one, which is Table (X-1) or Table (X+1). This change occurs only after processing target macroblock. As a result of these processing, scanned coefficients entropy coded.

2.10 Bitstream Structure

Figure 2.18 [1] shows the JPEG XR image structure hierarchy. JPEG XR support max 256 columns of tiles in the horizontal direction and max 256 rows of tiles in the vertical direction. Thus, an image may contain max 65536 tiles. When an image only contains one tile, it is said to be untiled. If the number of tiles is greater than 1, the image is said to be tiled. Tiles form a regular pattern on the image in other words, tiles in a horizontal row are aligned with the same height; tiles in a vertical column are of the same width and

Figure 2.17: VLC Coding Tables used in JPEG XR.

aligned. Subject to the above, tiles may be of arbitrary size which is a multiple of 16 and macroblock aligned.

There are two fundamental modes of operation of JPEG XR affecting the structure of the bitstream Spatial and Frequency. In both the modes, the bitstream is laid out as a header, followed by a sequence of tiles as shown in Figure 2.19 [1] .

In the Spatial Mode, the bitstream of each tile is laid out in macroblock order. The compressed bits pertinent to each macroblock are located together. Macroblock data is laid out in raster scan order: scanning left to right, top to bottom.

In the Frequency Mode, the bitstream of each tile is laid out as a hierarchy of bands. The first band is referred to as DC. The DC band carries information of the DC value of each macroblock, in raster scan order. The second band is referred to as Lowpass. This band carries information of the lowpass coefficients which are fifteen in number in each macroblock (for each color plane, with some exceptions). The third band is called the AC band and carries information of the remaining 240 coefficients of each macroblock color_plane. Finally, the fourth band called Flexbits is an optional layer that carries information regarding the low order bits of the AC coefficients particularly for lossless and

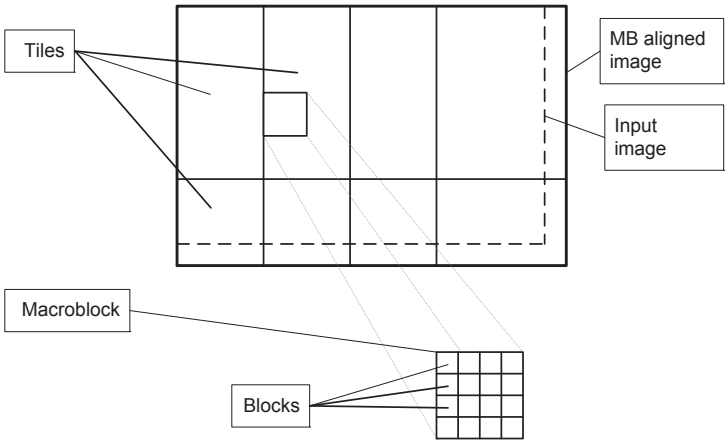


Figure 2.18: Image structure hierarchy.

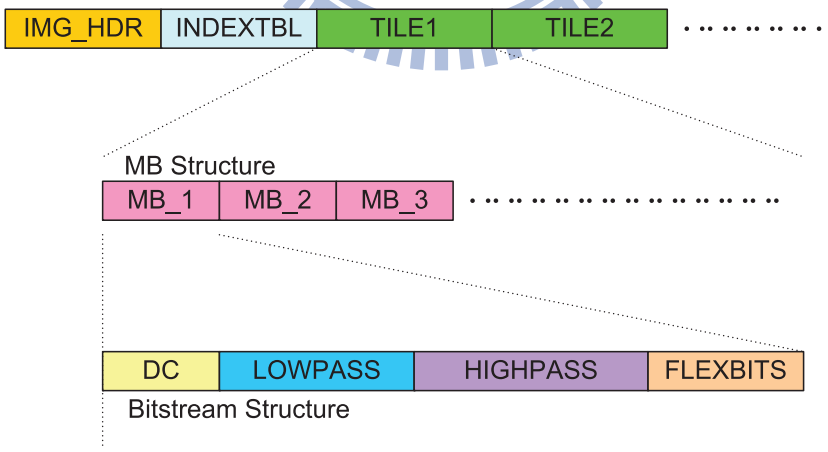


Figure 2.19: Layout of JPEG XR bitstream: Image header is followed by a sequence of tiles which are in Spatial or Frequency mode.

low loss cases.

Chapter 3

Architecture design of JPEG XR

3.1 Profiling

The full coding of JPEG XR process with three main parts: Frequency Transform, Prediction and Entropy Coding. Before our design of JPEG XR encoder architecture, performance for each encoding stage in C sample code builded by Microsoft is analyzed. Analysis for CPU processing time is described in Figure 3.1(a), Entropy Coding account for most of CPU usage and reach to 85%. Amount them, the entropy coding of coefficients in high pass band cost the most resources and up to 77%. Because there are 240 coefficients in high pass band encoded in each macroblock, but only 15 coefficients in low pass band and 1 dc coefficient are processed.

Entropy Coding contains many feedback mechanisms to detect the status of the current coefficient to optimize the best compression ratio of Entropy Coding. For example, adjust the normalization shift-bit numbers to enhance the performance of RUN-LEVEL coder. Adaptive scanning change the coefficient scan-order to concentrate all non-zero coefficients to improve the hit rate of the coefficient scanning. Adaptive Huffman Table switch different tables to minimum the using bits as much as possible. Although the optimized adaptive computing is helpful to increase the image compression ratio, but the relative computational complexity is also increase the cost of encoder architecture. Due to these problems, how to design these architectures become challenge to achieve high-speed performance for current embedded systems such as digital still cameras. How to

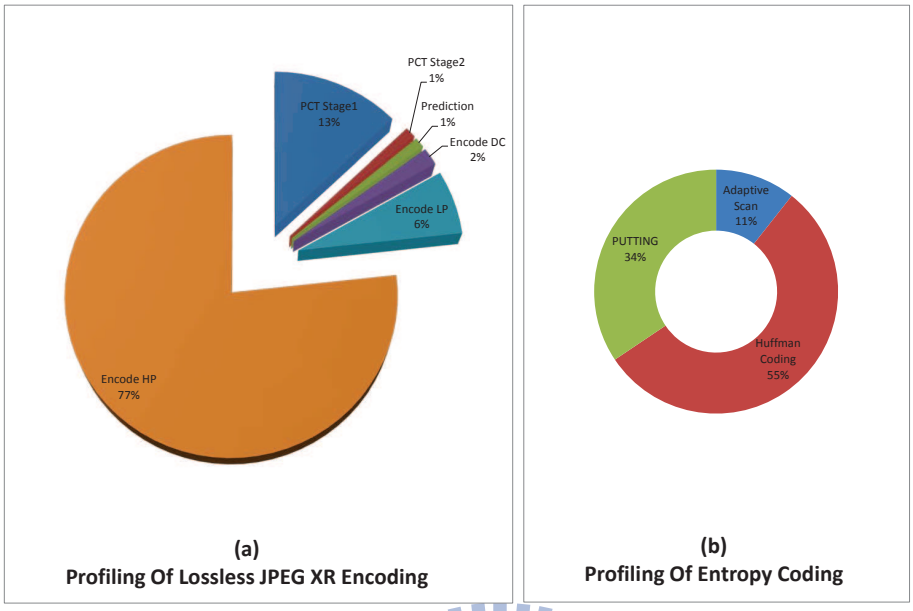


Figure 3.1:

speed up the entropy coding in high pass band is the primary task in architecture design of JPEG XR Encoder. The Entropy Coding of JPEG XR process with three parts: Adaptive Scan, Huffman Coding, Bitstream Putting. As the profiling of Entropy Coding stage, Figure 3.1(b), which also shows that huffman coding including of Index, Level, and Run costs 55% of total entropy coding usage. Our proposed architecture of Entropy Stage is also divided into three stages with pipeline buffers. The pipelined architecture can help to decrease the timing of critical patch and increase the throughput. The buffers record the arranged result from adaptive scanning smoothly, for this reason, the processed coefficients by RLE can be send to each huffman look-up table independently. Namely, our proposed huffman coding can translate Index, Level and Run to relative codewords without data dependency and improve the encode throughput.

The coding complexity of the prediction stage is fewest than PCT stage and Entropy stage, because that only 52 coefficients in one macroblock are necessary to be predicted. Pipelined structure can balance the different complexities of these three stages effectively.

But in Entropy Coding stage, the data dependency of coding coefficients also become the main bottleneck of full encoder. On the other hand, by considering the data dependency of all encoding processes including entropy coding, the fully pipelined structure is proposed, so the bottleneck in entropy coding is overcome and it achieves higher throughput than those of related works.

3.2 Design Challenges

The design challenges about implementation of JPEG XR architecture are described as below list.

3.2.1 Mismatch Data Processing Sequence

The encoding sequences in blocks are different when process PCT, Prediction and Entropy Coding. That means the processed result of PCT can not be transmitted to prediction stage currently. Similarly, the entropy coding also can not compute the predicted result directly. The buffers for recording operated coefficients temporarily is required, which will increase the cost of hardware.

3.2.2 Code Block Pattern

Coded block pattern(CBP) is one unit which is used to signal ALL-ZERO blocks. The normalized coefficients are detected and counted, then the blocks which contain all coefficients to be zeros are labeled. In the structure of JPEG XR bitstream, the CBP is compressed and arranged in the front of bitstream of each band. The entropy result should be held until the processing of CBP is ready.

3.2.3 Architecture Of Entropy Coding

According to the analysis of JPEG XR profiling, Entropy Coding is the most complex stage of all. Three feedback mechanisms which contain Adaptive Scan, ModelBits and Adaptive Huffman used to optimize the best compression ratio are all executed in this

stage. The status of the current coefficients are detected, then responded weights are concentrated to calculate out great solution for replacement of original one. The data dependency of coding coefficients in Entropy Coding stage become the main bottleneck of full encoder. Proposed architecture should overcome the complexity and accelerate the throughput of computing.

3.3 System Overview

The pipeline stage of this designed JPEG XR encoder can be divided into three Processing Element (PE) stages, as shown in Figure 3.2. The color conversion and PCT modules are computing with the 4×4 block matrix structure without feedback information, they are sensible to arranged into the same stage at the beginning. The input RGB image convert to YUV data and transfer into frequency coefficients in this stage. After processing of this stage, the coefficients of DC,AD,AC bands are saved into pipeline buffer.

The prediction unit is used as second stage. There are different direction compar-

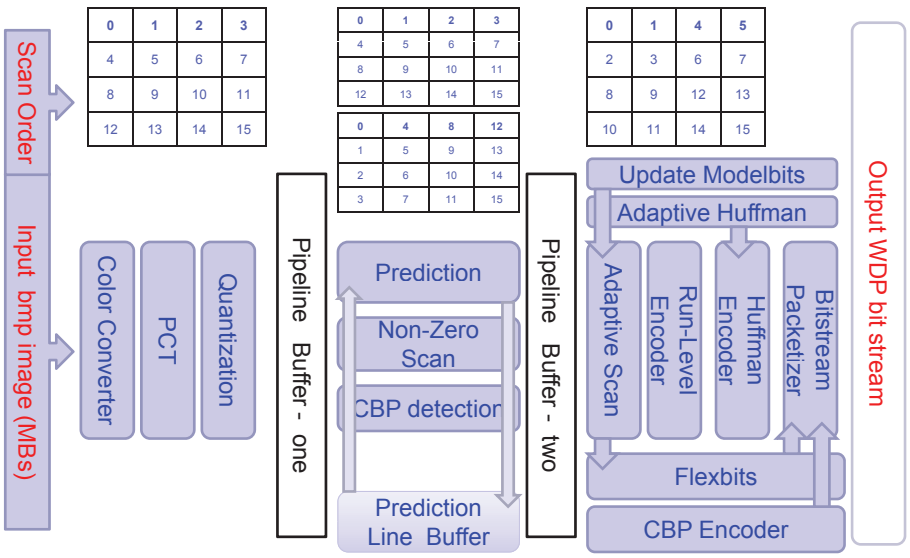


Figure 3.2: Three pipeline stages of JPEG XR architecture.

isons for separate DC,AD,AC bands to process prediction function. This stage also need prepare one buffer of line for coefficients prediction use when prediction mode is “Predict From Top”. Then the entropy encoding module which have highly data dependency is divided as the third stage. The data dependency cause this stage to become the most complex architecture of full JPEG XR design. The three main complex data dependency controller resulting coding speed in lower performance. This section will be discussed in later chapters.

3.4 Color space converter

Color domain converter is one simple process unit which is builded with only two adder and two subtractor. Figure 3.3 describes the structure how RGB data are converted into YUV data from input image.

3.5 Architecture of PCT

Figure 3.4 shows the implementation of the Hadamard transform, whick contains only trivial operations. In total there are 8 instances of addition / subtraction.

Figure 3.5 shows the implementation of the T_{ODD} transform, which contains only trivial operations. In total there are 24 instances of addition / subtraction.

Figure 3.6 shows the implementation of the T_{ODDODD} transform, whick contains only trivial operations. In total there are 19 instances of addition / subtraction.

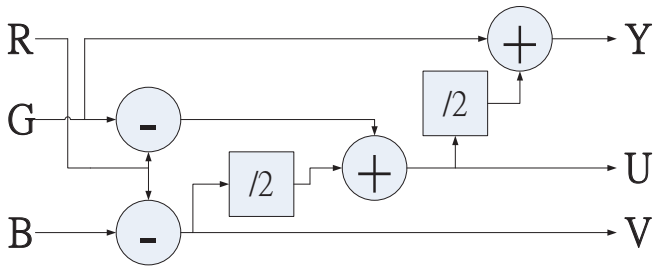


Figure 3.3: Structure of color space converter

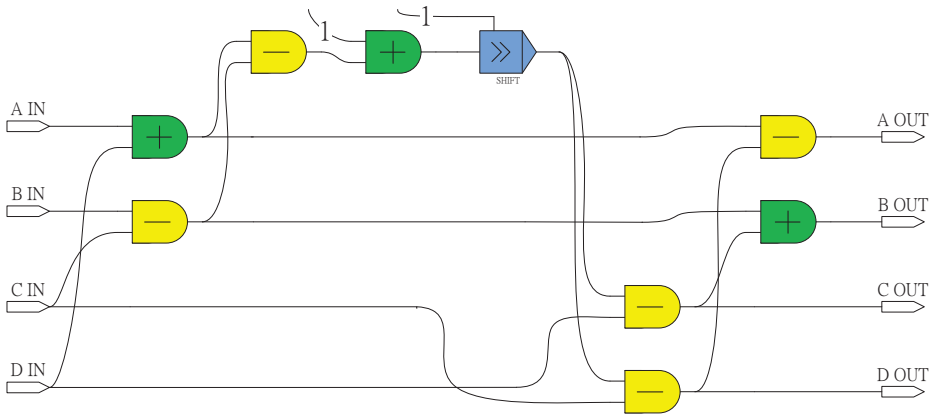


Figure 3.4: Structure overview of Th transform.

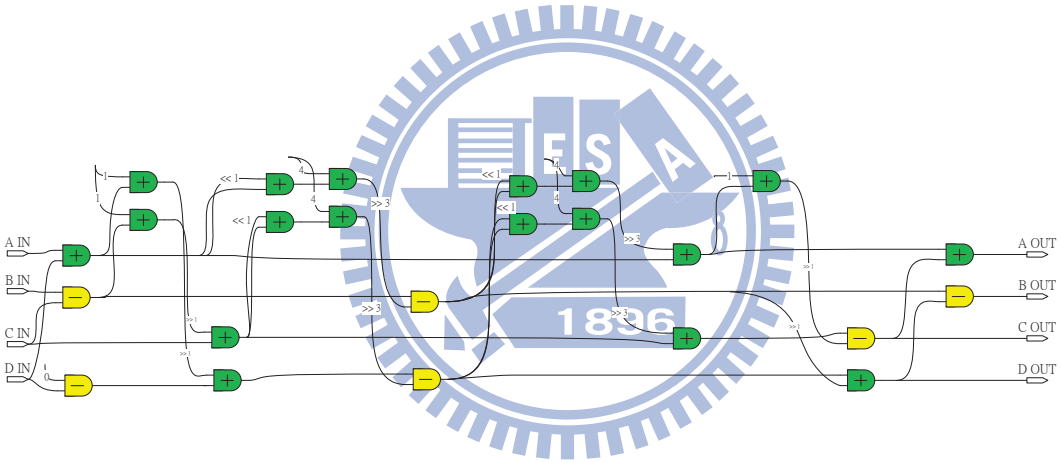


Figure 3.5: Structure overview of T odd transform.

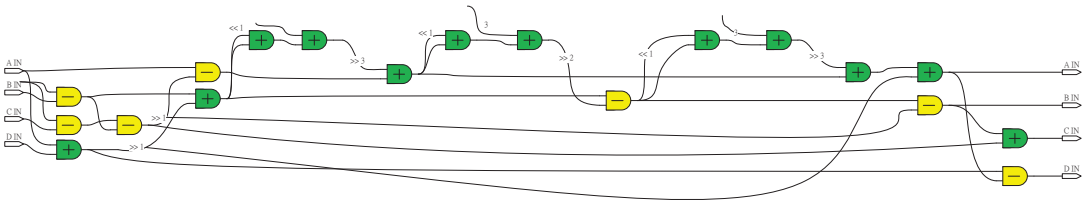


Figure 3.6: Structure overview of T oddodd transform.

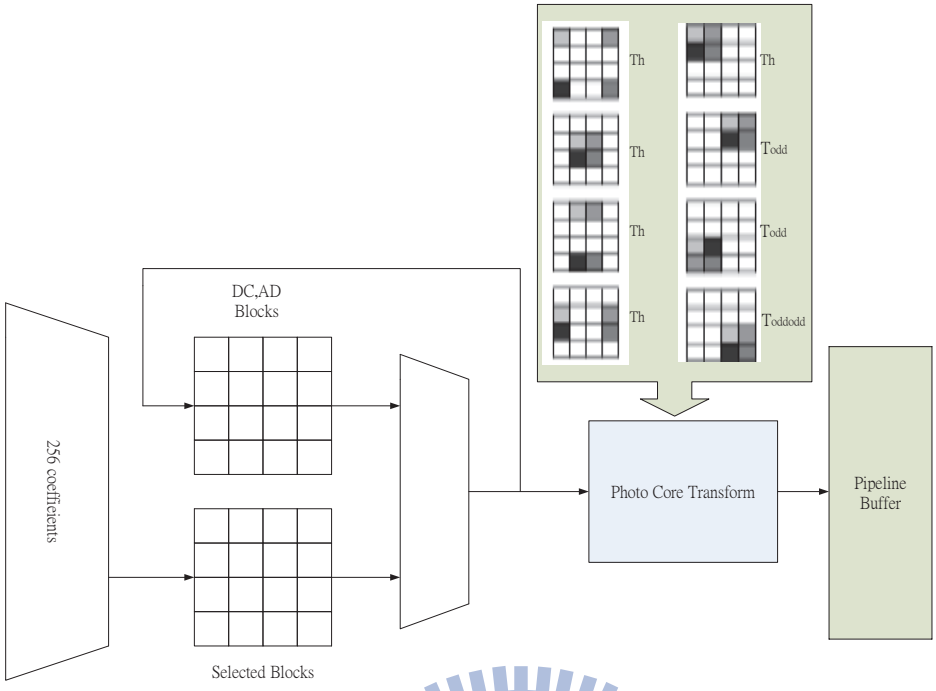


Figure 3.7: Architecture of full PCT stage.

The PCT process is designed with two stages. One Macroblock is devided into 16 blocks then 16 coefficients of each block are transferrd to the processing of main PCT architecture include T_H, T_{ODD}, T_{ODDODD} transform. Every transformer can process 4 coefficient pre clock and every block must be calculated with all three transform. After completion of the first transform stage, the 16 DC coefficients are stored and arranged in one new 4×4 block. MUX selector is designed and applied this new block into transforms as same as pct stage1 processing again. In the pct stage2 , the DC and AD bands are generated. The Y,U,V channels are independent and could be designed to parallel computing with no interfere with each other. Figure 3.7 shows the structure of full PCT transform.

3.6 Prediction structure

Prediction is the most simple unit of full JPEG XR implementation . The block diagram of our prediction module is shown in Figure 3.8. Prediction mode detectors are used

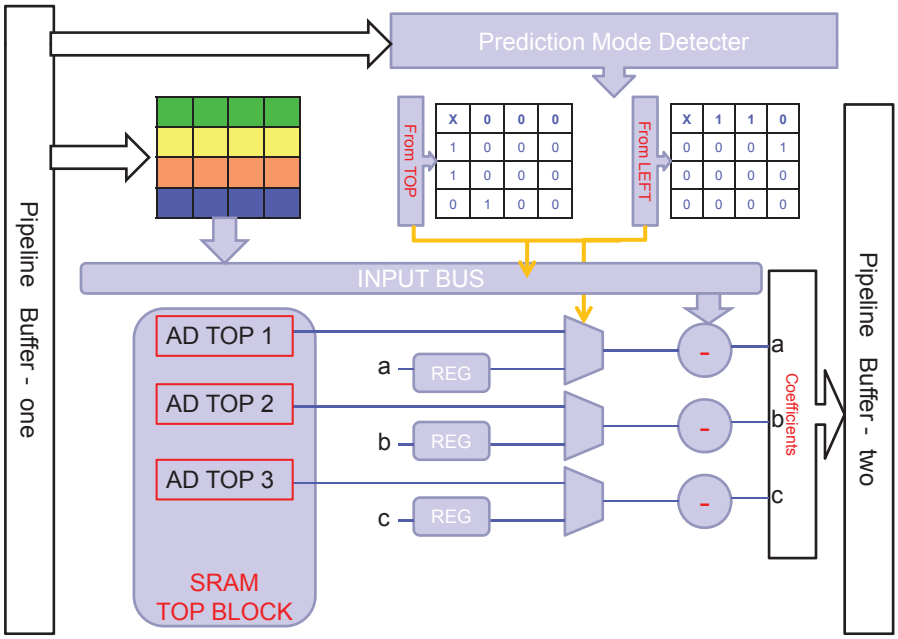


Figure 3.8: Structure overview of prediction.

to collect the necessary coefficients and calculate the correct prediction orientation for each DC,AD or AC band of target MB. The saved prediction value of neighboring top or left macroblock is output according to the prediction mode. FSM applied the selected block into Prediction Actuator and choose the right prediction value from TOP or LEFT coefficients registers. Since macroblocks are processed in raster scan order, the data for the current row of macroblocks should be stored. In the proposed architecture, the data of the neighboring top macroblock is stored in SRAM, and that of the neighboring top-left and left macroblocks is stored in registers. Prediction Actuator play the role to subtract the input coefficients and the corresponding selected prediction value, then save the result both to the registers of block and the pipeline sram for next stage use.

3.7 Coded block pattern

CBP is one special unit in JPEG XR design different to other image compression. CBP unit use 18 bits array for each channel of one macroblock to signal block which consist

ALL-Zero coefficients. Zero Detector is designed to check if the current coefficient is zero or not. After collection of status of all coefficients ,256 mark bits are arranged into three parts. One bit is for DC pattern ,and following 15 bits are re-organized with WIRE-OR operation then resulted new one bit. This bit is for AD symbol. The remaining 240 bits are re-organized with 15 bits as one group by same operation as AD region. AC band consume 15 bits for CBP symbols in each channel.

3.8 Structure of adaptive scan and entropy coding

The final stage of full JEEG XR architecture is the most complex part than other two stages. There are three main complex data dependency loops affect the encode performance of the operation in the entropy encoding module. The data flow dependency of entropy encoding is shown as Figure 3.9. The first one is the adaptive-scan function which is used to caculate coefficients distribution and refresh one new scan order table for

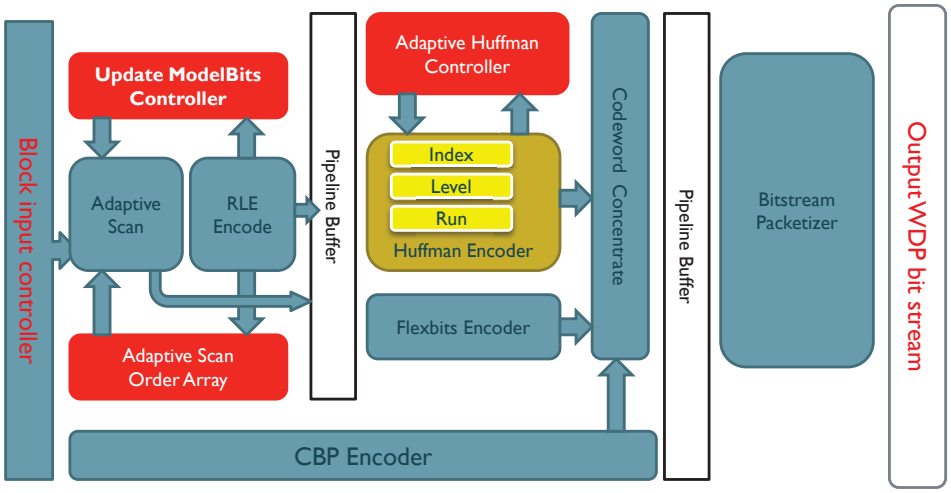


Figure 3.9: Structure overview of pipeline stage 3.

next block scan use. The second one is the Update ModelBits MB function block, which can decides how many bits are used to represent one coefficient and let RUN-LEVEL coefficients arrangement be more efficient. And the Adaptive Huffman Encode function choose the most efficient Huffman table to minimize the number of total entropy bits.

From the data flow path of Figure 3.9, the dependency path from the RLE Coder function block to the Update Model Bits block affect the efficiency of the encode. The scan order of DC/AD/AC coefficients decided by Adaptive Scan Order function block also require the feedback information from RLE Coder function block. The Adaptive Huffman Encode function block also needs to be updated according to the Index/Level/Run information. Coefficients after Adaptive Scan will be divided into two parts, one is bitstreams which is called Flexbits and another one is re-arranged coefficients coded by RLE module. JPEG XR use different Huffman Tables for each data ranked according to their symbol: Index, Level or Run. The Codewords of Index ,Level and Run are concentrated before packet to bitstream because it can reduce the loading of Packetizer then speed up the encoding.

Our proposed Entropy Stage of designed JPEG XR is also divided into three stages, as shown in Figure 3.9. The pipelined architecture can help to increase the throughput and decrease the timing of critical patch. Adaptive scan and Run-Level encoder precess in the first stage. Consider the feedback patch in this stage, related Update Modelbits Unit and “Scan Table Updater” are also designed in the same stage. Following Huffman Code Generator is placed in stage two, in the same, this stage should achieve related Adaptive Huffman Controller to update the optimal Huffman Table. The first codeword pre-concentration is also designed in this stage. In the final stage, Packagetizer arrange the concentrated codewords from entropy coding stage then output the wdp file. Ideally, the designed entropy coding can reduce the timing of critical path to 1/3 and increase the throughput about 3 times by well arranged pipeline timing schedule.

3.8.1 Data Dependency

The pipeline buffers are used to solve the problem of data dependency. In Entropy Coding Stage, one completely bitstream code is concentrated by information of Level and Run,

then the scanned data can be processed to huffman code when all required Level and Run are ready. First, one example of hype-pipeline structure is expressed. Assumes that one current block is processed in Entropy Coding Stage, which is shown as follows:

Block = [L1, R, R, R, L2, R, R, L3, L4, L5, End]

The designed “Level Reg” and “Run Reg” are used for hype-pipeline register between Adaptive Scan Stage and Huffman Coding Stage. In the first cycle of adaptive scan, Level Reg is updated to L1 and Run Reg is reset to Zero. After the 4th cycle of adaptive scan, the value of Run Reg is updated to R(3). In the 5th cycle, Adaptive Scan gets L2 and detects that counting of current Runs is finish. But the L2 can not be updated to Level Reg in this cycle, Huffman Coding Stage should be notified firstly to process the L1 which is kept in Level Reg and the R(3) of Run Reg. Otherwise, L1 of Level Reg has not yet been processed to huffman code but been refreshed to L2. Correct processing cycles of Adaptive Scan FSM are shown as following.

Adaptive Scan FSM = [L1, R(1), R(2), R(3), OK, L2, R(1), R(2), OK, L3, OK, L4, OK, L5, OK]

For Huffman Coding Stage, it requires 15 cycles to finish huffman code translation of Index, Level, and Run. Now considering the structure of pipeline buffers, the Level Register Array and Run Register Array are prepared to record the result of Adaptive Scan Stage. Adaptive Scan processes the coefficients of current block and arranges them to RLE format with 10 cycles. The arranged coefficients are recorded into register arrays which are shown as following.

Level Array = [L1, L2, L3, L4, L5]

Run Array = [R(3), R(2), R(0), R(0), R(0)]

For Huffman Coding Stage with pipeline buffers, it only requires 5 cycles to finish all huffman code translation. Adaptive Scan is unnecessary to check the status of Huffman Coding because of our proposed pipeline buffers, thus, the re-arranged result of adaptive scan stage can be recorded smoothly. Another coming advantage is that recorded data in the buffers can be send to each Huffman Look-Up Table independently. That is to say, huffman coding can translate Index, Level and Run to relative codewords without data dependency and improve the encode throughput. This advantage become more obvious

when parallel Adaptive Scan structure is designed. Because the parallel Adaptive Scan can process coefficients of one block faster than Huffman Coding. The data of parallel scanning may contains more than 2 Levels in one cycle, that means, the data dependency become worse when use hype-pipeline structure.

3.8.2 Adaptive scan structure

In the adaptive scanning module as shown in Figure 3.10, data of current block is selected to zero detector for Run/Level checking according to scan order. If input coefficient is detected as Level, the weight corresponding to the coefficient scan order is incremented by one. Then the weight of current scan order will be compared with last scan order. If the weight of current scan order is larger, the scan order of current one and last one should be exchange to optimal the adaptive scan. The updated scan order is obtained as the result of scan order sorting. After scan of one block finish, the scan order is sorted into newer status for next block using. After the processing of the adaptive scan module, then the rearranged coefficients are normalized according to ModelBits, and decomposed

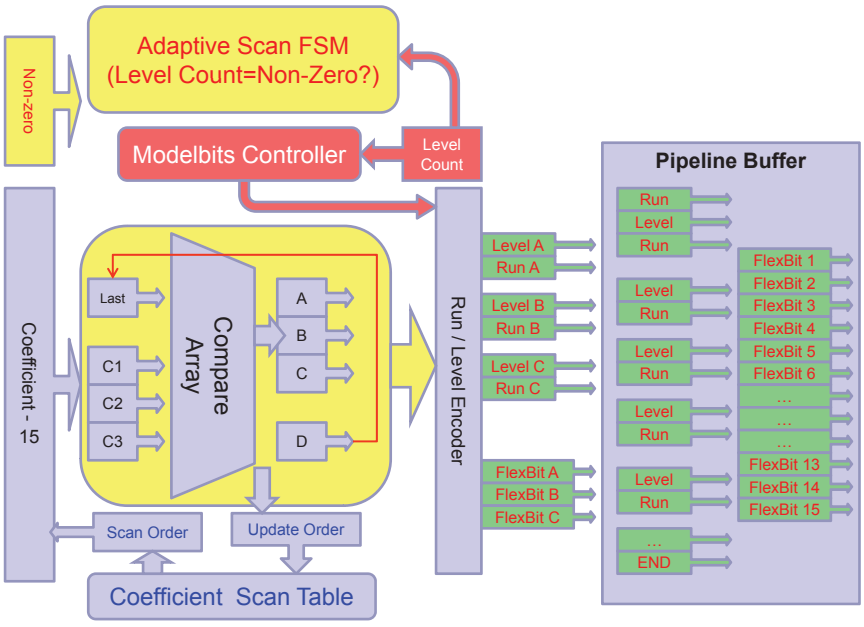


Figure 3.10: Structure overview of adaptive scan.

into normal data and FlexBits. Flexbits is composed from the shift-out coefficient with shifter which is controlled by ModelBits. JPEG XR group ModelBits into six types: DC band Y channel and C channel ,AD band Y channel and C channel, AC band Y channel and C channel . C channel include U and V channels. Normal data is sent to the run-level encoder and coded into runs and levels. ModelBits is also needed to be updated once in every MB based on the normalized result as total numbers of ”Levels”.

In our new designed architecture of adaptive scan, the FSM of this stage can receive the non-zero value of current block which is calculated when prediction stage and detect current scan sequence if all non-zero coefficients are processed. If the current scan counter reach to total need-to-scan level coefficients, FSM controller stop the current scan and save back the updated scan order. Then next scan block is began to set to initial status. Adaptive scan feedback mechanism will let coefficients which are probable Levels be arranged to the front of all scan sequence. Positive scan sequence detector is help to

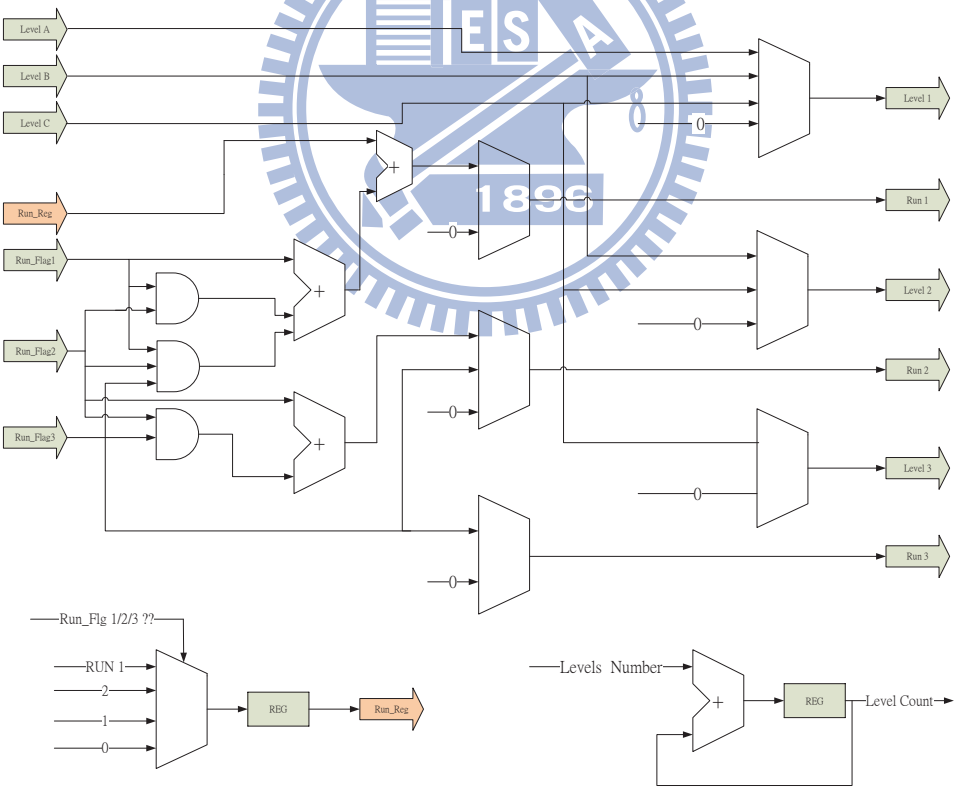


Figure 3.11: Implementation of Run Level Encoder

reduce un-necessary power consumption and save more time preparing for the next phase of coefficients scan.

As the analysis of adaptive scan result, the average number of normalized coefficients in high pass band is 5 Levels of one block. That is to say, in order to ensure the computing fluency of the entire entropy coding stage, adaptive scan stage should finish all coefficients scanning under 5 cycle. So the huffman coding stage can process the entropy coding without waiting. Our proposed adaptive scan structure improves the comparator array to faster 3 times than original design. Scan controller selects 3 coefficients in each cycle and processes Level Detection to match the updated weights by designed parallel comparator array. Adaptive scan processor swap the orders and weights according to result of comparison and refresh updated values to registers in next cycle. For each block composed with 15 coefficients, it just requires 5 cycles to be completed all coefficients scanning, which even be faster because of the adaptive order influence.

Run Level Encoder with parallel structure is also designed to determine and classify the multiple data from scanning results. Three scanned coefficients and their new updated flags namely "Runflag" are re-arranged to Run-Level-Coding format by this encoding unit. The first value of RUN array may contain four possible combinations: (A)Coefficient 1 is Level, (B)Coefficient 1 is Run, (C)Coefficients 1 and 2 are Runs, (D)Coefficients 1, 2 and 3 are all Runs. One particularly noteworthy logical judgment is that conditions(C) and (D) will not hold when condition(A) is true. According to the same logic, the condition(D) will not hold when (E)Coefficient 2 is Level, regardless of condition(A) is true or not. Two AND operators linked to the input Runflags serve as role as the above description. Runflag1 is connected to these two AND operators for checking condition(C) or (D) if true or false. Similarly, Runflag2 is also connected to the second and third AND operators for conditional judgment. Multiplexers are designed to arrange correct computed Runs and normalized Levels according to distribution of Runflags. Base on the updated counter of Levels, the organized results are saved to the buffers called RLCBuf and FlexBitsBuf. One prepared register in advance records the count of last Run to accumulate the passible continuing runs which should be send for next computing in next scan. Figure 3.11 shows the implementation of our purposed Run Level Encoder.

3.8.3 Huffman encoder

Figure 3.12 shows how the Level, Run and Index choose the suitable Huffman tables to generate the RLE codewords. The “Run” and “Level” are generated after RLE module, and “Index” is also created according to arrangement of the “Run” and “Level”. In the Huffman encoding stage, they are translated to the relative codewords by different Huffman index tables. According the type and value of input data, Codewords and Codesizes are generated from the corresponding Huffman Tables. All outputted Codewords are concentrated to become the bitstream of JPEG XR file. The Huffman encoder in the JPEG XR is different from the other standards. JPEG XR supply many but small Huffman tables for each type and can adaptively choose the best one from these tables to optimal the codesize of Run, Level and Index. Thus, JPEG XR can get the best compression ratio by following describing adaptive huffman controller.

After the Huffman encoding, three sets of codewords are produced and concentrated as one by pre-concentrater. Pre-concentrater is composed with 2 groups of “Barrel Shifter” and “Bitwise OR Operator”. First group of shifter shift the second concentrate codeword

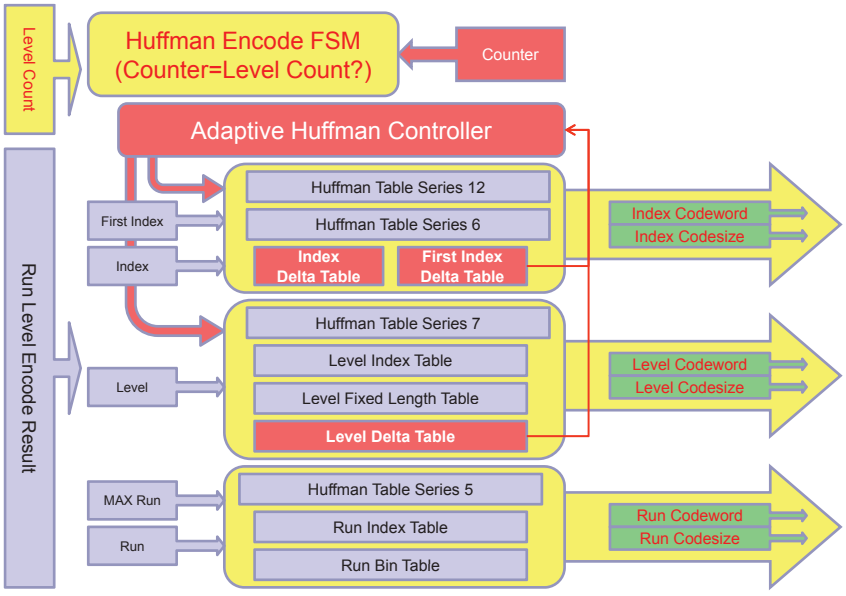


Figure 3.12: Structure overview of Huffman Coding.

according the first concentrate codesize. The shifted codeword will is arranged to the end of first codeword, then OR Operator is processed to combine these two codewords and the longer updated codeword is generated. The second group of concentrater process the third input codeword with the result of first concentrater. Architecture of Pre-Concentrater also be designed to contrate the Flexbits separated by Adaptive Scan stage. The structure of Pre-Concentrater is described as Figure 3.13 As the previous description of the above operation, the whole RLE codeword and RLE codesize will be produced to the packetizer of the entropy coding stage three.

3.8.4 Adaptive Huffman Coding

Runs and Levels are coded using adaptive Huffman tables, meanwhile, the difference of the code length which is coded from Delta Tables are also accumulated. The tables are changed if the accumulated result runs up or down to the threshold of detection. These thresholds for detection are called Upper Band and Lower Band. When accumulated result reach Upper Band, it means that the probability of partial Levels and Runs is relatively high. So the another designed Huffman Table may be suitable for next block

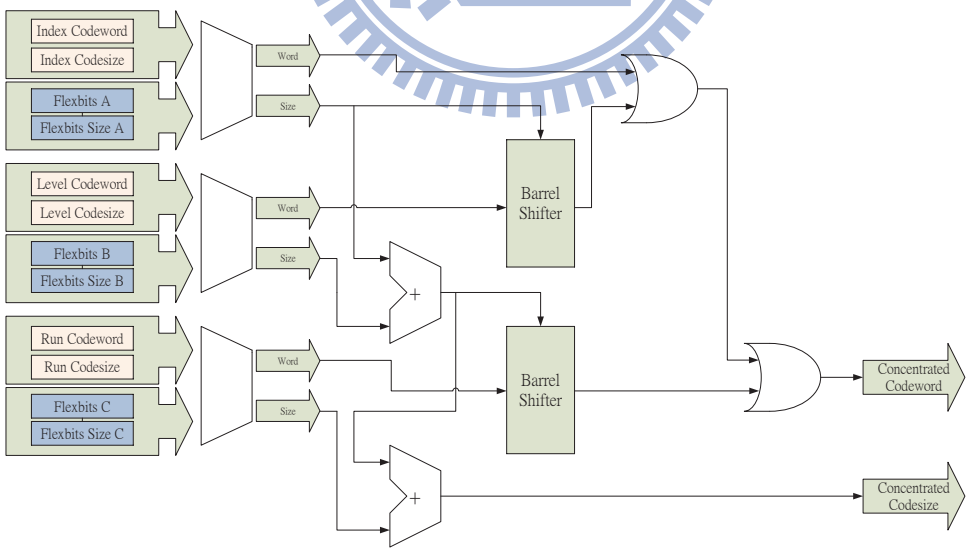


Figure 3.13: Structure of pre-concentrate for Huffman encode result.

encode, Huffman Table is updated. On the otherwise, the consequence of Lower Band will change the Huffman Table to the opposite side. Figure 3.14 shows the implementation of Adaptive Huffman Encoder. Discriminiant Register accumulates the value resulted from delta table, then the detection of upper/lower band generates the relative judgement for Table Selector to choose the suitable huffman table.

3.8.5 Architecture of Packetizer

The packetizer architecture is based on the [12] architecture which is shown in Figure 3.15. It is designed to combining all the RLE codewords and the FlexBits to generating the JPEG XR compressed file. The concentrate bus width support 32 bits for one complete format. The module detect the input codesize first, then shift the input codeword to the corresponding placement. ACC register accumlate the size of all input Codesizes and send out “OK” signal when amassed size reach 32 or higher. At the same time, the bus ”Word” is ready for main controller to collect and output JPEG XR file.

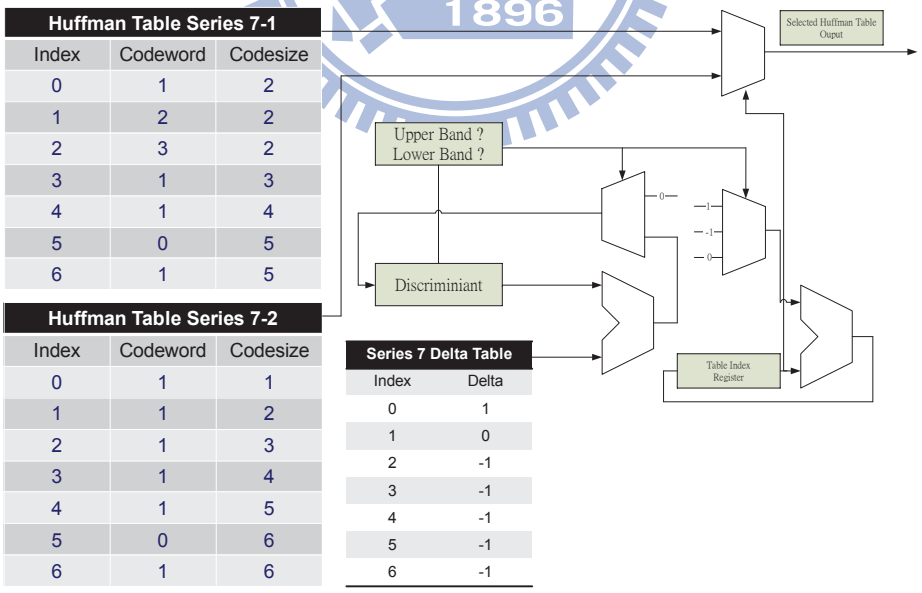


Figure 3.14: Structure overview of Adaptive Huffman Encoder.

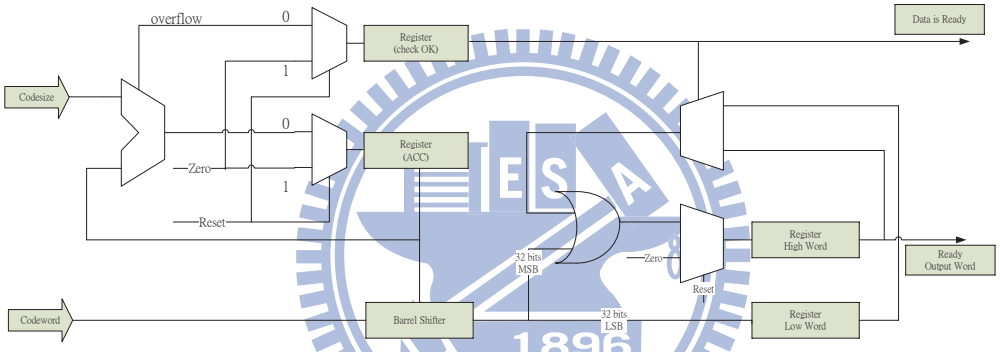


Figure 3.15: Implementation of Packetizer.

Chapter 4

Performance Analysis and Implementation Results

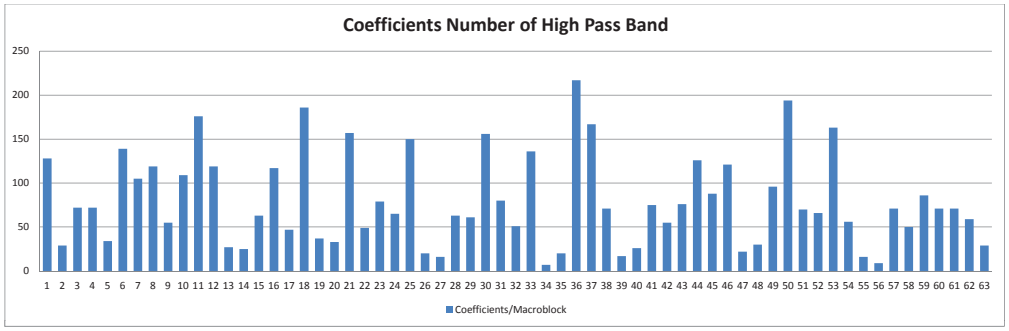
In this chapter, we analysis the performance of our implementation of JPEG XR encoder, including the encoding benchmark, and the coding comparison.

4.1 Performance Bottlenecks

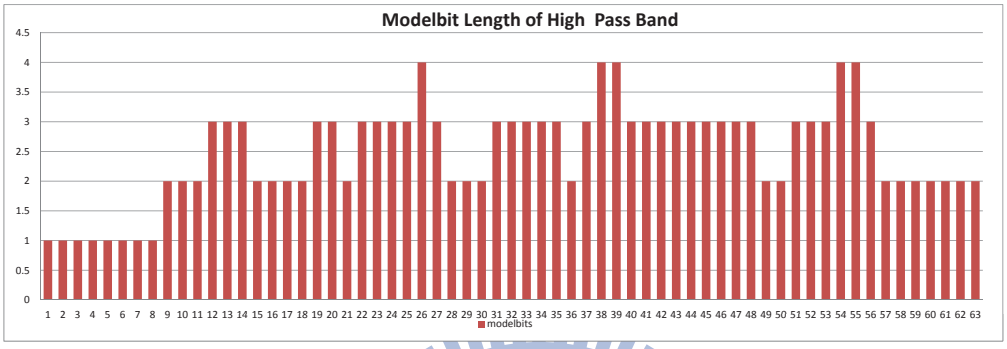
There are three feedback loops in adaptive scanning and entropy coding; (1) Control of ModelBits, (2) Updating of the scanning order, and (3) Decision of the Huffman table to be used. Therefore how to design a pipelined module in a straightforward implementation become a challenge. Used width in ModelBits should be calculated out at the end of the encoding of the current macroblock. The next block scan order is according to the computing result of scan weight in last block. The selection of Huffman table is also decided by constantly revised delta value from every used Huffman code. For these reasons, it is difficult to design processing macroblock in parallel encode structure.

4.2 Enhancement

Before discussing of performance improvements, we focus on some interesting and particular design of JPEG XR. Figure 4.1 shows the variation about “Update Modelbits”



(a)



(b)

Figure 4.1: Variation of High-Pass band update Modelbits in pre Macroblock

adjust bits-number relative to the sum of current macroblock in every band of every channel. One partial image with size 128×128 pixels captured from Lena.bmp is inputted for simulation, the total numbers of coefficients in high-pass band and the Modelbits width of current macroblock are recorded. The X-direction of Figure 4.1 count up for selected coded macroblock, the Y-direction of Figure 4.1(a) shows the coefficients number of High-Pass band and (b) shows the used ModelBits width in every current macroblock. Notice that the accumulated weight is growing up when high-throughput marcoblock is coding, and the Update ModelBits controller will increase the ModelBits size to improve whole Levels counts in next macroblock. Otherwise, ModelBits processor also reduce the ModelBits size when coding low-throughput macroblock in order to prevent the much Flexbits sizes influence the compress rate of JPEG XR bitstream.

Further, scan patterns in JPEG XR are allowed to change scan order with adaptive rules. The updated scan order enhance the performance of Run-Level Encode in positive way. That means the scan coefficients are arranged in the front of scan array as larger

probability as passible. Therefor, the architecture of all coding stages are designed with coefficients counter and detector to determine current processing if finish or not. The reason is described as above and the original Hard-Coding State Machine structure is replaced. FSMs of each stages complete all the need working and transit “Finish” information to another FSMs then ready for execution of next step. Processors no waste any energy in dealing with un-necessary operation.

Faster scan structure is also invented in 3 times than rudimental design. 15 coefficients of each block could be done in less than 5 clocks. Speed up the separation of Run-Level data and Flexbits advantage to accelerate the computing of Huffman encode stage. The latency of scan stage is shorter, the entropy coding stage could encode the data with minimal waiting and increase the throughput in straight. Consider that coefficients scanning result the Flexbits with large amount of data, and fixed 15 cluster of memory always be required to recode the Flexbits when ModelBits width is not zero. Pre-Concentrater is designed not only for concentrate Run and Level codewords, 3 Flexbits also be massed into one new sum of Flexbits. For packetizer, it just need more 5 clocks to collect all Flexbits into bitstreams of JPEG XR. The total numbers of RLE’s result and concentrated Flexbits in our design comes 10 in average of one block(High-Pass band). The statistic result lists as below Table 4.1

Table 4.1: Average clusters of one block after entropy coding result(HP part)

Pic Name	Lena.bmp	Baboon.bmp	Peppers.bmp	F16.bmp
Resolution	512×512	512×512	512×512	512×512
Blocks(HP)	*49152*	*49152*	*49152*	*49152*
Totals(HP)	*478609*	*465136*	*465785*	*466797*
Average	*9.74*	*9.46*	*9.48*	*9.5*

4.3 Coding Speed Estimation

After our optimization, estimation of the encoding speed in our implementation is listed in Table 4.2. The four images with same size but different manner are tested and represent quite similar results. The format of each image is 512×512 pixels and the color domain is RGB 24bits, then the whole sub-pixels in one image contains $512 \times 512 \times 3$ = 786432. In our measure, four tested images are encoded to wdp files with lossless mode and consuming time are also list in Table 4.2. The measure clock cycle is 10 ns, and the calculated throughput in terms of pixel/cycle shows that our implementation can achieve more than 1 pixel/cycle. The performance comparison with related works is described. In [9], JPEG XR encoding process is decomposed into (1) PCT/POT, (2)Quantization, (3) Prediction, (4) adaptive scanning, and (5) entropy coding. These five stages work in a pipeline manner. The throughput of this design is choked in the PCT/POT module and achieve 0.80 pixel/cycle. But this design process the entropy result and flexbits into bitstreams with CPU, seems no described any hardware architecture about Packetizer. In design [8], encoding process is decomposed into three main stages which are same to ours. Base on [8], [9], the process of third stage is decomposed into three phases, which work in a pipeline manner. This can contribute to the reduction of the timing of critical path to 1/3, and performance of this design is 112 fps for 4:4:4 CIF format at 62.5 MHz, which is equivalent to 0.54 pixel/cycle for one component.

The comparison of performance is summarized in Table 4.3. The throughput is listed for one component for the case of YUV 4:4:4. In [8] and [9],although the entropy coding is pipelined by three phases, the encoding process is bottlenecked by the entropy coding. In our proposed optimization, throughput is improved to 2 times than these related works. Therefore, our proposed architecture can achieve higher performance faster than other related words.

Our proposed JPEG XR encoder is designed by using Verilog-HDL to evaluate the present architecture. The architecture is synthesized by Synposys Design Compiler with 0.18 um CMOS standard cell library. The result is summarized in Table 4.4. The synthesis use 100MHz as the target frequency and our report about gate area is summarized in the

Table 4.2: Benchmark of JPEG XR encoder

Pic Name	Lena.bmp	Baboon.bmp	Peppers.bmp	F16.bmp
Resolution	512×512	512×512	512×512	512×512
Total Pixels(YUV444)	*786,432*	*786,432*	*786,432*	*786,432*
WDP File Size(Bytes)	*454K*	*591K*	*493K*	*395K*
Encoding Time(ns)	*7,359,835*	*7,198,545*	*7,233,145*	*7,244,795*
Throughput(Pixel/Cycle)	*1.069*	*1.092*	*1.087*	*1.085*

Table 4.3: Performance comparison among related works and the proposed architecture

Architecture	Throughput(Pixel/Cycle)	PS
[9]	0.54	
[10]	0.80	No Packetizer
[11]	1.58	Only PCT/POT
ours	1.00	

column named gate counts. In the column of SRAM , the sizes of SRAMs for each module are also summarized. The result shows that the gate count of the designed JPEG XR encoder is 235,377. The number of SRAMs is required by $992 \times 3channels$, which the predict buffer is configured as $480 \times 3channels$ when input image horizontal size is 1920 pixels.

Table 4.4: Gate count summarization of the proposed architecture

Paper	[9]		[10]		[11]		Ours	
Frequency	62.5 MHz	(0.18um)	125 MHz	(0.13um)	250 MHz	(90nm)	100 MHz	(0.18um)
Bus Width	32 bits		14 bits				16 bits	
Module name	Gate count	SRAM	Gate count	SRAM	Gate count	SRAM	Gate count	SRAM
PCT/POT Quantization	316,898	256×3	90,692	256×2	100,500 (POT1.2)	1640	73,340 (only PCT)	256×3
Prediction	81,980	256×3 480×3	6,726	120×7	None		62,029	256×3 480×3
CBP			Unknown		None		1,946	
Adaptive scan	105,323		41,990		None		63,009	
Entropy coding					None		32,069	
Packetizer			None		None		2,806	
Control Unit	1,866						178	
Top	506,167	992×3	142,157	1,352			235,377	992×3

Chapter 5

Conclusion

In this paper, we propose a novel and faster hardware architecture of JPEG XR, which is a new image coding standard and have advanced compression. One three-stage pipeline lossless JPEG XR encoder with YUV 4:4:4 was designed to support next generation HDR display. In previous architectures, the encoding throughput is limited in entropy coding stage because it was implemented to process coefficients according to the gathered statistics of running Macroblock.

We generalized the characteristic of Normalization(Update ModelBits) and took advantage of reduction of Levels. Our propose pipeline controller can optimal the forward step of the encoding to decrease un-necessary data processing. We could safely pipeline all the encoding processes including the entropy coding and achieves higher throughput than those of related works. In contrast to the complete and similar related architecture [9], our propose structure is twice as fast.

However, detection of finish in each processing stage and asymmetric pipeline execution time still cause the holding of processing unit and bring down the coding speed. Making higher record length of pipeline buffers or adaptive buffer controller may overcome this bottleneck remains as a future work.

Bibliography

[1] Microsoft Corporation, HD Photo specification version 1.0, Nov. 2006.

[2] S. Srinivasan, C. Tu, S. L. Regunathan, and G. J. Sullivan, "HD Photo: a new image coding technology for digital photography," in Proc. SPIE, vol.6696, Aug. 2007.

[3] Microsoft Corporation, "HD Photo device porting kit," Nov. 2006.

[4] ISO/IEC 15444-1: "Information technology X JPEG 2000 image coding system X part 1: core coding system.",2002.

[5] H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts," *IEEE Transactions on Signal Processing*, vol. 46, pp.1043 V- 1053, Apr. 1998.

[6] Maalouf, A.; Larabi, M.-C. "Low-complexity hierarchical lapped transform for lossy-to-lossless image coding in JPEG XR / HD Photo" Image Processing (ICIP), *16th IEEE International Conference on Digital Object Identifier*,pp.5 - 8,2009.

[7] S. Groder, Modeling and systhesis of the HD Photo compression algorithm, Masters thesis, Rochester Institute of Technology, Aug. 2008.

[8] C.-H. Pan, C.-Y. Chien, W.-M. Chao, S.-C. Huang, and L.-G. Chen, "Architecture design of full HD JPEG XR encoder for digital photography applications," *IEEE Transactions on Consumer Electronics*, vol. 54, pp.963 V- 971, Aug. 2008.

[9] C.-Y. Chien, S.-C. Huang, C.-H. Pan, C.-M. Fang, and L.-G. Chen, "Pipelined arithmetic encoder design for lossless JPEG XR encoder," in Proc. of 13th *IEEE International Symposium on Consumer Electronics*, pp.144 V- 147, May 2009.

[10] Hattori, K.; Tsutsui, H.; Ochi, H.; Nakamura, Y., “A High-Throughput Pipelined Architecture for JPEG XR Encoding” Embedded Systems for Real-Time Multimedia, *IEEE/ACM/IFIP 7th Workshop on Digital Object Identifier*, pp.9 - 17, 2009.

[11] Sheng-Wei Fan, Jia-Wai Chen, and Jiun-In Guo, LOW BANDWIDTH HD1080@60FPS JPEG-XR TRANSFORM DESIGN, *in VLSI Design, Automation, and Test (VLSI-DAT)*, Apr 2012.

[12] L.V. Agostini, I.S. Silva, and S. Bampi, “Pipelined Entropy Coders for JPEG Compression,” *Integrated Circuits and Systems Design, Proceedings 15th Symposium*, pp. 9 - 14, Sept. 2002