SWIFT COMPRESSED DOMAIN RESIDUE CALCULATION IN I4X4 MODE DECISION BASED ON INTEGER TRANSFORM IN H.264 ENCODER

[1]P. ESSAKI MUTHU, [2]Dr. R M O GEMSON
[1]Research Scholar, Department of ECE, Dr. MGR Educational and Research Institute, Chennai, INDIA
[2]Professor, Department of ECE, Sambhram Institute of Technology, Bangalore, INDIA
E-mail: [1]pessakimuthu@yahoo.com, [2]mogratnam@rediffmail.com

ABSTRACT

H.264 encoder has to evaluate exhaustively all the mode combinations of intra 4x4 predictions for deciding coding mode to achieve the minimum coding bits and high video quality. An Integer Transform-based Intra 4x4 Prediction method is proposed here to reduce the computational complexity of Intra 4x4 mode decision. This method calculates the transform domain residues in three steps, 1) calculating predicted coefficients directly 2) performing transform on input original coefficients and 3) finds the tranform domain residue coefficients by subtracting the transform domain predicted coefficients from transform domain original coefficients. The experimental results demonstrate that proposed method reduces at least 40% of computational complexity without compromising the coding efficiency (bitrate) and performance (quality).

Keywords: *H.264/AVC, Mode decision, Transform domain Intra 4x4 Prediction, Computational Complexity*

1. INTRODUCTION

The widely used, international video coding standard H.264/ advanced video coding (AVC) [1], achieves significantly better performance in both peak signal-to-noise ratio (PSNR) and visual quality at the same bit rate compared with prior video coding standards. H.264/AVC can save up to 39%, 49%, and 64% of the bit rate, when compared with MPEG-4, H.263, and MPEG-2 [2].

In order to perform Intra Mode Decision and Motion Estimation, H.264 Encoders use one important technique called Lagrangian based rate-distortion optimization (RDO). The RDO technique has higher computational complexity than the traditional SATD technique, but it gives lesser bits and lower distortion. SATD technique has less computational complexity, but it generates more bits with higher distortion. It is because SATD focuses only on distortion, not on bits, and SATD works based on Hadamard Transform. This paper focuses on Integer transform based Intra mode decision with reduced computational complexity than RDO method, but keeping same bitrate and quality of RDO method.

The method of calculating the computational complexity of the intra prediction module was done by number of cycles taken by basic operations [3]. Mustafa et al [4], [5] used number of additions and shifts performed in intra prediction, as metric for computational complexity. But the number of cycles for basic operations varies from processor to system. The metric for computational complexity used here is number of additions, subtractions and number of shifts.

Intra mode decision in H.264 Compression has two major types namely Intra4x4 Mode Decision and Intra16x16 Mode Decision. This paper focuses on Intra4x4 Mode Decision, because the same method can be adopted for Intra16x16 Mode Decision.

Intra 4x4 prediction has nine different modes and these prediction modes have common equations among them. Calculating the predicted values using these common equations for each mode is unnecessary. The Pixel equality based computation reduction (PECR) technique used in [4] focused only on pixel domain predictions. It saved computational complexity, but it did not include the computational complexity for comparison operation which is generally equivalent to addition operation. Still there is a scope of optimization in the data reuse techniques presented in [5] to reduce the computational complexity.

The techniques used in [6], [7], [8], [9] and [10] reduce the amount of computations performed by

Journal of Theoretical and Applied Information Technology
30th September 2014. Vol. 67 No.3
© 2005 - 2014 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Intra prediction by trying selected intra prediction modes rather than trying all possible intra prediction modes. The reference software, [12] and [13], use all the possible modes in RDO based mode decision.

The steps involved for each Intra4x4 mode are (1) finding predicted values (2) finding residue values (3) performing forward transform on residue values, (4) performing quantization to generate quantized residues (5) finding number of bits required for quantized residues (6) performing dequantization and reconstruction (7) finding distortion between original and reconstruction, (8) calculating RD-Cost. The above-said steps are done sequentially for all modes. At last, best mode is decided based on RD-Costs among all modes.

This paper focuses on deriving transform domain residue coefficients (Steps 1 to 3) for all modes at once. The proposed method can be extended or altered to all other block sizes also, if RDO technique is adopted for its mode decision.

The objective of this paper is (i) to calculate the computational complexity of pixel domain prediction of all nine Intra4x4 modes and its forward transformation used in [12] and [13], (ii) to reduce the computational complexity of Intra 4x4 Prediction of a given 4x4 sub-macroblock (sMB), and (iii) to give the transform domain residue directly.

This paper is organized in five sections. Section 2 and Section 3 describe the computational complexity of traditional method of mode decision. Section 2 explains the pixel domain Intra 4x4 Prediction for all modes and its computational complexity. Section 3 explains the computational complexity of finding residue and its standard forward transform. Section 4 explains the novel method to find key pixel domain predicted values, transformed coefficients and its computational complexity. Finally, in Section 5, the comparison of complexity between the proposed and the reference software [12] and [13] and the inferences of the proposed method are presented.

2. COMPLEXITY OF PIXEL DOMAIN INTRA 4x4 PREDICTION

An Intra macroblock (I-MB) is coded without referring to any data outside the current slice. I-MBs may occur in any slice type. Every Macroblock (MB) in an I-slice is an I-MB. I-MBs are coded using intra prediction, i.e. prediction from previously-coded data in the same slice. For a typical block of Luma or Chroma samples, there is a relatively high correlation between samples in the block and samples that are immediately adjacent to the block. Intra prediction therefore uses samples from adjacent, previously coded blocks to predict the values in the current block [11].

There are nine different Intra 4x4 Prediction modes namely, Vertical, Horizontal, DC, Diagonal-Down-Left, Diagonal-Down-Right, Vertical-Right, Horizontal-Down, Vertical-Left and Horizontal-Up, possible for a given 4x4 sMB in Intra 4x4 Prediction. The best mode is decided based on the Rate-Distortion Cost (RD-Cost) calculated for each mode Predictions.

In the process of RD-Cost calculation, each 4x4 sMB is subjected to all nine prediction modes. The 4x4 sMB is predicted by all nine different prediction techniques. The residue 4x4 sMB is found out by subtracting the predicted 4x4 sMB from original 4x4 sMB. There are nine sets of residue 4x4 sMBs, which are core forward transformed and quantized. The rate is calculated for each prediction mode by subjecting the quantized coefficients to variable length coding. The quantized coefficients are dequantized and core inverse transformed to get reconstructed residue 4x4 sMBs. These reconstructed residue 4x4 sMBs are added with already predicted 4x4 sMBs to get reconstructed 4x4 sMBs. The distortions are calculated between original 4x4 sMB and reconstructed 4x4 sMBs, by means of Sum of Absolute Error (SAE) or Sum of Squares of Error (SSE). For all 9 different modes, the RD-Cost are calculated by the following equation.

$$RD\_Cost = Distortion + \lambda.Rate \qquad (1)$$

The minimum RD-Cost will decide the mode for the given 4x4 sMB. The same process is continued for other fifteen 4x4 sMBs in a MB. This way of finalizing Intra 4x4 Prediction modes are followed by Joint Model (JM) Group [12] and x264 [13].

M	A	B	C	D	E	F	G	H
I	a	b	C	d				
J	e	f	G	h				
K	i	j	K	l				
L	m	n	O	p				

←Predicted Block

Figure 1: Pictorial Representation Of Neighbourhood In Intra Prediction

The intra 4x4 prediction mainly depends upon the eight pixels in the previous row (A, B, C, D, E, F, G and H), four pixels in the previous column (I, J, K, and L) and a top-left pixel (M) of reconstructed frame with respect to co-location as shown in Figure 1.

2.1 Vertical (VER) Mode

Vertical (VER) mode copies down the top four pixels (A, B, C and D) as predicted block shown below. As there are only copying operations, the computational complexity of Vertical Prediction is zero.

$$VER\_Px = \begin{pmatrix} A & B & C & D \\ A & B & C & D \\ A & B & C & D \\ A & B & C & D \end{pmatrix}$$

2.2 Horizontal (HOR) Mode

Horizontal (HOR) mode copies right the left four pixels (I, J, K and L) as predicted block shown below. As there are only copying operations, the computational complexity of Horizontal Prediction is zero.

$$HOR\_Px = \begin{pmatrix} I & I & I & I \\ J & J & J & J \\ K & K & K & K \\ L & L & L & L \end{pmatrix}$$

2.3 DC Prediction (DC) Mode

DC Prediction (DC) mode finds the average of top four pixels (A, B, C and D) and left four pixels (I, J, K, and L), and copies the average in the entire predicted block.

$$Av = (A + B + C + D + I + J + K + L + 4) \gg 3$$

If the any of top or left four pixels are not available, the Av is calculated by averaging the available four pixels.

$$Av = \begin{cases} (A + B + C + D + 2) \gg 2, & LEFT\ not\ available \\ (I + J + K + L + 2) \gg 2, & TOP\ not\ available \end{cases}$$

If the current block is first in the frame, then Av is the predicted as half of Pixel bitwidth. The Pixel domain DC Predicted values are,

$$DC\_Px = \begin{pmatrix} Av & Av & Av & Av \\ Av & Av & Av & Av \\ Av & Av & Av & Av \\ Av & Av & Av & Av \end{pmatrix}$$

Considering the best case, the computational complexity is 8 additions and 1 shift operations.

2.4 Diagonal-Down-Left (DDL) Mode

In DDL mode Prediction, the pixel domain values (DDL_Px) are calculated as follows.

$$DDL\_Px = \begin{pmatrix} DDL(1,1) & DDL(1,2) & DDL(1,3) & DDL(1,4) \\ DDL(1,2) & DDL(1,3) & DDL(1,4) & DDL(2,4) \\ DDL(1,3) & DDL(1,4) & DDL(2,4) & DDL(3,4) \\ DDL(1,4) & DDL(2,4) & DDL(3,4) & DDL(4,4) \end{pmatrix}$$

There are only 7 unique pixel values which are calculated from neighbourhood pixels. The other 9 pixels are copied from the seven unique pixel values. The 7 unique pixels are calculated as follows.

$$\begin{aligned} DDL(1,1) &= (A + B \ll 1 + C + 2) \gg 2 \\ DDL(1,2) &= (B + C \ll 1 + D + 2) \gg 2 \\ DDL(1,3) &= (C + D \ll 1 + E + 2) \gg 2 \\ DDL(1,4) &= (D + E \ll 1 + F + 2) \gg 2 \\ DDL(2,4) &= (E + F \ll 1 + G + 2) \gg 2 \\ DDL(3,4) &= (F + G \ll 1 + H + 2) \gg 2 \\ DDL(4,4) &= (G + H \ll 1 + H + 2) \gg 2 \end{aligned}$$

The computational complexity of DDL Prediction is 21 additions and 14 shifts.

2.5 Diagonal-Down-Right (DDR) Mode

In DDR mode Prediction, the pixel domain values (DDR_Px) are calculated as below.

$$DDR\_Px = \begin{pmatrix} DDR(1,1) & DDR(1,2) & DDR(1,3) & DDR(1,4) \\ DDR(2,1) & DDR(1,1) & DDR(1,2) & DDR(1,3) \\ DDR(3,1) & DDR(2,1) & DDR(1,1) & DDR(1,2) \\ DDR(4,1) & DDR(3,1) & DDR(2,1) & DDR(1,1) \end{pmatrix}$$

There are 7 pixel values which are calculated from neighbourhood pixels. Out of 7 pixel values, five are unique and two can be derived from DDL Prediction.

$$\begin{aligned} DDR(1,1) &= (A + M \ll 1 + I + 2) \gg 2 \\ DDR(1,2) &= (M + A \ll 1 + B + 2) \gg 2 \\ DDR(1,3) &= (A + B \ll 1 + C + 2) \gg 2 = DDL(1,1) \\ DDR(1,4) &= (B + C \ll 1 + D + 2) \gg 2 = DDL(1,2) \\ DDR(2,1) &= (M + I \ll 1 + J + 2) \gg 2 \\ DDR(3,1) &= (I + J \ll 1 + K + 2) \gg 2 \\ DDR(4,1) &= (J + K \ll 1 + L + 2) \gg 2 \end{aligned}$$

The computational complexity of DDR Prediction is 21 additions and 14 shifts.

2.6 Vertical-Right (VR) Mode

The pixel domain Vertical-Right (VR) mode Prediction values are given as follows.

$$VR\_Px = \begin{pmatrix} VR(1,1) & VR(1,2) & VR(1,3) & VR(1,4) \\ VR(2,1) & VR(2,2) & VR(2,3) & VR(2,4) \\ VR(3,1) & VR(1,1) & VR(1,2) & VR(1,3) \\ VR(4,1) & VR(2,1) & VR(2,2) & VR(2,3) \end{pmatrix}$$

VR mode has 10 distinct pixel values. Among that, only four are unique pixel values and the other six values can be derived from DDL and DDR Predictions.

$$\begin{aligned}
&VR(1,1) = (M + A + 1) \gg 1\\
&VR(1,2) = (A + B + 1) \gg 1\\
&VR(1,3) = (B + C + 1) \gg 1\\
&VR(1,4) = (C + D + 1) \gg 1\\
&VR(2,1) = (A + M \ll 1 + I + 2) \gg 2 = DDR(1,1)\\
&VR(2,2) = (M + A \ll 1 + B + 2) \gg 2 = DDR(1,2)\\
&VR(2,3) = (A + B \ll 1 + C + 2) \gg 2 = DDL(1,1)\\
&VR(2,4) = (B + C \ll 1 + D + 2) \gg 2 = DDL(1,2)\\
&VR(3,1) = (M + I \ll 1 + J + 2) \gg 2 = DDR(2,1)\\
&VR(4,1) = (I + J \ll 1 + K + 2) \gg 2 = DDR(3,1)
\end{aligned}$$

The computational complexity of VR Prediction is 26 additions and 16 shifts.

2.7 Horizontal-Down (HD) Mode

The pixel domain Horizontal-Down (HD) mode Prediction values are given below.

$$HD\_Px = \begin{pmatrix} HD(1,1) & HD(1,2) & HD(1,3) & HD(1,4)\\ HD(2,1) & HD(2,2) & HD(1,1) & HD(1,2)\\ HD(3,1) & HD(3,2) & HD(2,1) & HD(2,2)\\ HD(4,1) & HD(4,2) & HD(3,1) & HD(3,2) \end{pmatrix}$$

HD mode has 10 distinct pixel values given below. Out of those, 6 values can be derived from DDL and DDR Predictions.

$$\begin{aligned}
&HD(1,1) = (M + I + 1) \gg 1\\
&HD(1,2) = (A + M \ll 1 + I + 2) \gg 2 = DDR(1,1)\\
&HD(1,3) = (M + A \ll 1 + B + 2) \gg 2 = DDR(1,2)\\
&HD(1,4) = (A + B \ll 1 + C + 2) \gg 2 = DDL(1,1)\\
&HD(2,1) = (I + J + 1) \gg 1\\
&HD(2,2) = (M + I \ll 1 + J + 2) \gg 2 = DDR(2,1)\\
&HD(3,1) = (J + K + 1) \gg 1\\
&HD(3,2) = (I + J \ll 1 + K + 2) \gg 2 = DDR(3,1)\\
&HD(4,1) = (K + L + 1) \gg 1\\
&HD(4,2) = (J + K \ll 1 + L + 2) \gg 2 = DDR(4,1)
\end{aligned}$$

The computational complexity of HD Prediction is 26 additions and 16 shifts.

2.8 Vertical-Left (VL) Mode

The pixel domain Vertical-Left (VL) mode Prediction values are shown below.

$$VL\_Px = \begin{pmatrix} VL(1,1) & VL(1,2) & VL(1,3) & VL(1,4)\\ VL(2,1) & VL(2,2) & VL(2,3) & VL(2,4)\\ VL(1,2) & VL(1,3) & VL(1,4) & VL(3,4)\\ VL(2,2) & VL(2,3) & VL(2,4) & VL(4,4) \end{pmatrix}$$

VL mode has 10 distinct pixel values. Among that, only two are unique pixel values and the other eight values can be derived from DDL and VR Predictions.

$$\begin{aligned}
&VL(1,1) = (A + B + 1) \gg 1 = VR(1,2)\\
&VL(1,2) = (B + C + 1) \gg 1 = VR(1,3)\\
&VL(1,3) = (C + D + 1) \gg 1 = VR(1,4)\\
&VL(1,4) = (D + E + 1) \gg 1\\
&VL(2,1) = (A + B \ll 1 + C + 2) \gg 2 = DDL(1,1)\\
&VL(2,2) = (B + C \ll 1 + D + 2) \gg 2 = DDL(1,2)\\
&VL(2,3) = (C + D \ll 1 + E + 2) \gg 2 = DDL(1,3)\\
&VL(2,4) = (D + E \ll 1 + F + 2) \gg 2 = DDL(1,4)\\
&VL(3,4) = (E + F + 1) \gg 1\\
&VL(4,4) = (E + F \ll 1 + G + 2) \gg 2 = DDL(2,4)
\end{aligned}$$

The computational complexity of VL Prediction is 25 additions and 15 shifts.

2.9 Horizontal-Up (HU) Mode

The pixel domain Horizontal-Up (HU) mode Prediction values are shown below.

$$HU\_Px = \begin{pmatrix} HU(1,1) & HU(1,2) & HU(1,3) & HU(1,4)\\ HU(1,3) & HU(1,4) & HU(2,3) & HU(2,4)\\ HU(2,3) & HU(2,4) & L & L\\ L & L & L & L \end{pmatrix}$$

HU mode has 7 distinct pixel values. Out of those, 5 pixel values can be derived from DDR and HD Predictions and one pixel value is L itself.

$$\begin{aligned}
&HU(1,1) = (I + J + 1) \gg 1 = HD(2,1)\\
&HU(1,2) = (I + J \ll 1 + K + 2) \gg 2 = DDR(3,1)\\
&HU(1,3) = (J + K + 1) \gg 1 = HD(3,1)\\
&HU(1,4) = (J + K \ll 1 + L + 2) \gg 2 = DDR(4,1)\\
&HU(2,3) = (K + L + 1) \gg 1 = HD(4,1)\\
&HU(2,4) = (K + L \ll 1 + L + 2) \gg 2
\end{aligned}$$

The computational complexity of HU Prediction is 15 additions and 9 shifts.

For a 4x4 sMB, all possible predictions are done for all the nine modes. The computational Complexity of Intra 4x4 Prediction is 142 additions and 85 shifts in [12] and [13] (Table 2 and Table 3).

3. COMPLEXITY OF FINDING RESIDUE AND FORWARD TRANSFORM

Each predicted 4x4 sMB is subtracted from the original 4x4 sMB to get residue 4x4 sMBs. The computational complexity for one 4x4 sMB is 16 subtractions and zero shift. The computational complexity of finding residue for all nine prediction modes is 9x16 subtractions, i.e., 144 addition and zero shift operations in [12] and [13] (Table 3).

The residue sMB is core forward transformed using the following equation.

$$RES = Tx(res) = Cf \times res \times Cft$$

$$where\ Cf = \begin{pmatrix} 1 & -1 & -1 & -1 \\ 2 & -1 & -1 & -2 \\ 1 & -1 & -1 & -1 \\ 1 & -2 & -2 & -1 \end{pmatrix}$$

$$and\ Cft = transpose(Cf)$$

This core forward transform (It is a matrix multiplication) is simplified with additions and shifts only as follows.

```
for row = 1 to 4
   f(row,1) = res(row,1) + res(row,4)
   f(row,2) = res(row,2) + res(row,3)
   f(row,3) = res(row,2) - res(row,3)
   f(row,4) = res(row,1) - res(row,4)
end
for row = 1 to 4
   g(row,1) = f(row,1) + f(row,2)
   g(row,2) = f(row,3) + f(row,4) << 1
   g(row,3) = f(row,1) - f(row,2)
   g(row,4) = f(row,4) - f(row,3) << 1
end
for col = 1 to 4
   h(1,col) = g(1,col) + g(4,col)
   h(2,col) = g(2,col) + g(3,col)
   h(3,col) = g(2,col) - g(3,col)
   h(4,col) = g(1,col) - g(4,col)
end
for col = 1 to 4
   RES(1,col) = h(1,col) + h(2,col)
   RES(2,col) = h(3,col) + h(4,col) << 1
   RES(3,col) = h(1,col) - h(2,col)
   RES(4,col) = h(4,col) - h(3,col) << 1
 end
```

Algorithm to perform core forward transform in [12] and [13]

The computational complexity of core forward transform of one residue 4x4 sMB is 64 additions and 16 shifts. For all nine modes, the computational complexity is 9x64 additions and 9x16 shifts, i.e., 576 addition and 144 shift operations in [12] and [13] (Table 3).

4. PROPOSED METHOD WITH COMPUTATIONAL COMPLEXITY

It is clearly found that there are 24 unique pixel values for all nine intra 4x4 modes. The number of unique pixel values corresponding to each mode is listed in the Table 1.

$DC(1,1) = (A + B + C + D + E + F + G + H + 4) \gg 3$
$DDL(1,1) = (A + B \ll 1 + C + 2) \gg 2$
$DDL(1,2) = (B + C \ll 1 + D + 2) \gg 2$
$DDL(1,3) = (C + D \ll 1 + E + 2) \gg 2$
$DDL(1,4) = (D + E \ll 1 + F + 2) \gg 2$
$DDL(2,4) = (E + F \ll 1 + G + 2) \gg 2$
$DDL(3,4) = (F + G \ll 1 + H + 2) \gg 2$
$DDL(4,4) = (G + H \ll 1 + H + 2) \gg 2$
$DDR(1,1) = (A + M \ll 1 + I + 2) \gg 2$
$DDR(1,2) = (M + A \ll 1 + B + 2) \gg 2$
$DDR(2,1) = (M + I \ll 1 + J + 2) \gg 2$
$DDR(3,1) = (I + J \ll 1 + K + 2) \gg 2$
$DDR(4,1) = (J + K \ll 1 + L + 2) \gg 2$
$HU(2,4) = (K + L \ll 1 + L + 2) \gg 2$
$VR(1,1) = (M + A + 1) \gg 1$
$VR(1,2) = (A + B + 1) \gg 1$
$VR(1,3) = (B + C + 1) \gg 1$
$VR(1,4) = (C + D + 1) \gg 1$
$VL(1,4) = (D + E + 1) \gg 1$
$VL(3,4) = (E + F + 1) \gg 1$
$HD(1,1) = (M + I + 1) \gg 1$
$HD(2,1) = (I + J + 1) \gg 1$
$HD(3,1) = (J + K + 1) \gg 1$
$HD(4,1) = (K + L + 1) \gg 1$

Mode	Number of unique pixel values
DC Prediction	1
Diagonal-Down-Left Prediction	7
Diagonal-Down-Right Prediction	5
Vertical-Right Prediction	4
Horizontal-Down Prediction	4
Vertical-Left Prediction	2
Horizontal-Up Prediction	1
Total	24

Table 1 Number of unique pixel values for each mode

The other pixel values are copied from the already calculated pixel values by the above-said 24 equations. Thus reduces the computational complexity to 67 addition and 37 shift operations. But those 24 equations are split intelligently and modified as follows.

$AB = A + B + 1$
$BC = B + C + 1$
$CD = C + D + 1$
$DE = D + E + 1$
$EF = E + F + 1$
$FG = F + G + 1$
$GH = G + H + 1$
$AM = A + M + 1$
$MI = M + I + 1$
$IJ = I + J + 1$
$JK = J + K + 1$
$KL = K + L + 1$

$VR(1,2) = AB \gg 1$
$VR(1,3) = BC \gg 1$
$VR(1,4) = CD \gg 1$
$VR(1,1) = AM \gg 1$

$VL(1,4) = DE \gg 1$
$VL(3,4) = EF \gg 1$

$HD(1,1) = MI \gg 1$
$HD(2,1) = IJ \gg 1$
$HD(3,1) = JK \gg 1$
$HD(4,1) = KL \gg 1$

$DC(1,1) = (AB + CD + EF + GH) \gg 3$

$DDL(1,1) = (AB + BC) \gg 2$
$DDL(1,2) = (BC + CD) \gg 2$
$DDL(1,3) = (CD + DE) \gg 2$

$DDL(1,4) = (DE + EF) \gg 2$
$DDL(2,4) = (EF + FG) \gg 2$
$DDL(3,4) = (FG + GH) \gg 2$
$DDL(4,4) = (GH + H \ll 1 + 1) \gg 2$

$DDR(1,1) = (AM + MI) \gg 2$
$DDR(1,2) = (AM + AB) \gg 2$
$DDR(2,1) = (MI + IJ) \gg 2$
$DDR(3,1) = (IJ + JK) \gg 2$
$DDR(4,1) = (JK + KL) \gg 2$

$HU(2,4) = (KL + L \ll 1 + 1) \gg 2$

Now, computational complexity of finding pixel domain predictions for all nine modes is 42 addition and 26 shift operations (Table 4).

The facts of transform domain predicted coefficients are given below.

1) The pixel domain predicted 4x4 sMB of each mode has spatial redundancy. When predicted sMB is subtracted from original sMB to form residue sMB, this spatial redundancy is not utilized/exploited. So, the pixel domain predicted values are core forward transformed to exploit the spatial redundancy.
2) The additive property of Core Forward Transform is shown below.
$Tx(A\text{-}B) = Tx(A) - Tx(B)$
3) Using the above-said equation and the spatial redundancy of pixel domain predicted sMB, certain selected transform domain predicted sMB coefficients are directly calculated.

The novel approach of finding transform domain residue is calculated by subtracting the transform domain predicted values from transform domain original values, that is, $RES = Tx(res) = Tx(org - pred) = Tx(org) - Tx(pred) = ORG - PRED$.

Finding ORG and PRED & RES are explained here-after.

4.1 Transform Of Original

The original 4x4 sMB is core-forward transformed and its computational complexity is 64 additions and 16 shifts.

$$ORG = Cf \times org \times Cft$$

$$where\ ORG\ is\ core\ forward\ transformed\ original$$

4.2 Vertical Mode Residue

The transform domain Vertical predicted values are calculated as below.

$$VP\_Tx = Cf \times VP\_Px \times Cft = \begin{pmatrix} VP11 & VP12 & VP13 & VP14 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The four transform coefficients are calculated as follows.

$VP11 = (AD + BC) \ll 2$
$VP12 = (AD1 \ll 1 + BC1) \ll 2$
$VP13 = (AD - BC) \ll 2$
$VP14 = (AD1 - BC1 \ll 1) \ll 2$
where,
$AD = A + D$
$AD1 = A - D$
$BC = B + C$
$BC1 = B - C$

The computational complexity to find transform domain Vertical Prediction mode coefficients is 8 additions and 6 shifts. The transform domain residue for Vertical Prediction Mode is calculated by subtracting these four transform coefficients from original transform coefficients.

$$VP\_RES = \begin{cases} ORG(i,j) - VP\_Tx(i,j), & i = 1, j = 1\ to\ 4 \\ ORG(i,j), & i = 2\ to\ 4, j = 1\ to\ 4 \end{cases}$$

The computational complexity to find transform domain residue coefficients of Vertical Prediction mode is 4 additions and zero shifts.

4.3 Horizontal Mode Residue

The transform domain Horizontal predicted values are calculated as below.

$$HP\_Tx = Cf \times HP\_Px \times Cft = \begin{pmatrix} HP11 & 0 & 0 & 0 \\ HP21 & 0 & 0 & 0 \\ HP31 & 0 & 0 & 0 \\ HP41 & 0 & 0 & 0 \end{pmatrix}$$

The four transform coefficients are calculated as follows.

$HP11 = (IL + JK) \ll 2$
$HP21 = (IL1 \ll 1 + JK1) \ll 2$
$HP31 = (IL - JK) \ll 2$
$HP41 = (IL1 - JK1 \ll 1) \ll 2$
where,
$IL = I + L$
$IL1 = I - L$
$JK = J + K$
$JK1 = J - K$

The computational complexity to find transform domain Horizontal Prediction mode coefficients is 8 additions and 6 shifts. The transform domain residue for Horizontal Prediction Mode is calculated by subtracting these four transform coefficients from original transform coefficients.

$$HP\_RES = \begin{cases} ORG(i,j) - HP\_Tx(i,j), & i = 1\ to\ 4, j = 1 \\ ORG(i,j), & i = 1\ to\ 4, j = 2\ to\ 4 \end{cases}$$

The computational complexity to find transform domain residue coefficients of Horizontal Prediction mode is 4 additions and zero shifts.

4.4 DC Mode Residue

The unique and only one pixel domain predicted value is Av, where $Av = DC(1,1)$ (Refer Page 4). The transform domain coefficients are calculated as follows.

$$DC\_Tx = Cf \times DC\_Px \times Cft = \begin{pmatrix} DC11 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where, $DC11 = DC(1,1) \ll 4$

The computational complexity to find transform domain DC Prediction mode coefficients is zero additions and 1 shift. The transform domain residue for DC Prediction Mode is calculated by subtracting this one transform coefficient from original transform coefficient.

$$DC\_RES = \begin{cases} ORG(i,j) - DC11, & i = 1, j = 1 \\ ORG(i,j), & i = 1\ to\ 4, j = 1\ to\ 4 \end{cases}$$

The computational complexity to find transform domain residue coefficients of DC Prediction mode is 1 addition and zero shifts.

4.5 Diagonal-Down-Left Mode Residue

The transform domain DDL predicted values are derived as follows: $DDL\_Tx = Cf \times DDL\_Px \times Cft$

$$DDL\_Tx = \begin{pmatrix} DL11 & DL12 & DL13 & DL14 \\ DL12 & DL22 & DL23 & DL24 \\ DL13 & DL23 & DL33 & DL34 \\ DL14 & DL24 & DL34 & DL44 \end{pmatrix}$$

The ten unique transform coefficients are calculated as follows.

$DL11 = DDL\_11 + DDL\_08 + DDL\_10$
$DL12 = DDL\_12 \ll 1 + DDL\_09$
$DL13 = DDL\_00 - DDL\_02$
$DL14 = DDL\_12 - DDL\_05$
$DL22 = (DDL\_00 + DDL\_04) \ll 2 - DDL\_08 - DDL\_07$
$DL23 = (DDL\_01 - DDL\_13) \ll 1 - DDL\_05$
$DL24 = (DDL\_00 - DDL\_04) \ll 1 + DDL\_02 - DDL\_04$
$DL33 = DDL\_11 - DDL\_02 - DDL\_10$
$DL34 = DDL\_12 + DDL\_13 - DDL\_09$
$DL44 = DDL\_00 + DDL\_02 \ll 3 - DDL\_04 \ll 2 - DDL\_07$

where,

$DDL\_00 = DDL(1,1) + DDL(4,4)$
$DDL\_01 = DDL(1,1) - DDL(4,4)$
$DDL\_02 = DDL(1,3) + DDL(2,4)$
$DDL\_03 = DDL(1,3) - DDL(2,4)$
$DDL\_04 = DDL(1,2) + DDL(3,4)$
$DDL\_05 = DDL(1,2) - DDL(3,4)$
$DDL\_06 = DDL(1,4) \ll 2$
$DDL\_07 = DDL(1,4) \ll 3 + DDL(1,4) \ll 1$
$DDL\_08 = DDL\_02 \ll 1 + DDL\_02$
$DDL\_09 = DDL\_05 \ll 1 + DDL\_05$
$DDL\_10 = DDL\_04 \ll 1$
$DDL\_11 = DDL\_00 + DDL\_06$
$DDL\_12 = DDL\_01 + DDL\_03$
$DDL\_13 = DDL\_03 \ll 1$

The computational complexity to find transform domain coefficients of DDL Prediction mode is 31 additions and 13 shifts. The transform domain residue for DDL Prediction Mode is calculated by subtracting these transform coefficients from original transform coefficient.

$$DDL\_RES = ORG(i,j) - DDL\_Tx(i,j), i = 1\ to\ 4, j = 1\ to\ 4$$

The computational complexity to find transform domain residue coefficients of DDL Prediction mode is 16 additions and zero shifts.

4.6 Diagonal-Down-Right Mode Residue

The transform domain DDR predicted values are derived as follows: $DDR\_Tx = Cf \times DDR\_Px \times Cft$

$$DDR\_Tx = \begin{pmatrix} DR11 & DR12 & DR13 & DR14 \\ -DR12 & DR22 & DR23 & DR24 \\ DR13 & -DR23 & DR33 & DR34 \\ -DR14 & DR24 & -DR34 & DR44 \end{pmatrix}$$

The ten unique transform coefficients are calculated as follows.

$DDR\_Tx(1,1) = DDR\_05 + DDR\_11 + DDR\_12$
$DDR\_Tx(1,2) = -DDR\_03 - DDR\_08 \ll 1$
$DDR\_Tx(1,3) = DDR\_01 - DDR\_04$
$DDR\_Tx(1,4) = DDR\_02 - DDR\_08$
$DDR\_Tx(2,2) = DDR\_05 + DDR\_10 - DDR\_14 - DDR\_13 \ll 1$
$DDR\_Tx(2,3) = (DDR\_07 - DDR\_15) \ll 1 - DDR\_02$
$DDR\_Tx(2,4) = DDR\_00 - DDR\_04 + DDR\_11 - DDR\_13$
$DDR\_Tx(3,3) = DDR\_12 - DDR\_11 - DDR\_04$
$DDR\_Tx(3,4) = DDR\_03 - DDR\_06 - DDR\_07 - DDR\_15$
$DDR\_Tx(4,4) = DDR\_14 - DDR\_01 - DDR\_04 \ll 3 + DDR\_10$

where,

$DDR\_00 = DDL(1,1) + DDR(3,1)$
$DDR\_01 = DDL(1,2) + DDR(4,1)$
$DDR\_02 = DDL(1,1) - DDR(3,1)$
$DDR\_03 = DDR\_02 \ll 1 + DDR\_02$
$DDR\_04 = DDR(1,2) + DDR(2,1)$
$DDR\_05 = DDR\_04 \ll 1 + DDR\_04$
$DDR\_06 = DDR(1,2) - DDR(2,1)$
$DDR\_07 = DDL(1,2) - DDR(4,1)$
$DDR\_08 = DDR\_06 + DDR\_07$
$DDR\_09 = DDR(1,1) \ll 2$
$DDR\_10 = (DDR(1,1) + DDR\_09) \ll 1$
$DDR\_11 = DDR\_00 \ll 1$
$DDR\_12 = DDR\_01 + DDR\_09$
$DDR\_13 = DDR\_01 \ll 1$
$DDR\_14 = DDR\_11 \ll 1$
$DDR\_15 = DDR\_06 \ll 1$

The computational complexity to find transform domain coefficients of Diagonal-Down-Right Prediction mode is 32 additions and 12 shifts. The transform domain residue for Diagonal-Down-Right Prediction Mode is calculated by subtracting these transform coefficients from original transform coefficient.

$$DDR\_RES = ORG(i,j) - DDR\_Tx(i,j), i = 1 \; to \; 4, j = 1 \; to \; 4$$

The computational complexity to find transform domain residue coefficients of Diagonal-Down-Right Prediction mode is 16 additions and zero shifts.

4.7 Vertical-Right Mode Residue

The transform domain VR predicted values are derived as follows: $VR\_Tx = Cf \times VR\_Px \times Cft$

All the transform coefficients (VR_Tx) calculated as follows.

$VR\_Tx(1,1) = VR\_02 + VR\_05 + VR\_08$
$VR\_Tx(1,2) = VR\_13 - VR\_10 \ll 1$
$VR\_Tx(1,3) = VR\_02 - VR\_08$
$VR\_Tx(1,4) = -VR\_10 - VR\_12$
$VR\_Tx(2,1) = VR\_11 + VR\_16 + VR\_18$
$VR\_Tx(2,2) = VR\_20 - VR\_03 \ll 1 + VR\_09$
$VR\_Tx(2,3) = VR\_11 + VR\_13 - VR\_18$
$VR\_Tx(2,4) = VR\_04 - VR\_03 + (VR\_05 - VR\_09) \ll 1$
$VR\_Tx(3,1) = VR\_00$
$VR\_Tx(3,2) = (VR\_18 - VR\_14) \ll 1 + VR\_16$
$VR\_Tx(3,3) = VR\_00 + VR\_06$
$VR\_Tx(3,4) = VR\_17 - VR\_14 - VR\_18 \ll 2$
$VR\_Tx(4,1) = VR\_15 + VR\_17 + VR\_19$
$VR\_Tx(4,2) = (VR\_06 - VR\_01) \ll 1 - VR\_04 - VR\_07$
$VR\_Tx(4,3) = VR\_15 - VR\_12 - VR\_19$
$VR\_Tx(4,4) = VR\_08 - VR\_01 - VR\_20$

where,

$tp\_00 = VR(1,4) + DDR(3,1)$
$tp\_01 = VR(1,4) - DDR(3,1)$
$tp\_02 = DDR(2,1) + DDL(1,2)$
$tp\_03 = DDR(2,1) - DDL(1,2)$
$tp\_04 = VR(1,1) + DDL(1,1)$
$tp\_05 = VR(1,1) - DDL(1,1)$
$tp\_06 = VR(1,3) - DDR(1,1)$
$tp\_07 = tp\_04 \ll 1$
$VR\_00 = tp\_00 - tp\_02$
$VR\_01 = VR\_00 - tp\_02$
$VR\_02 = tp\_00 + tp\_02$
$VR\_03 = tp\_00 + VR\_02$
$VR\_04 = VR(1,3) + DDR(1,1)$
$VR\_05 = tp\_07 + VR\_04 \ll 1$
$VR\_06 = tp\_07 - VR\_04 \ll 1$
$VR\_07 = VR(1,2) + DDR(1,2)$
$VR\_08 = VR\_07 \ll 1$
$VR\_09 = VR\_08 + VR\_07$
$VR\_10 = tp\_01 - tp\_03$
$VR\_11 = tp\_01 + VR\_10$
$VR\_12 = tp\_05 - tp\_06$
$VR\_13 = VR\_12 \ll 1 + VR\_12$
$VR\_14 = tp\_01 + tp\_03$
$VR\_15 = VR\_14 + tp\_03$
$VR\_16 = tp\_05 + tp\_06$
$VR\_17 = VR\_16 \ll 1 + VR\_16$
$VR\_18 = VR(1,2) - DDR(1,2)$
$VR\_19 = VR\_18 \ll 1 + VR\_18$
$VR\_20 = tp\_04 + tp\_07$

The computational complexity to find transform domain coefficients of Vertical-Right Prediction mode is 55 additions and 12 shifts. The transform domain residue for Vertical-Right Prediction Mode is calculated by subtracting these transform coefficients from original transform coefficient.

$$VR\_RES = ORG(i,j) - VR\_Tx(i,j), i = 1 \; to \; 4, j = 1 \; to \; 4$$

The computational complexity to find transform domain residue coefficients of Vertical-Right Prediction mode is 16 additions and zero shifts.

4.8 Horizontal-Down Mode Residue

The transform domain VR predicted values are derived as follows: $HD\_Tx = Cf \times HD\_Px \times Cft$

All the transform coefficients (HD_Tx) calculated as follows.

$HD\_Tx(1,1) = HD\_12 + HD\_23 + HD\_21$
$HD\_Tx(1,2) = HD\_25 - HD\_18$
$HD\_Tx(1,3) = -HD\_13$
$HD\_Tx(1,4) = HD\_19 + HD\_25 \ll 1 + HD\_25$
$HD\_Tx(2,1) = HD\_16 \ll 1 + HD\_27$
$HD\_Tx(2,2) = HD\_20 - HD\_15 \ll 1 + HD\_22$
$HD\_Tx(2,3) = HD\_09 - HD\_17 \ll 1 + HD\_25$
$HD\_Tx(2,4) = (HD\_14 + HD\_24) \ll 1 - HD\_06 - HD\_08$
$HD\_Tx(3,1) = HD\_12 - HD\_21$
$HD\_Tx(3,2) = HD\_27 - HD\_18 - HD\_09$
$HD\_Tx(3,3) = HD\_24 - HD\_13$
$HD\_Tx(3,4) = HD\_19 - HD\_10 - HD\_09 \ll 1 - HD\_09$
$HD\_Tx(4,1) = HD\_16 - HD\_10$
$HD\_Tx(4,2) = HD\_06 - HD\_15 - (HD\_22 - HD\_23) \ll 1$
$HD\_Tx(4,3) = HD\_26 - HD\_17 - HD\_09 + HD\_26 \ll 1$
$HD\_Tx(4,4) = HD\_14 - HD\_20 + HD\_21$

where,

$HD\_00 = DDR(1,2) + DDR(4,1)$

$$
\begin{aligned}
HD\_01 &= DDR(1,2) - DDR(4,1)\\
HD\_02 &= DDL(1,1) + HD(4,1)\\
HD\_03 &= DDL(1,1) - HD(4,1)\\
HD\_04 &= HD(1,1) + DDR(3,1)\\
HD\_05 &= HD(1,1) - DDR(3,1)\\
HD\_06 &= DDR(1,1) + HD(3,1)\\
HD\_07 &= DDR(1,1) - HD(3,1)\\
HD\_08 &= HD(2,1) + DDR(2,1)\\
HD\_09 &= HD(2,1) - DDR(2,1)\\
HD\_10 &= HD\_05 + HD\_07\\
HD\_11 &= HD\_05 - HD\_07\\
HD\_12 &= HD\_00 + HD\_02\\
HD\_13 &= HD\_00 - HD\_02\\
HD\_14 &= HD\_13 + HD\_00\\
HD\_15 &= HD\_12 + HD\_02\\
HD\_16 &= HD\_01 + HD\_03\\
HD\_17 &= HD\_01 - HD\_03\\
HD\_18 &= HD\_16 + HD\_03\\
HD\_19 &= HD\_17 + HD\_01\\
HD\_20 &= HD\_04 \ll 1 + HD\_04\\
HD\_21 &= HD\_08 \ll 1\\
HD\_22 &= HD\_21 + HD\_08\\
HD\_23 &= (HD\_04 + HD\_06) \ll 1\\
HD\_24 &= (HD\_04 - HD\_06) \ll 1\\
HD\_25 &= HD\_11 + HD\_09\\
HD\_26 &= HD\_11 - HD\_09\\
HD\_27 &= HD\_10 \ll 1 + HD\_10
\end{aligned}
$$

The computational complexity to find transform domain coefficients of Horizontal-Down Prediction mode is 56 additions and 12 shifts. The transform domain residue for Horizontal-Down Prediction Mode is calculated by subtracting these transform coefficients from original transform coefficient.

$$HD\_RES = ORG(i,j) - HD\_Tx(i,j), i = 1\ to\ 4, j = 1\ to\ 4$$

The computational complexity to find transform domain residue coefficients of Horizontal-Down Prediction mode is 16 additions and zero shifts.

4.9 Vertical-Left Mode Residue

The transform domain VL predicted values are derived as follows: $VL\_Tx\ = Cf \times VL\_Px \times Cft$

All the transform coefficients (VL_Tx) calculated as follows.

$$
\begin{aligned}
VL\_Tx(1,1) &= VL\_10 + VL\_21 + VL\_22\\
VL\_Tx(1,2) &= VL\_14 \ll 1 - VL\_26\\
VL\_Tx(1,3) &= VL\_10 - VL\_21\\
VL\_Tx(1,4) &= VL\_14 + VL\_19\\
VL\_Tx(2,1) &= VL\_07 + VL\_16 + VL\_18\\
VL\_Tx(2,2) &= (VL\_12 - VL\_06) \ll 1 - VL\_06 - VL\_20\\
VL\_Tx(2,3) &= VL\_16 - VL\_07 + VL\_26\\
VL\_Tx(2,4) &= VL\_12 - VL\_04 + VL\_21 + (VL\_21 - VL\_22) \ll 1\\
VL\_Tx(3,1) &= VL\_11\\
VL\_Tx(3,2) &= (VL\_15 - VL\_07) \ll 1 - VL\_18\\
VL\_Tx(3,3) &= VL\_11 - VL\_23\\
VL\_Tx(3,4) &= VL\_07 \ll 2 + VL\_15 - VL\_25\\
VL\_Tx(4,1) &= VL\_24 + VL\_17 + VL\_25\\
VL\_Tx(4,2) &= VL\_04 + VL\_06 + (VL\_13 + VL\_23) \ll 1\\
VL\_Tx(4,3) &= VL\_17 - VL\_24 - VL\_19\\
VL\_Tx(4,4) &= VL\_13 + VL\_20 - VL\_21
\end{aligned}
$$

where

$$
\begin{aligned}
VL\_00 &= VR(1,2) + DDL(2,4)\\
VL\_01 &= VR(1,2) - DDL(2,4)\\
VL\_02 &= DDL(1,1) + VL(3,4)\\
VL\_03 &= DDL(1,1) - VL(3,4)\\
VL\_04 &= VR(1,3) + DDL(1,4)\\
VL\_05 &= VR(1,3) - DDL(1,4)\\
VL\_06 &= VR(1,4) + DDL(1,3)\\
VL\_07 &= VR(1,4) - DDL(1,3)\\
VL\_08 &= VL(1,4) + DDL(1,2)\\
VL\_09 &= VL(1,4) - DDL(1,2)\\
VL\_10 &= VL\_00 + VL\_02\\
VL\_11 &= VL\_00 - VL\_02\\
VL\_12 &= VL\_10 + VL\_00\\
VL\_13 &= VL\_11 - VL\_02\\
VL\_14 &= VL\_01 + VL\_03\\
VL\_15 &= VL\_01 - VL\_03\\
VL\_16 &= VL\_01 + VL\_14\\
VL\_17 &= VL\_15 - VL\_03\\
VL\_18 &= VL\_09 + VL\_05\\
VL\_19 &= VL\_09 - VL\_05\\
VL\_20 &= VL\_08 \ll 1 + VL\_08\\
VL\_21 &= VL\_06 \ll 1\\
VL\_22 &= (VL\_04 + VL\_08) \ll 1\\
VL\_23 &= (VL\_04 - VL\_08) \ll 1\\
VL\_24 &= VL\_07 \ll 1 + VL\_07\\
VL\_25 &= VL\_18 \ll 1 + VL\_18\\
VL\_26 &= VL\_19 \ll 1 + VL\_19
\end{aligned}
$$

The computational complexity to find transform domain coefficients of Vertical-Left Prediction mode is 56 additions and 14 shifts. The transform domain residue for Vertical-Left Prediction Mode is calculated by subtracting these transform coefficients from original transform coefficient.

$$VL\_RES = ORG(i,j) - VL\_Tx(i,j), i = 1\ to\ 4, j = 1\ to\ 4$$

The computational complexity to find transform domain residue coefficients of Vertical-Left Prediction mode is 16 additions and zero shifts.

4.10 Horizontal-Up Mode Residue

The transform domain HU predicted values are derived as follows: $HU\_Tx\ = Cf \times HU\_Px \times Cft$

All the transform coefficients (HU_Tx) calculated as follows.

$$
\begin{aligned}
HU\_Tx(1,1) &= HU\_00 + (HU\_02 + HU\_04 + HU\_09) \ll 1\\
HU\_Tx(1,2) &= HU\_06 + HU\_03 + HU\_11\\
HU\_Tx(1,3) &= HU\_01\\
HU\_Tx(1,4) &= HU\_14 + HU\_10\\
HU\_Tx(2,1) &= (HU\_00 + HU\_02 - L) \ll 1 + HU\_02 - L \ll 3\\
HU\_Tx(2,2) &= (HU\_06 - HU\_12) \ll 1 - HU\_12\\
HU\_Tx(2,3) &= (HU\_01 - HU\_05) \ll 1 - HU\_03\\
HU\_Tx(2,4) &= HU\_14 \ll 1 - HU\_03 + HU\_12\\
HU\_Tx(3,1) &= HU\_00 - HU\_08 \ll 1 - HU\_08\\
HU\_Tx(3,2) &= HU\_06 - HU\_02 \ll 1 - HU\_02 - HU\_11\\
HU\_Tx(3,3) &= HU\_01 - HU\_03 \ll 1
\end{aligned}
$$

$HU\_Tx(3,4) = HU\_07 + HU\_02 - HU\_10$
$HU\_Tx(4,1) = HU\_00 - HU\_02$
$HU\_Tx(4,2) = HU\_06 - HU\_15 \ll 2 - HU\_15 + HU\_12$
$HU\_Tx(4,3) = HU\_01 - (HU\_03 - HU\_05) \ll 2 + HU\_03$
$HU\_Tx(4,4) = HU\_07 + HU\_08 + HU\_13 \ll 1 + HU\_13$
where,
$HU\_00 = HD(2,1) + DDR(3,1)$
$HU\_01 = HD(2,1) - DDR(3,1)$
$HU\_02 = HD(3,1) + DDR(4,1)$
$HU\_03 = HD(3,1) - DDR(4,1)$
$HU\_04 = HD(4,1) + HU(2,4)$
$HU\_05 = HD(4,1) - HU(2,4)$
$HU\_06 = HD(2,1) + HU\_00$
$HU\_07 = HU\_01 - DDR(3,1)$
$HU\_08 = HU\_04 - L$
$HU\_09 = L \ll 1 + L$
$HU\_10 = HU\_05 \ll 1 + HU\_05 + L$
$HU\_11 = HU\_05 - HU\_09$
$HU\_12 = DDR(4,1) + HU\_08$
$HU\_13 = DDR(4,1) - HU\_08$
$HU\_14 = HU\_07 + HU\_03 \ll 1 + HU\_03$
$HU\_15 = HU\_02 - HU\_08$

The computational complexity to find transform domain coefficients of Horizontal-Up Prediction mode is 51 additions and 16 shifts. The transform domain residue for Horizontal-Up Prediction Mode is calculated by subtracting these transform coefficients from original transform coefficient.

$$HU\_RES = ORG(i,j) - HU\_Tx(i,j), i = 1\ to\ 4, j = 1\ to\ 4$$

The computational complexity to find transform domain residue coefficients of Horizontal-Up Prediction mode is 16 additions and zero shifts.

5. COMPARISON AND INFERENCES OF THE PROPOSED METHOD

The computational complexity of reference software and the proposed method are compared here. The computational complexity of Pixel domain Intra 4x4 Prediction for all the nine prediction modes in the reference software [12] and [13] are tabulated in the table 2.

The reference software does the pixel domain prediction first, and then finds the residue and last it does the forward transform for all nine modes. In that order, the computational complexity of the existing reference software is listed in the table 3.

The proposed method calculates the selective pixel domain values, the selective transform domain coefficients and finds the transform domain residue coefficients. The computational complexity of the proposed method is listed in table 4.

Table 2 Complexity of Pixel domain Intra 4x4 prediction in reference software

Prediction Type	Reference Software	
	Addition	Shifts
Vertical	0	0
Horizontal	0	0
DC	8	1
Diagonal-Down-Left	21	14
Diagonal-Down-Right	21	14
Vertical-Right	26	16
Horizontal-Down	26	16
Vertical-Left	25	15
Horizontal-Up	15	9
Total	142	85

Table 3 Complexity of Reference software during Intra 4x4 prediction

Process	Reference Software	
	Addition	Shift
Pixel domain Prediction	142	85
Finding Residue	144	0
Forward Transform	576	144
Total	862	229

Table 4 Complexity of Proposed method during Intra 4x4 prediction

Process	Proposed Method	
	Addition	Shift
Pixel domain Prediction	42	26
Finding PRED		
VP Mode	8	6
HP Mode	8	6
DC Mode	0	1
DDL Mode	31	13
DDR Mode	32	12
VR Mode	55	12
HD Mode	56	12
VL Mode	56	14
HU Mode	51	16
Finding RES		
VP Mode	4	0
HP Mode	4	0
DC Mode	1	0
DDL Mode	16	0
DDR Mode	16	0
VR Mode	16	0
HD Mode	16	0
VL Mode	16	0
HU Mode	16	0
Total	508	134

The transform domain residue sMBs for all the nine modes are available by 508 additions and 134 shifts only, whereas the existing reference software can produce the same by 862 additions and 229 shifts. After finalizing mode, the corresponding pixel domain prediction values can be generated just by copying from available 24 unique pixel values.

6. CONCLUSION

In H.264 Encoder, the computational complexity of Intra4x4 mode decision by RD-Cost method is higher than any other methods. But RD-Cost method gives best results. Especially, the process of predicting the intra 4x4 prediction for all the nine modes, finding the residue and forward transforming will take more complexity in terms of addition and shifts. Every MB in the frame of the sequence is subjected to do this process repeatedly for all its sixteen sMBs. This paper proposed a method to find transform domain residue values directly with reduced computational complexity. The proposed method used (i) the unique equations for prediction, (ii) the additive property of core forward transform and (iii) the spatial redundancy of pixel domain predicted values. Thus the proposed method reduced the computational complexity into 59% addition and 59% shift operations, thus saved 41% computational complexity. But the proposed method did not compromise the accuracy of the results, thus maintaining the same quality and bitrate. The same method can be extended to Intra 16x16 Prediction, Intra 8x8 Prediction and Intra Chroma Predictions also in H.264 Encoder.

ACKNOWLEDGMENT

The author thanks RiverSilica Technologies Private Limited, INDIA for their encouragement and support.

REFRENCES:

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology.*, Vol. 13, No. 7, pp. 560–576, July 2003.

[2] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G. J. Sullivan, "Performance comparison of video standards using Lagrangian control," in *Proc. IEEE Int. Conf. Image Process.*, pp. 501–504, 2002.

[3] W. Zouch, A. Samet, M. A. Ben Ayed, F Kossentini, N. Masmoudi, "Complexity Analysis of Intra Prediction in H.264/AVC", *16th International Conference on Micro Electronics Proceedings,* on December 6-8, 2004.

[4] Mustafa Parlak, Yusuf Adibelli, and Ilker Hamzaoglu, "A Novel Computational Complexity and Power Reduction Technique for H.264 Intra Prediction", *IEEE Transactions on Consumer Electronics*, pp. 2006 – 2014, Vol. 54, No. 4, November 2008.

[5] Y. Adibelli, M. Parlak, and I. Hamzaoglu, "A Computation and Power Reduction Technique for H.264 Intra Prediction", *13th Euromicro Conference on Digital System Design: Architecture, Methods and Tools*, pp. 753 – 760, 2010.

[6] E. Sahin, I. Hamzaoglu, "An Efficient Hardware Architecture for H.264 Intra Prediction Algorithm", *Design, Automation & Test in Europe Conference & Exhibition,* 16-20 April 2007.

[7] Y. Lai, T. Liu, Y. Li, C. Lee, "Design of An Intra Predictor with Data Reuse for High-Profile H.264 Applications", *IEEE ISCAS*, pp. 3018 – 3021, May 2009.

[8] F. Pan, X. Lin, S. Rahardja, K. Lim, Z. Li, S. Dajun Wu, "Fast Mode Decision Algorithm for Intra Prediction in H.264/AVC Video Coding", *IEEE Transactions on CAS for Video Technology,* Vol. 15, No. 7., pp 813 – 822, July 2005.

[9] I. Choi, J. Lee, B. Jeon, "Fast Coding Mode Selection With Rate-Distortion Optimization for MPEG-4 Part 10 AVC/H.264", *IEEE Transactions on CAS for Video Technology,* Vol. 16, No. 12, pp 1557 – 1561, December 2006.

[10] A. Elyousfi, A. A. Tamtaoui, and E. Bouyakhf, "A New Fast Intra Prediction Mode Decision Algorithm for H.264/AVC Encoders", *World Academy of Science, Engineering and Technology,* 2007.

[11] Iain E. Richardson, "The H.264 – Advanced Video Compression Standard" – Second Edition – A John Wiley and Sons, Ltd., Publication, 2010.

[12] Joint Video Team, JM Reference Software V17.1 - iphome.hhi.de/suehring/tml/.

[13] x264 Encoder Open Source – http://www.videolan.org/developers/x264.html - May, 2012.