Compressed Domain H.264 Baseline Encoder in Video Transcoding Architecture

P. Essaki Muthu [#1]

[#] Research Scholar, Dept of ECE, Dr. MGR Educational and Research Institute University, Chennai, Tamil Nadu, INDIA

[1] pessakimuthu@yahoo.com

Abstract – With the growth of technology, there is an increase in the number of networks, types of devices and different content representation formats as a result of which interoperability between different systems and networks is gaining in importance. Video transcoding is the process of converting video from one format to another. H.264/AVC, developed by the Joint Video Team (JVT), is new standard which fulfils significant coding efficiency, simple syntax specifications and seamless integration of video coding into all current protocols and multiple architectures. The research work presented in this paper is carrying out compressed domain video encoding through integer transform in compliance with H.264 Standard in the transcoding pipeline. The core forward transform or H.264 Transform (HT) and inverse transform are analysed and adopted in this paper. The complete encoder has been demonstrated with the help of three different types of input video sequences. This paper analysed different metrics/parameters involved in encoding. This research work proposed the method of computing the complexity. It has also been proposed that the combination of Quality, File size and Complexity shall be used as one of the important metrics to evaluate the video processing.

Keyword – Compressed domain encoding, H.264 Encoder, Video Transcoder, Baseline profile, integer transform operations

I. INTRODUCTION

In this fast growing world of multimedia and telecommunications, there is a great demand for efficient usage of the available bandwidth [1 – 3]. Transcoding of video content is one such effort in this direction. A format is basically defined by the characteristics such as bitrate, frame rate, spatial resolution etc. Ishfaq, A. et al. [4] have summarized various techniques and research issues associated with Video transcoding. They have detailed various research issues arising in transcoding and illustrated them using an architectural approach. Hari Kalva's papers [5 – 16] specified that the main goals of an efficient transcoder are, 1) To maintain the quality of the transcoded bitstream to the one obtained by direct decoding and re-encoding of the input stream, 2) To reuse the information available in the bitstream as much as possible so as to avoid multigenerational deterioration and 3) The process should be efficient, low in complexity and achieve the highest quality possible.

The H.264 Video Standard is substantially different from previous MPEG and ITU standards [17]. The syntax and the algorithms used in H.264 are so different that transcoding a video compressed by traditional DCT-based standards to H.264 will face many difficulties, especially to perform transcoding in the compressed domain. Gao, C. et al. [18] performed conversion of transforms between MPEG-2 and H.264. Fast DCT to Integer Transform (IT) conversion improved the computational efficiency. The DCT operation in H.264 is split into two parts, termed core forward transform and scaling matrix. The scaling matrix is combined with quantization in encoding process and the inverse scaling matrix with de-quantization in H.264. The combined matrices are scaled and truncated into integer numbers to perform integer operations. This integer transform process cannot be adopted for transcoding, because it is combined with quantization which leads to data loss.

However the precision is considered, because of truncation of data, the transform domain prediction will not be equivalent to pixel domain prediction. So there is a need of integer transform which does not involve with the precision. At the same time, the transform should be reversible, so that it can switch between the pixel domain and transform domain without any data loss. So far, according to literature survey and review, the integer transform based compressed domain transcoding (intra prediction, motion compensation, mode decision and motion estimation) has not been reported.

In compressed domain transcoding, the coefficients in DCT and IDCT, (which were explained by Pedro, A. et al. [19]), were taken as truncated float numbers. So, the compressed (transform) domain intra prediction and motion compensation in the decoder cause data loss with respect to pixel domain operations. In the same way, the mode decision and motion estimation were performed with approximation of compressed domain data. They were not as equivalent as Pixel domain mode decision and motion estimation results explained by Pedro, A. et al. [19]. The integer transform [20] used in H.264 Standard was the derivative of DCT/IDCT. He explained the transform and quantization done in H.264 Standard. The core forward transform or H.264 Transform (HT) and

inverse transform are analyzed and adopted in this paper. Based on this transform, all the processes involved in encoding are designed.

This paper is organized as follows. The encoder architecture with its prediction techniques, motion estimation, mode decision and coding are explained in Section II. Experimentation of transform domain encoder is detailed in Section III. Section IV lists the results and result analysis. Section V concludes the findings of encoder with further scopes.

II. ENCODER ARCHITECTURE

The compressed domain input video frame is encoded by compressed domain H.264 encoder in this chapter. The processes like Intra prediction, Motion Compensation, Intra mode Decision and Motion Estimation are designed to perform in compressed domain. The architecture of compressed domain H.264 Baseline encoder (shown in Fig. 1) uses CAVLC coding with single reference frame. The original compressed domain video frame is predicted by compressed domain prediction engines. The best type is selected based on RD-Cost of the different types. The metrics used in the encoder are SAD, SATD and RD-Cost.

Gerardo, F. et al. [21] explained that DC values of all Intra4x4 and Intra16x16 modes were calculated and compared to find best mode - less computation. Here, the intra prediction performed all possible modes of Intra 16x16 and Intra 4x4, because the highest complexity comes out when all modes are performed. In Intra 16x16, the best mode among Intra 16x16 modes is selected based on SATD based cost calculation. In Intra 4x4, the best mode for each Sub-Macroblock (4x4) is decided based on RD-Cost. The RD-Cost under Intra 16x16 is also calculated. The best among Intra 16x16 and Intra 4x4 is decided by minimum RD-Cost. The best intra Chroma mode is selected based on SATD based cost calculation on all possible Chroma modes.

The motion vector is estimated for a given block size in two steps: Integer level and Sub-pel level. In the integer level motion vector estimation, the best match is searched in a defined search area thoroughly. The search technique used here is called fast full search technique. The cost of each match is calculated as follows: cost = SAD + λ x (number of bits for MVD). The lowest cost decided the best match at the integer level, where λ is Lagrangian Multiplier. In Sub-pel level motion estimation, there are two steps involved: half-pel motion estimation and qual-pel motion estimation. In half-pel, the intermediate pixel values are calculated and compared with the original to find the best match. Out-of-9 possibilities of combination in half-pel motion estimation, the best match is found out by minimum cost. With respect to the best match position, the qual-pel motion estimation is done as same way as half-pel motion estimation. The best match out-of-9 possibilities, is selected based on minimum cost.

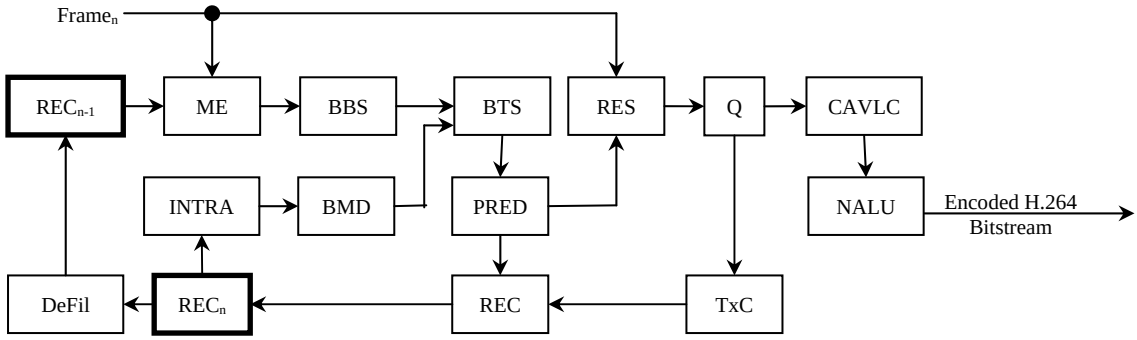

Fig. 1 Architecture of compressed domain H.264 Baseline encoder

Frame$_n$: Compressed domain Input Frame

REC$_{n-1}$: Compressed domain Previous Reconstructed Frame

ME: Compressed domain Motion Estimation

BBS: Best Block Selection

INTRA: Compressed domain Intra Mode Decision

BMD: Best Mode Decision

BTS: Best type Selection

PRED: Predicted block

REC: Reconstruction with drift manager

REC$_n$: Compressed domain Present Reconstructed Frame

DeFil: Compressed domain Deblocking loop filter

RES: Residue Calculation

Q: Quantization

TxC: Transform converter

CAVLC: Context Adaptive Variable Length Coding

NALU: Network Abstraction Layer Unit

The MVD is calculated as the difference between the actual motion vector and MVP. The prediction of motion vector for different block sizes is defined by H.264 Standard and the same is adopted here. The motion vector generally includes the sub-pel motion information. The motion vectors are estimated for all possible block sizes of a given Macroblock. The RD-Cost of PSKIP, P16x16, P16x8, P8x16 and P8x8 are calculated. P8x8 has the best block size among P8x8, P8x4, P4x8 and P4x4. The minimum RD-Cost among all these block sizes decided the best Inter block. The minimum most RD-Cost among Intra and Inter decided the best type. The compressed domain residue is directly obtained by subtracting predicted from original values, because of compressed domain operations. The residues are directly quantized by simply multiplying the scaling matrices based on the QP (0 to 51, defined by H.264 Standard). Qiang, T. et al. [22] explained the quantization in compliance with H.264 Standard.

The quantized residue coefficients are coded by CAVLC and its syntax elements are coded by Exp-Golomb coder. The coded bitstream is packed in NAL Unit. The encoder has decoder path also to imitate the receiver side operations, thus adding the receiver side distortion into account. The loop filter is applied on the present reconstructed frame so as to keep it as reference for future. The generated transcoded bitstream is comparable with the standard reference software, like JM and x264 in terms of complexity. Each Macroblock from the compressed domain resized frame is compressed by the best possible method, either Intra mode or Inter mode. The decision of Macroblock is very complex and mandatory in the encoder. Each Macroblock is encoded as H.264 bitstream at last. The steps involved in generation of H.264 bitstream are, 1) Compressed Domain Intra Mode Decision, 2) Compressed Domain Inter Mode Decision, 3) Mode Decision for a Macroblock, 4) Compressed Domain Reconstruction, and 5) Entropy Encoding and Generation of H.264 bitstream.

A. Compressed Domain Intra Prediction

Mode decision [23] is not specified in H.264/AVC standard. It is left as an encoder issue and is the most important step at the encoder side because it affects the coding performance most. Chia-Wei, T. et al. [24] performed a partial cost intra prediction algorithm based on transform-domain property using HADAMARD Transform. Inchoon, C. et al. [25] explained early SKIP and fast mode decision by selective intra coding. Jun, X. et al. [26] performed SATD based Mode decisions and Branko, P. et al. [14] used SAD based mode decision between Intra 16x16 and Intra 4x4. Kan, C. et al. [27] demonstrated SAD based intra mode decision for H.264. In Mohammed, G, S. et al. [28]'s algorithm, the enhanced cost function used sum of absolute HADAMARD-transformed differences (SATD) and mean absolute deviation of the residual block to estimate distortion part of the cost function. A threshold based large coefficients count was also used for estimating the bitrate part. Chao-Hsuing, T. et al. [29] used SAITD based mode decision. This research work performed the same prediction mechanism, but in the most efficient way to reduce the computational complexity.

For any given Macroblock, there are two possible types of Intra modes, namely Intra 16x16 and Intra 4x4. The best mode among these two types is decided based on $rdcost$. This process gives not only the best mode, but the reconstructed Macroblock which is required in the reconstruction path. The results of the process are listed below.

- Best Luma Intra 16x16 Mode, Best Luma Intra 4x4 Modes and Best Chroma Mode
- Distortions in terms of $TxSSEY_{I16\times16}$, $TxSSEY_{I4\times4}$ and $TxSSEC_I$
- Rate in terms of $rateY_{I16\times16}$ (Total number of bits required to code Residue coefficients under Intra 16x16 type), $rateY_{I4\times4}$ (Total number of bits required to code Residue coefficients under Intra 4x4 type, Intra 4x4 mode deviations with respect to most probable mode) and $rateC_I$ (Total number of bits required to code Chroma Residue coefficients and its mode)
- Code block patterns for Luma and Chroma Residue coefficients under I16x16 and I4x4, represented as $cbpY_{I16\times16}$, $cbpY_{I4\times4}$ and $cbpC_{IC}$
- Predicted and Reconstructed Macroblock

In compressed domain, the reconstructed frame is available in transform format, i.e., 4x4 core forward transformed. The pixel values are obtained if inverse transform is applied on each 4x4 sub-Macroblock. But this chapter explains the compressed domain intra prediction with reference to the compressed domain reconstructed frame. Here the predicted values are calculated in compressed domain. This process is performed commonly by compressed domain decoder and compressed domain encoder. There are two different types of Luma Intra prediction, namely Intra 4x4 and Intra 16x16. For Chroma, it is called Chroma Intra 8x8 prediction.

1. Compressed Domain Intra 4x4 Prediction: The compressed domain top 4x4 sub-Macroblock (TOP), left 4x4 sub-Macroblock (LEFT), topright 4x4 sub-Macroblock (TOPRIGHT) and topleft 4x4 sub-Macroblock (TOPLEFT) are used for compressed domain intra 4x4 prediction (PRED). The way of finding the compressed domain prediction is given below.

1. All the neighbourhood sub-Macroblocks are inverse transformed.
2. The appropriate matrices are applied on the pixel domain sub-Macroblocks to get pixel domain predicted values.
3. The pixel domain predicted values are transformed to compressed domain by core forward transform.

But instead of performing the above-said operations, all the operations are combined and modified in [30].

For example, Vertical Prediction is performed as follows.

1. The pixel values of top sub-Macroblocks are found out from compressed domain TOP sub-Macroblock.

$$top = Cf^{-1} \times TOP \times Cf \quad (1)$$

2. The pixel domain vertical predicted values are calculated.

$$vert = A \times top \quad (2)$$

where, $A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

3. The compressed domain vertical predicted values are obtained as follows.

$$VERT = Cf \times vert \times Cf^{T} \quad (3)$$

4. So, as a whole, compressed domain vertical prediction is performed by

$$VERT = Cf \times A \times Cf^{-1} \times TOP \times Cf \times Cf^{T} \quad (4)$$

Like this, the equations for performing compressed domain Intra 4x4 predictions for all modes are derived. The whole equations are simplified to avoid redundant operations and matrix multiplications. The compressed domain predictions of all intra 4x4 modes are combined and simplified to three steps. These three steps bring the compressed domain predicted values of all the intra 4x4 modes with less complexity. Those three steps to get compressed domain predicted values / coefficients are 1) Finding Root coefficients, 2) Forming basic 24 components and 3) Calculating prediction values or coefficients.

2. Compressed Domain Intra 16x16 Prediction: There are four compressed domain top 4x4 sub-Macroblocks, four compressed domain left 4x4 sub-Macroblocks, and the compressed domain topleft 4x4 sub-Macroblock are used for compressed domain intra 16x16 prediction. Like Intra 4x4, the entire operations are simplified to two steps. Those two steps to get compressed domain predicted values / coefficients are 1) finding Root coefficients and 2) Calculating prediction values or coefficients and Clipping the drift

3. Compressed Domain Chroma Intra 8x8 Prediction: There are two compressed domain top 4x4 sub-Macroblocks, two left 4x4 sub-Macroblocks, and the topleft 4x4 sub-Macroblock are used for compressed domain Chroma intra 8x8 prediction. There are same two steps as Intra 16x16 to get compressed domain predicted values / coefficients, 1) finding Root coefficients and 2) calculating prediction values or coefficients.

B. Compressed Domain Intra Mode Decision

There are two types of I-Macroblock, namely Intra 4x4 and Intra 16x16. The type of I-Macroblock is decided between Intra 4x4 and Intra 16x16. Firstly, the mode decision for each 4x4 block in a Macroblock under Intra 4x4 type is explained here. Secondly, the mode decision under Intra 16x16 type is explained. Thirdly, the best mode decision for Chroma blocks is explained. At last, the best mode decision among Intra 4x4 and Intra 16x16 for a Macroblock is explained. There are sixteen 4x4 blocks in a Macroblock. Each 4x4 block is represented by intra 4x4 modes. There are nine intra 4x4 modes, namely, Vertical (V), Horizontal (H), DC, Diagonal Down Left (DDL), Diagonal Down Right (DDR), Vertical Right (VR), Horizontal Down (HD), Vertical Left (VL) and Horizontal Up (HU). The best mode is decided for each 4x4 block one-by-one.

The compressed domain predicted blocks for possible modes of a 4x4 block are calculated. The only possible mode on first row and first column is DC mode. The possible modes on first row of blocks are DC, H, and HU modes. The possible modes on first column of blocks are DC, V, DDL and VL modes. In other places, all the nine modes are possible. Compressed domain residue blocks are calculated by subtracting the compressed domain predicted block from compressed domain original block. Those residues are quantized aided by QP. The number of bits required to code those quantized residue coefficients and number of bits required to represent the intra 4x4 mode are estimated for each mode of that 4x4 block. The sum of all these estimated bits is considered as '$rate$' for each mode of the 4x4 block.

The quantized residue coefficients are transformed to compressed domain reconstructed residue coefficients. The compressed domain reconstructed residue coefficients are added with compressed domain predicted coefficients to yield compressed domain reconstructed coefficients. This is done for all the modes of that 4x4 block. The distortions (TxSSE), for each mode, between the compressed domain reconstructed block and compressed domain original block are calculated.

$$TxSSE = \sum_{row=0}^{3}\sum_{col=0}^{3}\left(\left[\frac{h(row,col)}{2^{22}}\right]\right)^2 \quad (5)$$

where,

$h = S \times (ORG - REC) \times S^T$

$$S = \begin{pmatrix} 512 & 409 & 512 & 204 \\ 512 & 204 & -512 & -410 \\ 512 & -205 & -512 & 409 \\ 512 & -410 & 512 & -205 \end{pmatrix}$$

ORG is the compressed domain original 4x4 block

REC is the compressed domain reconstructed 4x4 block

S^T is the transpose of S.

Now for each mode, rate-distortion cost is calculated by following equation.

$$rdcost_{mode} = TxSSE_{mode} \ll 18 + \lambda_1(QP) \times rate_{mode},\ mode = 0\ \ldots\ 8 \quad (6)$$

$\lambda_1(QP)$ is the Lagrangian Multiplier with respect to QP. The mode which has the minimum $rdcost$ among all modes is selected as the best intra 4x4 mode. The finalized cost is $rdcost_{block}$. The corresponding compressed domain reconstructed 4x4 block is appropriately stored in the reconstructed frame. The same process is repeated for all sixteen 4x4 blocks.

At the end of sixteen blocks, $rate_{syntax}$, the number of bits required to code the Macroblock type, coded block pattern and delta QP, is calculated. The cost of Luma intra 4x4 Macroblock type is calculated as follows.

$$rdcost_{i4} = \left(\sum_{block=0}^{15} rdcost_{block}\right) + \lambda_1(QP) \times rate_{syntax} \quad (7)$$

The cost for Luma Intra 16x16 Macroblock type is calculated as follows.

$$rdcost_{i16} = TxSSE \ll 18 + \lambda_1(QP) \times rate \quad (8)$$

where,

$$TxSSE = \sum_{block=0}^{15}\sum_{row=0}^{3}\sum_{col=0}^{3}\left(\left[\frac{h(row,col)}{2^{22}}\right]\right)^2$$

$h = S \times (ORG_{block} - REC_{block}) \times S^T$

ORG_{block} is the compressed domain original 4x4 block

REC_{block} is the compressed domain reconstructed 4x4 block

$$TxSSE = TxSSE_{Cb} + TxSSE_{Cr} \quad (9)$$

where,

$$TxSSE_{Cb} = \sum_{block=0}^{3} TxSSE_{Cb,block} \qquad TxSSE_{Cb,block} = \sum_{row=0}^{3}\sum_{col=0}^{3}\left(\left[\frac{hCb(row,col)}{2^{22}}\right]\right)^2$$

$hCb = S \times (ORGCb_{block} - RECCb_{block}) \times S^T$

$ORGCb_{block}$ is the compressed domain original 4x4 Cb block

$RECCb_{block}$ is the compressed domain reconstructed 4x4 Cb block

And the same equations are applicable to Cr block.

The cost for Chroma Intra 8x8 Macroblock type is calculated as follows.

$$rdcost_c = TxSSE \ll 18 + \lambda_1(QP) \times rate \quad (10)$$

The Macroblock type is decided based on the cost calculated for each type of Macroblock (Intra 4x4 or Intra 16x16).

The final cost under Intra 4x4 is, $cost_{i4} = rdcost_{i4} + rdcost_c$ (11)

The final cost under Intra 16x16 is, $cost_{i16} = rdcost_{i16} + rdcost_c$ (12)

The Macroblock type which has the minimum cost among these two costs is declared as the Macroblock type for a given I-Macroblock.

C. Compressed Domain Motion Compensation

The motion compensation process is to predict the block values from the reference frame (previously reconstructed frame) with the help of motion vectors. Motion vector is the displacement of the best predicted block in the reference frame with the current location of the present frame. The motion vector has two levels of motion, namely integer level and sub-pel level. Motion vector is four times of Integer level motion vector plus the sub-pel level motion vector. If mvx is the x-direction motion vector and mvy is the y-direction motion vector, then the motion vector is $mv = (mvx, mvy)$.

There are different block sizes defined in H.264 Standard, namely 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 blocks. They are defined for each Macroblock while decoded from the H.264 bitstream. The motion vector for each block size is used to calculate the predicted block from the reference frame. The location from which the block is predicted from reference frame is calculated as follows.

$$(refx, refy) = (currx, curry) \ll 2 + (mvx, mvy) \quad (13)$$

where,

$(refx, refy)$ is the location in the reference frame from where the block is extracted

$(currx, curry)$ is the location of current block in the current frame

(mvx, mvy) is the motion vector calculated while decoding the H.264 bitstream

First, for each block, the reference block is extracted from reference frame using the integer level motion vector. Based on sub-pel motion vector information, one or two filters are applied on the extracted reference block to obtain the predicted block. Getting the predicted block from reference frame using the motion vector is called motion compensation. This is done for Luma and Chroma components separately using the same motion vector. The steps followed in the research work for compressed domain motion compensation are 1) Reference Block Extraction 2) Sub-pel level Luma Motion Compensation, 3) Compressed Domain Padding, 4) Chroma Motion Compensation, and 5) Motion Vector Prediction.

D. Compressed Domain Motion Estimation

Bharanitharan, K. et al. [31] proposed a low complexity fast mode decision algorithm for H.264/AVC intra prediction that uses discrete cross differences (DCD) to reduce the unlikely candidate modes in the RDO calculation. Gerardo, F. et al. [12, 13] presented Macroblock mode decision algorithm for inter frame prediction based on data mining techniques to be used as part of a very low complexity heterogeneous video transcoder.

For a given Macroblock, the best inter block type is decided in the compressed domain motion estimation process which is explained here. The results of motion estimation process are listed below.

1. Best Block Type, i.e., Macroblock Type
2. Sub-Macroblock Type
3. Coded Block Pattern for Luma and Chroma, $cbpY_P$ and $cbpC_P$
4. Motion Vectors and Motion Vector Differences
5. Distortions in terms of $TxSSEY_P$ and $TxSSEC_P$
6. Number of bits $rate_P$ required to code Luma and Chroma Residue coefficients, MVDs, Macroblock type, and Sub-Macroblock types (if any)
7. Predicted and Reconstructed Macroblock

Finding the best match for a given Macroblock in the reference frame is called Motion Estimation. The motion is quantified by motion vector, and the difference in the values. Motion vector is the displacement of current block in the reference frame in terms of location deviation. The difference between the original current block values and the best matched block values is encoded in H.264. This compressed domain motion estimation explains, 1) finding the best match for different block sizes, 2) finding the best block size, and 3) doing motion compensation for decided block size.

There are five different types of Macroblock representation under P-Macroblock, namely, P16x16, P16x8, P8x16, P8x8 and PSKIP. The P8x8 type again has sub-types, called P8x8, P8x4, P4x8 and P4x4. The type directly says about its block size. In P16x16 type, the block size is 16x16 (Width x Height), the Macroblock size. The motion estimation finds the best match of block size 16x16 in the reference frame and gives only one motion vector. In P16x8 type, the block size is 16x8, and there are two blocks in a Macroblock, namely top 16x8 and bottom 16x8. Each block will be represented by a motion vector that motion estimation derives with its best match. There are two motion vectors in this P16x8 type. In P8x16 type, the block size is 8x16, and there are two blocks in a Macroblock, namely left 8x16 and right 8x16. Each block will be represented by a motion vector that motion estimation derives with its best match. There are two motion vectors in this P8x16 type.

In P8x8 type, the Macroblock is partitioned to four blocks with size of 8x8. Again further partitioning each 8x8 block, there are four block types for each 8x8 sub-Macroblock. They are P8x8, P8x4, P4x8 and P4x4. If the partition is P8x8, then block size is 8x8. There will be only one block in 8x8 sub-Macroblock and so only one

motion vector for this block is decided. Under P8x8, there will be minimum four motion vectors to maximum of 16 motion vectors. When the partition is P8x4, then bock size is 8x4. There will be two blocks in 8x8 sub-Macroblock, namely top 8x4 and bottom 8x4 and there are two motion vectors representing each block. When the partition is P4x8, then bock size is 4x8. There will be two blocks in 8x8 sub-Macroblock, namely left 4x8 and right 4x8 and there are two motion vectors representing each block. When the partition is P4x4, then bock size is 4x4. There will be four blocks in 8x8 sub-Macroblock, namely topleft 4x4, topright 4x4, bottomleft 4x4 and bottomright 4x4 and there are four motion vectors representing each block.

In PSKIP type, the block size under this type is 16x16. But this Macroblock is skipped with MVP and assuming the difference is zero (It means that there is the equal match available in the reference frame). The motion vector will be derived by the decoder as part of H.264 Standard, so it need not be coded. Motion estimation process is to find the best match of a given block in the reference frame. It is very easy to slide pixel-by-pixel to find the best match, when the reference frame and given block are in spatial domain. In compressed domain, both the reference frame and given block are in transform domain. So sliding technique is not possible in compressed domain. In order to perform the sliding technique of motion estimation, both the reference frame and block are transformed to pseudo domain (it is neither pixel domain nor transform domain). And then the best match, its motion vector, and its cost under each block type are calculated.

The steps involved in motion estimation are given below.

1. Pre-processing the original input Macroblock - Pre-processing is the process of transforming the compressed domain 4x4 coefficients to pseudo domain 4x4 coefficients. This pre-processing is applied on each 4x4 block of a Macroblock.
2. Predicting motion vector under PSKIP type, under P16x16 type and its integer offset vectors
3. Finding Search Area and creating energy tables
4. Motion Vector estimation for P16x16 type, P16x8 type, P8x16 type, P8x8 type
5. Motion compensation of all types and finding its $rdcost$
6. Finding the best block type among PSKIP, P16x16, P16x8, P8x16 and P8x8

E. Compressed Domain Mode Decision

The best mode (Intra or Inter) is decided for a Macroblock as follows.

Coded block patterns under Intra 4x4 is calculated.

$$cbp_{I4\times4} = cbpY_{I4\times4} + (cbpC_I \ll 4) \tag{14}$$

The number of bits required to code $cbp_{I4\times4}$ is calculated [32], represented as $ratecbp_{I4\times4}$. If the coded block pattern is not zero, then one bit ($ratedeltaQP_{I4\times4}$) is required to mention the delta QP (variation of QP of present Macroblock against the previous Macroblock). But $ratedeltaQP_{I16\times16}$ = 1 always.

The rate for Macroblock Type is calculated as follows.

$$mbtype_{I16\times16} = \begin{cases} 13 + bestmode_{I16\times16} + cbpC_I \ll 2, & cbpY_{I16\times16} = 15 \\ 1 + bestmode_{I16\times16} + cbpC_I \ll 2, & else \end{cases} \tag{15}$$

$$mbtype_{I16\times16} = \begin{cases} mbtype_{I16\times16}, & I-frame \\ 5 + mbtype_{I16\times16}, & P-frame \end{cases} \tag{16}$$

$$mbtype_{I4\times4} = \begin{cases} 1, & I-frame \\ 5, & P-frame \end{cases} \tag{17}$$

$ratembtype_{I4\times4}$ and $ratembtype_{I16\times16}$ are calculated as shown below.

$$ratembtype = 2 \times \left\lfloor log_2\left(\left[\frac{mb\_type}{2}\right] + 1\right)\right\rfloor + 1 \tag{18}$$

Now rate for Intra 16x16 and I4x4 are calculated.

$$rate_{I4} = rateY_{I4\times4} + rateC_{I4\times4} + ratembtype_{I4\times4} + cbp_{I4\times4} + ratedeltaQP_{I4\times4} \tag{19}$$

$$rate_{I16} = rateY_{I16\times16} + rateC_{I16\times16} + ratembtype_{I16\times16} + ratedeltaQP_{I16\times16} \tag{20}$$

Now RD-Costs for Intra 16x16, Intra 4x4 and Inter mode types are calculated.

$$rdcost_{I4} = (TxSSEY_{I4\times4} + TxSSEC_{I4\times4}) \ll 18 + \lambda_1 \times rate_{I4} \tag{21}$$

$$rdcost_{I16} = (TxSSEY_{I16\times16} + TxSSEC_{I16\times16}) \ll 18 + \lambda_1 \times rate_{I16} \tag{22}$$

$$rdcost_P = (TxSSEY_P + TxSSEC_P) \ll 18 + \lambda_1 \times rate_P \tag{23}$$

The mode which has **minimum $rdcost$** is selected as best mode of the given Macroblock. Based on the decided mode, the syntax elements (all the related parameters) are updated for further reference.

F. Quantization

The quantization process is defined in H.264 Standard [33]. The quantizer is merged with the scaling matrix from DCT to reduce rounding loss. The quantization is performed as,

$$QX(y,x) = sign\big(X(y,x)\big) \cdot \left\lfloor \frac{|X(y,x)| \cdot MF(y,x) + Offset}{2^{15+qpper}} \right\rfloor,\ y = 0 \dots 3, x = 0 \dots 3 \quad (24)$$

where, QX is the quantized values, $sign(X)$ is the sign of transformed values. +1 for positive and -1 for negative, $\cdot$ is the dot product and $|X|$ is the absolute values of X

$MF = \begin{pmatrix} a & b & a & b \\ b & c & b & c \\ a & b & a & b \\ b & c & b & c \end{pmatrix}$ is Scaling matrix combined with quantization. It has three unique values which are spread in a 4x4 matrix, shown in Table I (Equn 8-427 of [33]). $Offset$ is the offset value added for possible rounding

$$Offset = \begin{cases} 342 \ll (4 + qpper), & P - frame \\ 682 \ll (4 + qpper), & I - frame \end{cases} \quad (25)$$

$$qpper = \left\lfloor \frac{QP}{6} \right\rfloor \ where\ QP\ is\ the\ quantization\ parameter$$

TABLE I MULTIPLICATION FACTOR TABLE

mod(QP)	**a**	**b**	**c**	**mod(QP)**	**a**	**b**	**c**	**mod(QP)**	**a**	**b**	**c**
0	13107	8066	5243	2	10082	6554	4194	4	8192	5243	3355
1	11916	7490	4660	3	9362	5825	3647	5	7282	4559	2893

G. Compressed Domain Reconstruction

The reconstruction process in the encoder is listed as follows.

1. The best Macroblock Type, Sub-Macroblock Types, Luma Intra modes, Chroma Intra modes, Number of SKIP Macroblocks, Number of nonzero coefficients of each Sub-Macroblocks, motion vectors, and QPs are stored appropriately for further reference.
2. The reconstructed Macroblock which is calculated by best mode decided is stored in reconstructed frame for further prediction / reference.
3. The predicted Macroblock which is calculated by best mode decided is used in decoding path of the encoder to find the reconstructed residue Macroblock.
4. The quantization, transform converters are followed the same as explained in decoding process. The QP could be forced to adjust the compression ratio.
5. After finishing all Macroblocks in the frame, the deblocking filter is applied to smoothen the boundaries. The deblocking filtered reconstructed frame is kept as reference frame.
6. The unfiltered reconstructed frame is used for intra prediction of the current frame.

H. Entropy Encoding and Generation of H.264 bitstream

The residue coefficients are encoded by Context Adaptive Variable Length Coding (CAVLC) technique. The syntax elements are encoded by Exp-Golomb Encoding technique mentioned in [33].

III. EXPERIMENTATION

The compressed domain processes, i.e., Intra mode decision, Motion Estimation, Mode Decision and Motion Compensation are designed and developed. The entire Encoder was coded in MATLAB. The input of encoder was compressed domain frames. The inputs were generated by transforming the raw YUV 4:2:0 video with the help of core forward transform. The encoding process was performed and H.264 bitstreams were generated. The compliance of generated bitstream was checked by decoding the bitstream by JM decoder. It was found that the bitstream was in compliance with H.264 Standard. For the presentation in this research work, three different CIF sequences (which are 4:2:0 Chroma Subsampling) have been used as data to assess the efficacy of the algorithm developed. They are 1) Akiyo (which has low motion) 2) Foreman (Camera panning and fast motion) and 3) Mobile (which is colorful and various motion). Each video sequences are having 300 frames. The frame rate is 25 fps. The compression characteristics of those video sequences are shown in Fig. 2. It shows the motion / behavioral deviations between the sequences. This plot is obtained by compressing those video sequences by x264 software with different QP (varying from 1 to 51).

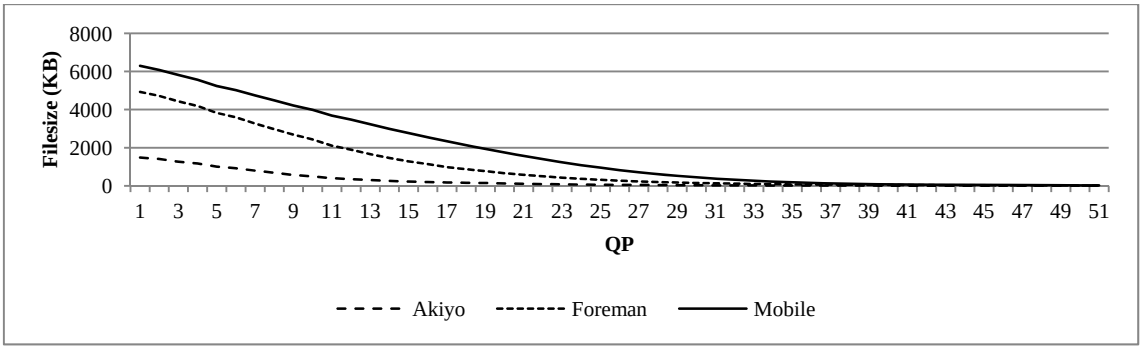

Fig. 2 Characteristics of CIF sequences (Akiyo, Foreman and Mobile)

IV. Results and its Analysis

Metrics measured as results of Transcoding are 1) File size, 2) Quality (in terms of PSNR) and 3) Complexity (in terms of mopf). Akiyo, Foreman and Mobile sequences are compressed by x264 encoder with QP = 7. The H.264 bitstreams are used as input of the transcoder models. The screen size is resized from CIF (352 x 288) to QCIF (176 x 144). The QP is varied at the encoder from 7 to 42 insteps of 7 (i.e., QP = 7, 14, 21, 28, 35, 42). File Sizes (in Kilo Bytes) of the transcoded bitstreams for Reference Model, Standalone Mode and Reuse Mode of Research Model are listed in Table II. The quality of output of those models for the given bitstreams is measured in terms of PSNR (in dB) and listed in Table III. The complexity in terms of million operations per frame (mopf) of those models for the given bitstreams is listed in Table IV.

TABLE II
FILE SIZES (KILO BYTES) OF TRANSCODED BITSTREAMS

QP	Akiyo		Foreman		Mobile	
	Reference Model	Research Model	Reference Model	Research Model	Reference Model	Research Model
7	883	865	3023	3443	5261	5354
14	328	326	1419	1775	3410	3518
21	148	142	617	826	1948	2013
28	61	58	223	332	842	857
35	27	26	85	134	241	229
42	15	14	40	58	71	79

TABLE III
QUALITY (PSNR IN DB) OF TRANSCODED BITSTREAMS

QP	Akiyo		Foreman		Mobile	
	Reference Model	Research Model	Reference Model	Research Model	Reference Model	Research Model
7	53.5	53.8	52.6	53.2	52.3	52.8
14	48.5	48.9	46.7	47.3	45.6	46.4
21	44.2	44.3	41.8	42.1	39.5	39.8
28	39.4	38.6	37.1	37.2	33.4	33.8
35	35.1	33.6	33.3	33.1	28.2	28.2
42	31.4	30.8	30.0	28.7	24.5	22.6

TABLE IV
COMPLEXITY (MOPF) OF TRANSCODED BITSTREAMS

QP	Akiyo		Foreman		Mobile	
	Reference Model	Research Model	Reference Model	Research Model	Reference Model	Research Model
7	44.23	62.29	47.18	67.40	63.62	89.60
14	38.88	49.85	45.94	57.42	48.03	60.04
21	33.73	47.51	41.28	55.79	40.75	53.62
28	34.39	47.11	37.39	52.67	29.14	37.85
35	28.94	40.18	32.72	44.83	28.66	35.83
42	20.85	29.79	29.10	39.32	18.60	23.25

1. Compliance to H.264 Standard

The output of the transcoder must be compliance to H.264 Standard.

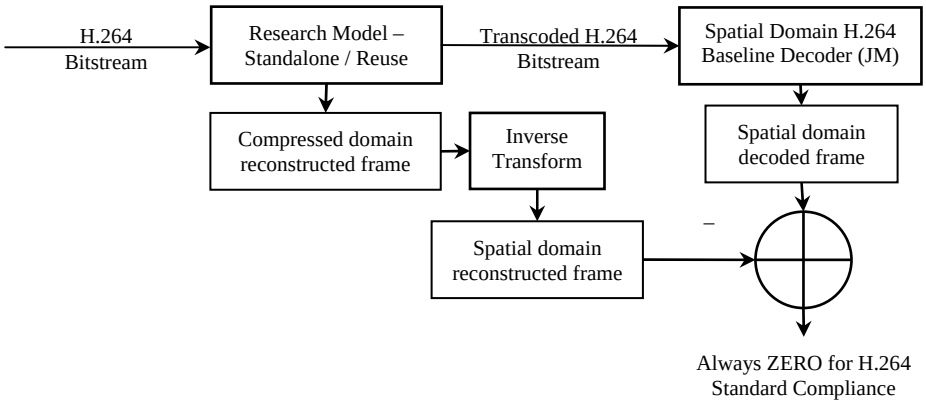

Fig. 3 Architecture for H.264 Standard compliance check

The syntax elements for those transcoded bitstreams are checked and found compliance with H.264 Standard by simply decoding the transcoded bitstreams by JM decoder as shown in Fig. 3. The analysis of results between research model and reference model is done in terms of the metrics as follows, 1) Compliance to H.264 Standard, 2) Perceptual Video Quality, 3) Objective Quality (PSNR) of Frames, 4) Bits consumed by Frames, 5) Complexity (mopf) of Frames, 6) Objective Quality (PSNR) vs. Bitrate – RD-Plot, and 7) Quality + Bitrate + Complexity of Frames – RDC-Plot.The compressed domain reconstructed frames which are used in research model are converted to spatial domain reconstructed frames. The transcoded H.264 bitstream is decoded to spatial domain frames by reference decoder. These two video frames are compared and found same, resulting the transcoded H.264 bitstream is in compliance with H.264 Standard.

2. Perceptual Video Quality

The visual quality and bitrate of the research work are compared with reference model. The quality of the resized video is comparable with the reference model and found less distortion. Three different input bitstreams of same screen size, CIF (352 x 288) with QP = 7 are transcoded to QCIF (176 x 144) with QP = 14. The sample shots are shown in Fig. 4 to 6. Akiyo sequence, which is slow motion, news reading sequence is shown in Fig. 4. Foreman sequence, which is commonly used in Video compression world, is shown in Fig. 5. The decoded bitstream (in CIF resolution) is shown in left side of Fig. 5. The video sequence is resized to QCIF resolution. Then the resized video is encoded by reference model and research work, which are shown in right side of Fig. 5. The fast moving and colorful sequence is mobile. The decoded frame, encoded frame by reference model and research model of mobile sequence are shown in Fig. 6. The perceptual visual qualities of those sequences are same. The visual comparisons of reference and research model of those sequences transcoded with QP = 14 are shown Fig. 7.

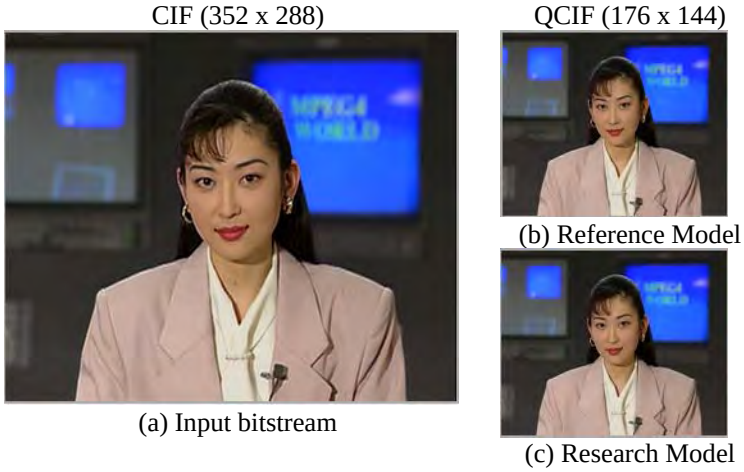

Fig. 4 Frame number 1 of Akiyo Sequence a) Input bitstream b) Encoded by reference model c) Encoded by research model

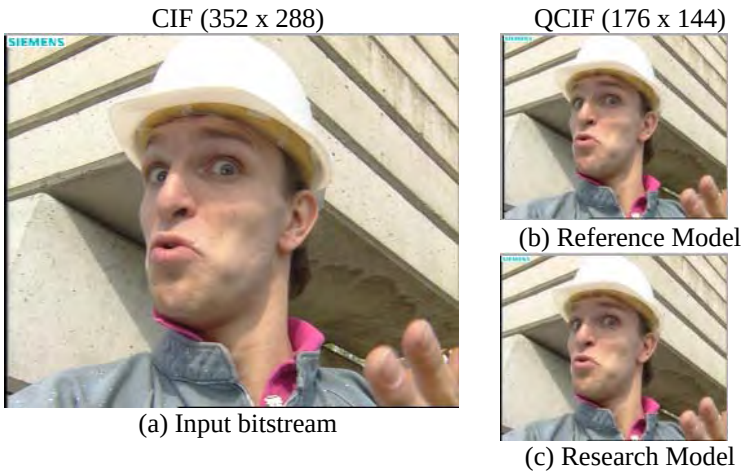

Fig. 5 Frame number 90 of Foreman sequence a) Input bitstream b) Encoded by reference model c) Encoded by research model

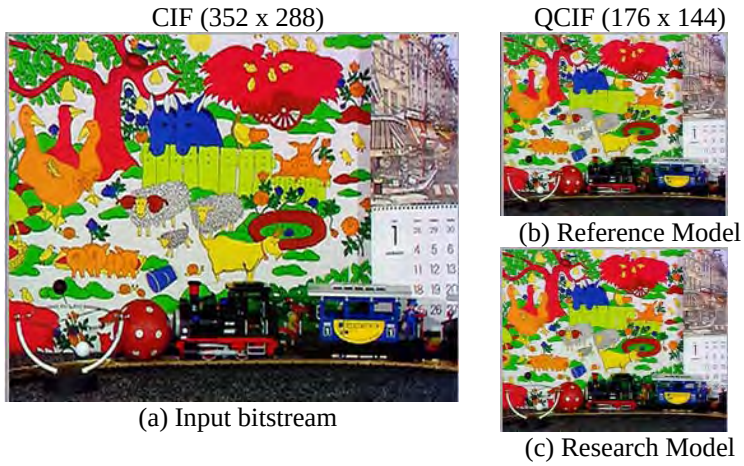

Fig. 6 Frame number 238 of Mobile sequence a) Input bitstream b) Encoded by reference model c) Encoded by research model

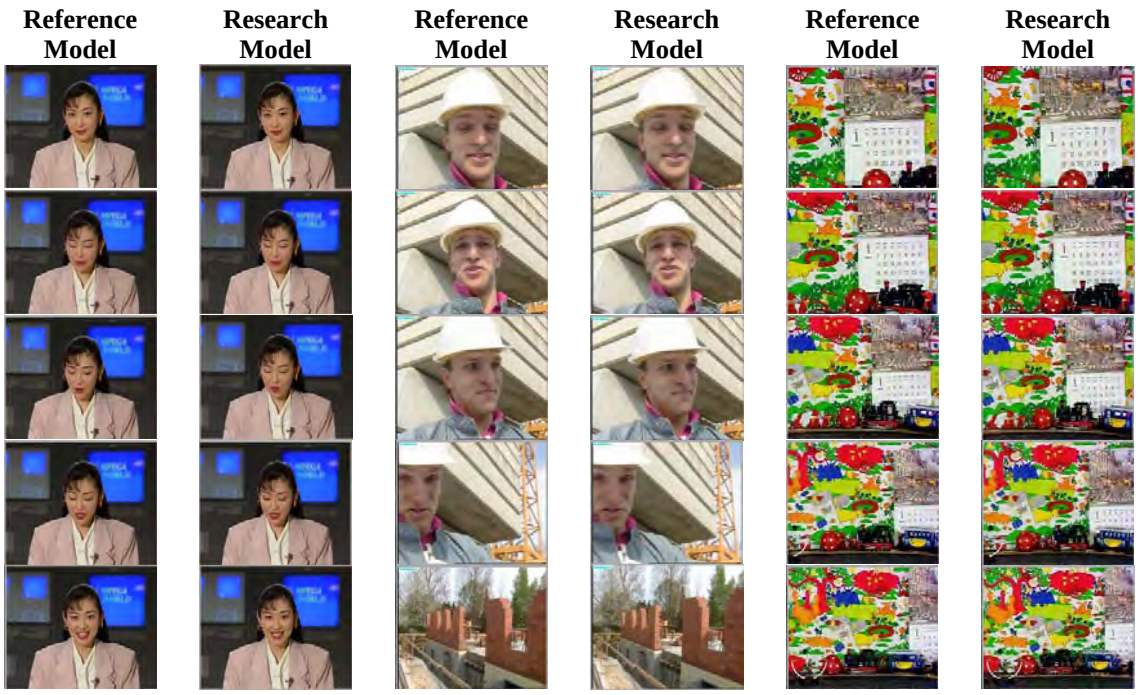

Fig. 7 Frame number 1, 61, 121, 181 and 241 of encoded Akiyo, Foreman and Mobile Sequence

3. Objective Quality (PSNR) of Frames

PSNRs with respect to resizer output (spatial domain) for each frame of reference model and research model output bitstreams with QP = 14 have been compared. The Fig. 8 to Fig. 10 indicated that PSNR of research model is very closer to reference model, but it very negligible. At least 2dB deviation cannot be identified by naked eyes. PSNR for each component is calculated as follows.

$$PSNR = 20 \times \log\left(\frac{255}{\sqrt{MSE}}\right) \tag{26}$$

where $MSE = \frac{SSE}{width \times height}$ and $SSE = \sum_{x=0}^{width-1}\sum_{y=0}^{height-1}(out(x,y) - ref(x,y))^2$

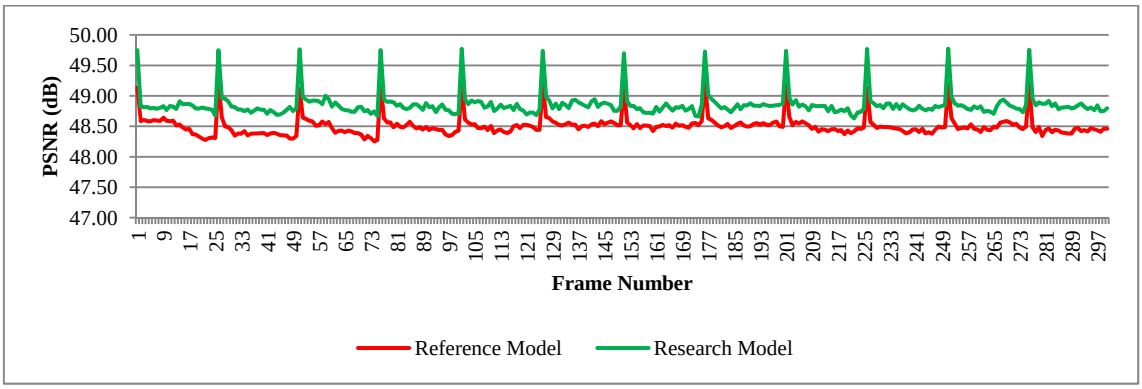

Fig. 8 Quality Comparison of Akiyo Sequence

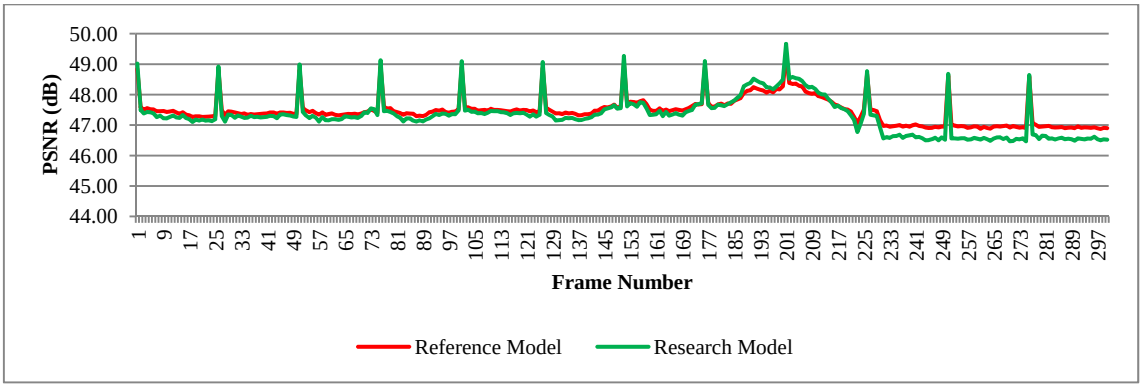

Fig. 9 Quality Comparison of Foreman Sequence

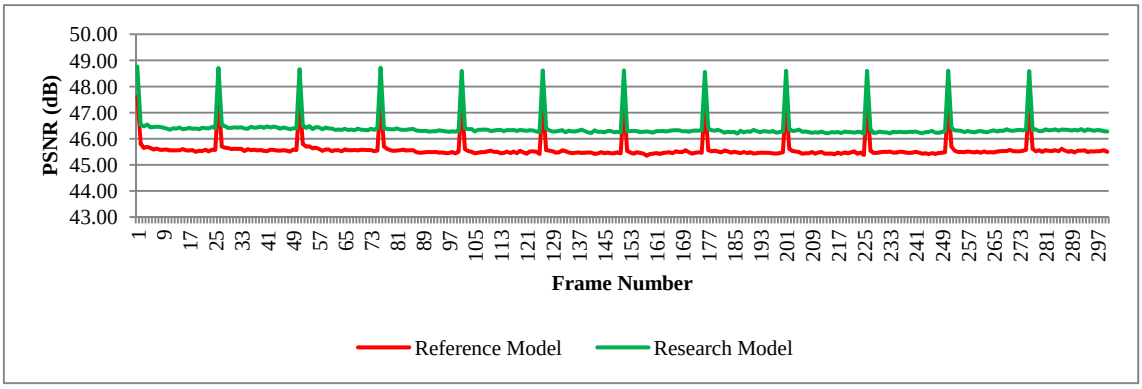

Fig. 10 Quality Comparison of Mobile Sequence

For a 4:2:0, average quality i.e., PSNR of a frame is calculated as a weighted average of three components as shown below.

$$PSNR = \frac{4 \times PSNRY + PSNRCb + PSNRCr}{6} \tag{27}$$

where $PSNRY$ is PSNR of Luma component, $PSNRCb$ is that of Cb and $PSNRCr$ is that of Cr component. The average PSNR at different bitrates of different sequences are shown in Fig. 11 to Fig. 13.

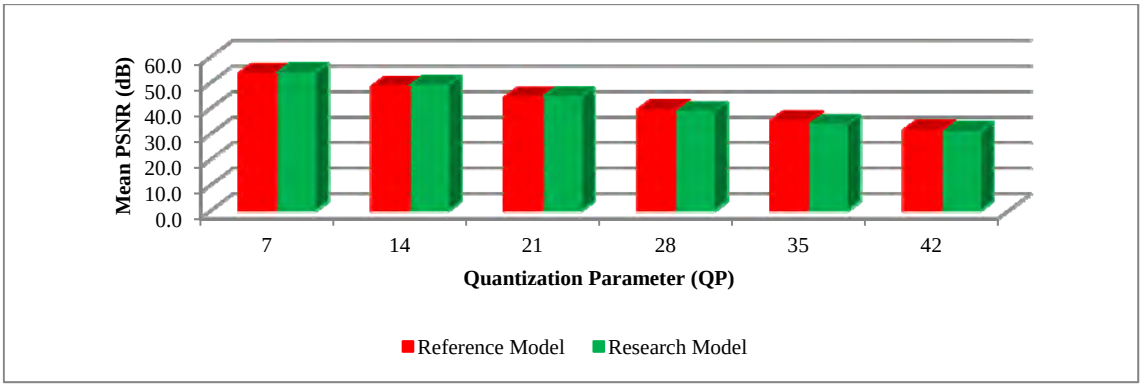

Fig. 11 Mean PSNR vs. QP of Akiyo Sequence

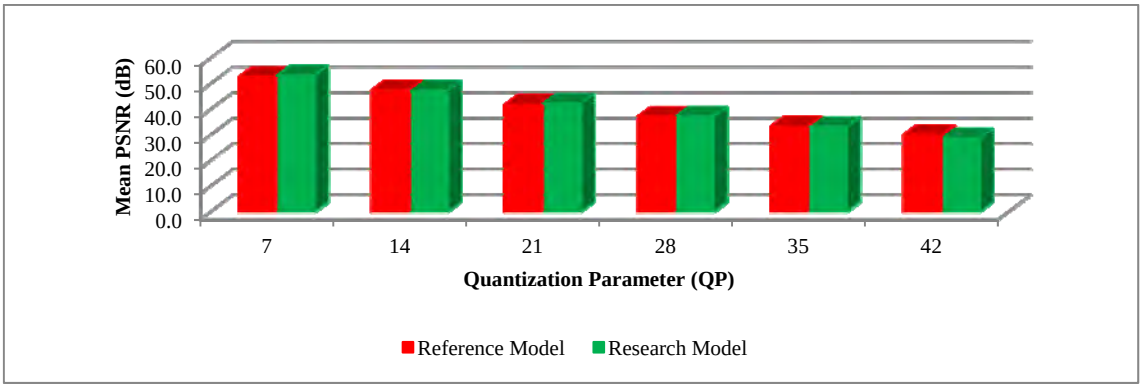

Fig. 12 Mean PSNR vs. QP of Foreman Sequence

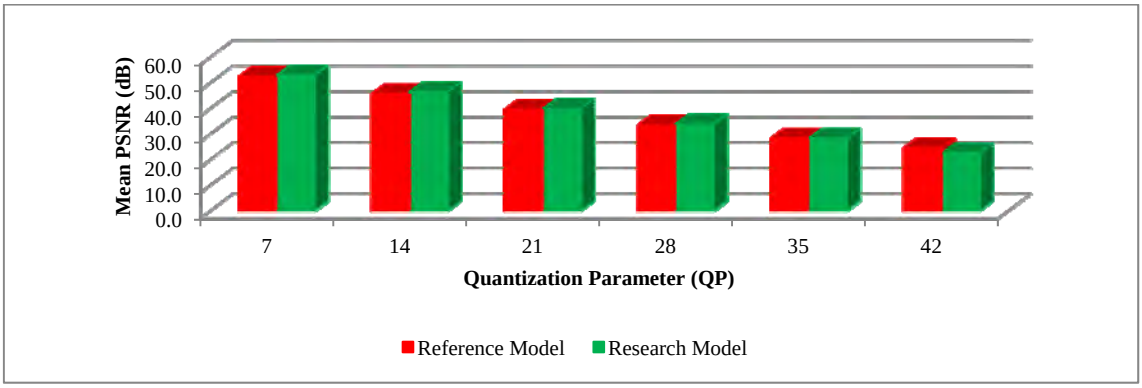

Fig. 13 Mean PSNR vs. QP of Mobile Sequence

4. Bits consumed by Frames

The Fig. 14 to 16 showed that the bits consumed by each frame of those sequences. The results showed that research model has taken 10% extra bits that of reference model for each frame. But the deviation may not be visible in Akiyo and Mobile; but it is visible in Foreman plot. The reasons for taking excessive bits than reference model are 1) The reference model works on spatial domain; research model works on compressed domain where the precision is limited in mode decision and motion estimation, 2) The reference model uses rigorous sliding techniques to find the best match in motion estimation; the research model uses restricted searching technique to address the hardware implementation.

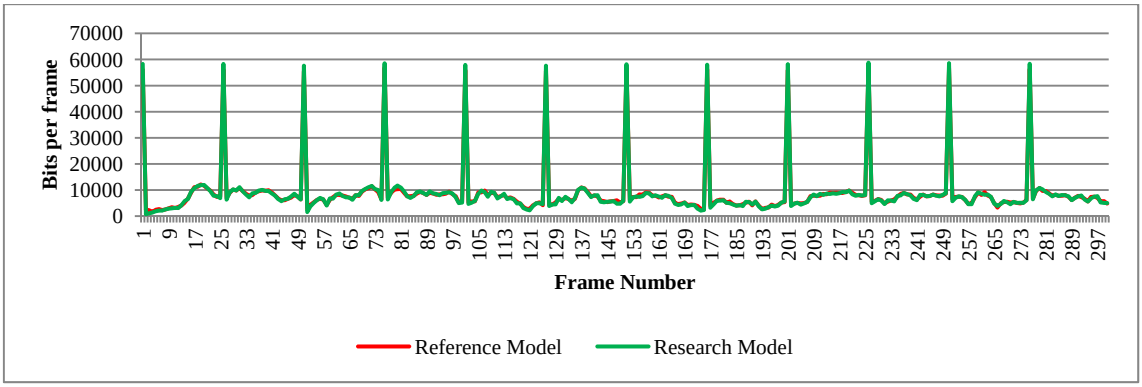

Fig. 14 Bits spent Comparison of Akiyo Sequence

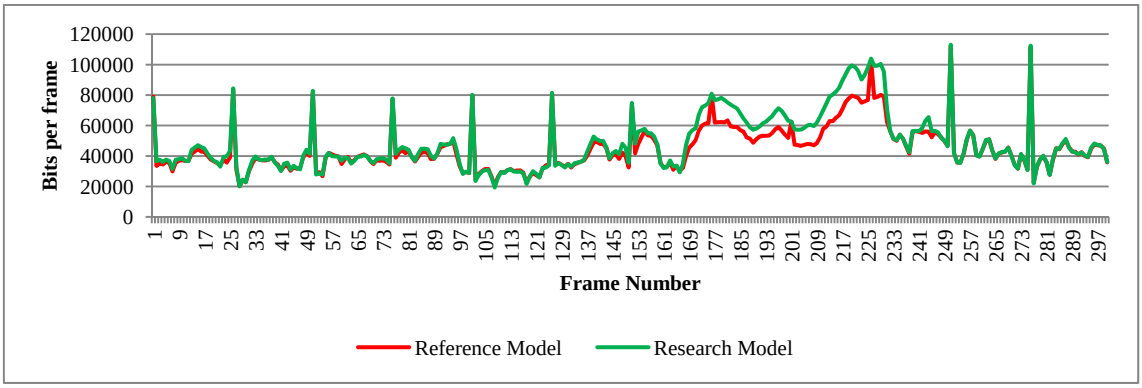

Fig. 15 Bits spent Comparison of Foreman Sequence

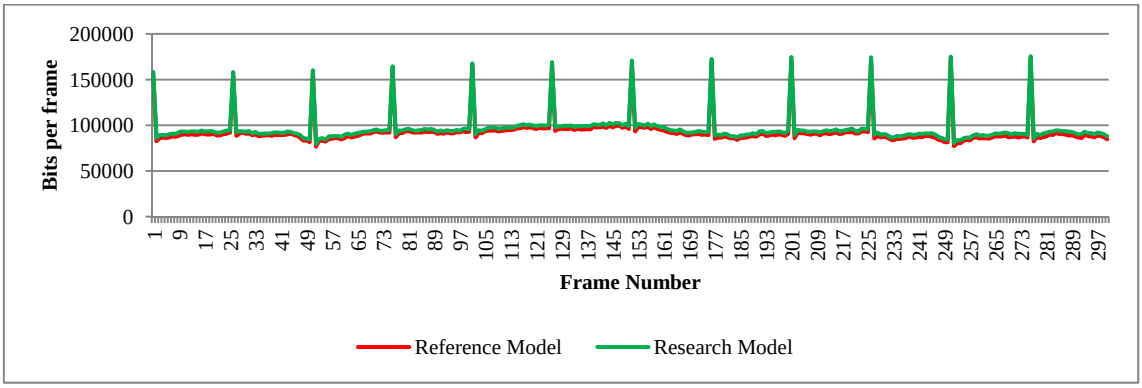

Fig. 16 Bits spent Comparison of Mobile Sequence

The bits taken by bitstreams for different QPs are compared in Fig. 17 to 19. The bits of the entire sequence are summed to get total bits (file size) for a given QP. The Figs. 17 to 19 are plot with file size vs. QP. It is observed that the file size of research model is 10% more than that of reference model.

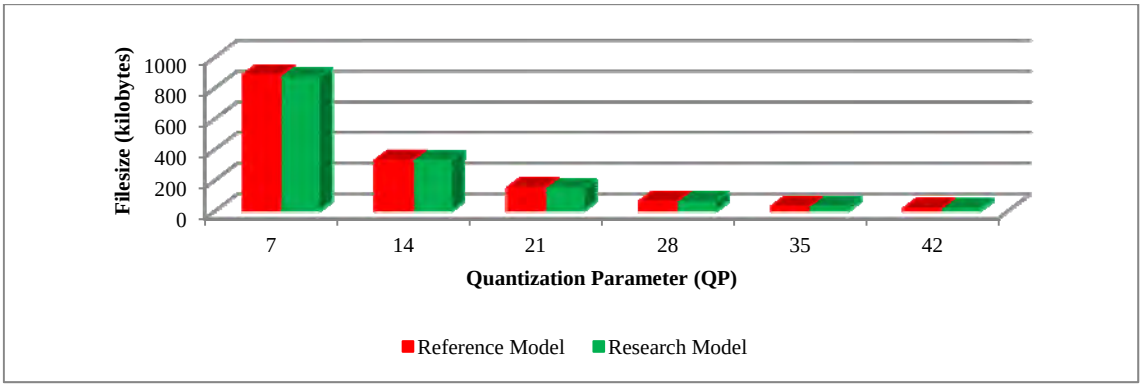

Fig. 17 File size vs. QP of Akiyo Sequence

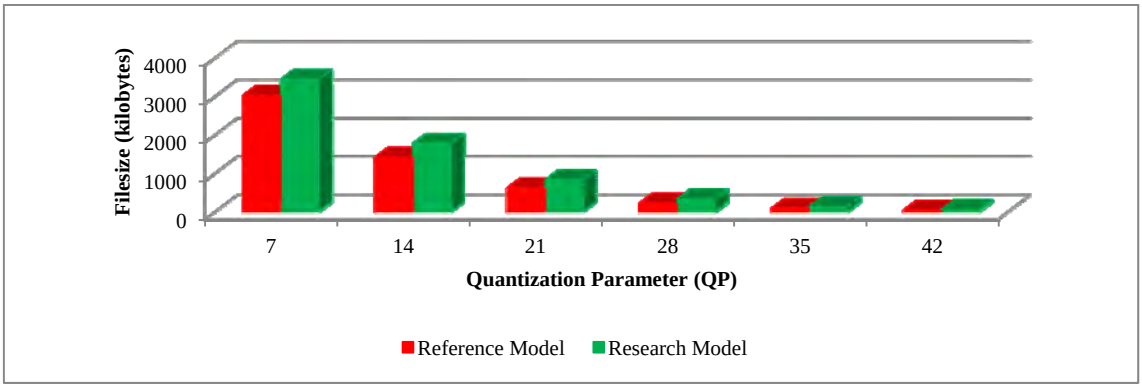

Fig. 18 File size vs. QP of Foreman Sequence

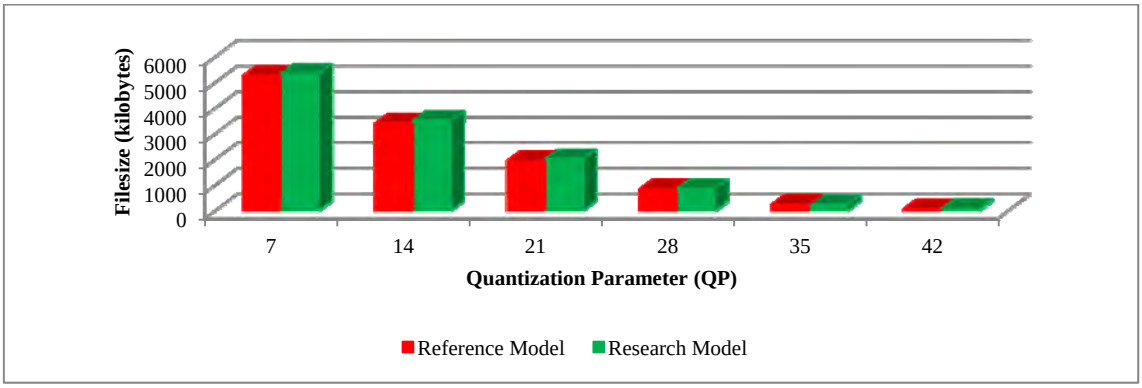

Fig. 19 File size vs. QP of Mobile Sequence

5. Complexity (mopf) of Frames

The computational complexity is calculated based on number of operations such as addition, subtraction, multiplication and shifting in the transcoding path. The complexity of decoding, resizing and encoding each frame is computed in terms of operations. The computational complexities obtained from transcoding by reference model and transcoding by research model are compared which are shown in Fig. 20 to 22. Because research model worked in compressed domain, its complexity is 10-20% higher than that of pixel domain reference model.

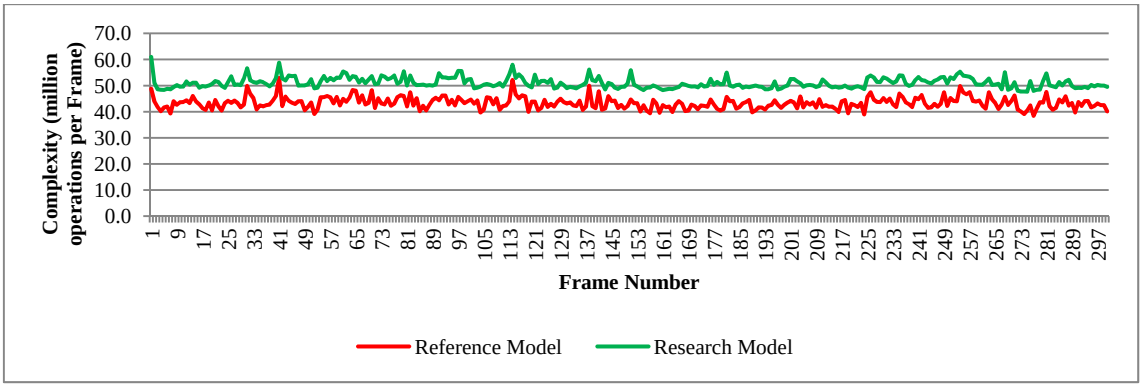

Fig. 20 Complexity comparison for Akiyo Sequence

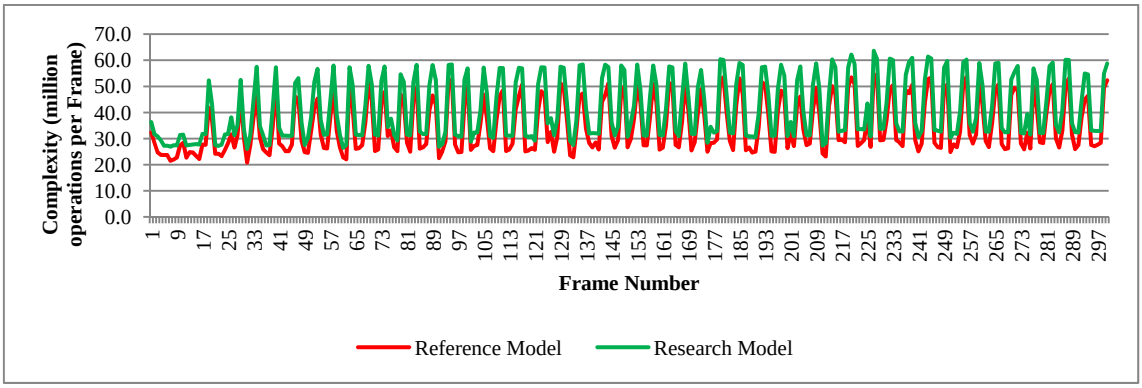

Fig. 21 Complexity comparison for Foreman Sequence

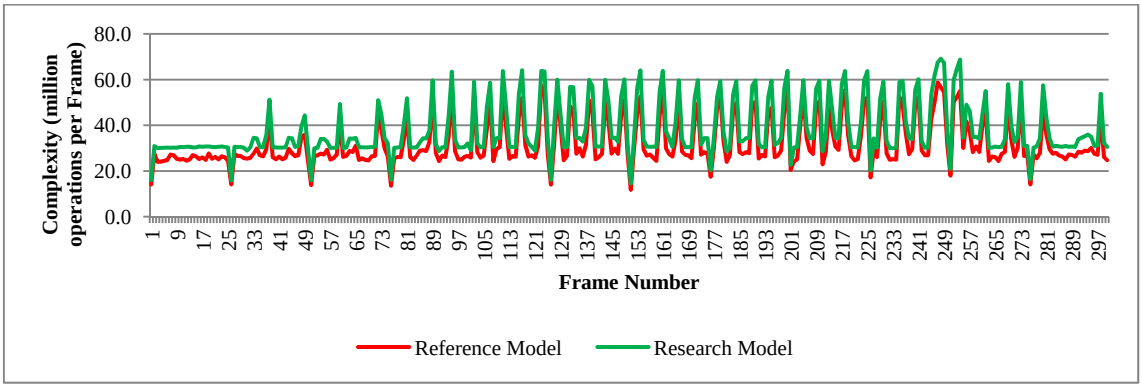

Fig. 22 Complexity comparison for Mobile Sequence

The computational complexities of the entire sequence are averaged to get mean complexity (mopf) for a given QP. The computational complexities by bitstreams for different QPs are compared in Fig. 23 to 25, which are plot with mean Complexity vs. QP.

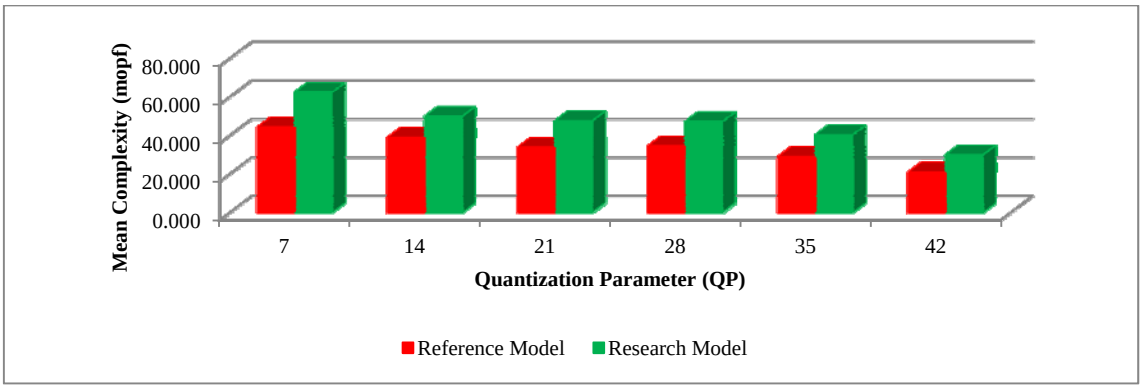

Fig. 23 Mean Complexity vs. QP for Akiyo Sequence

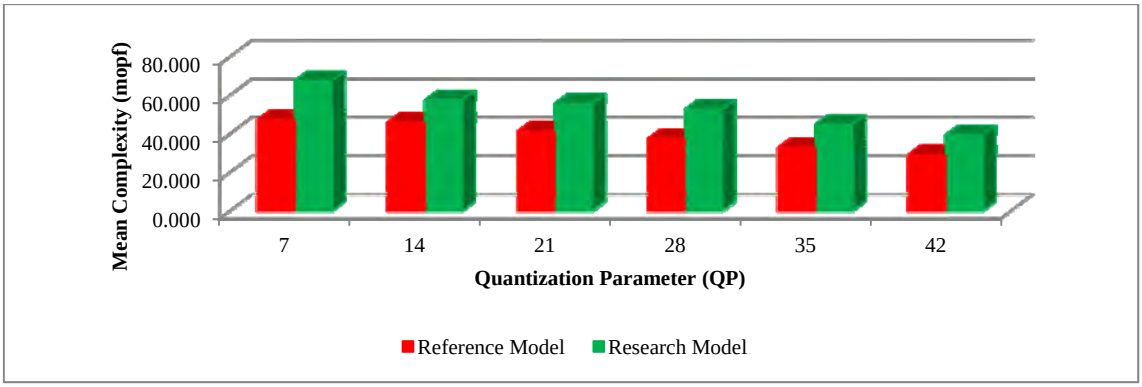

Fig. 24 Mean Complexity vs. QP for Foreman Sequence

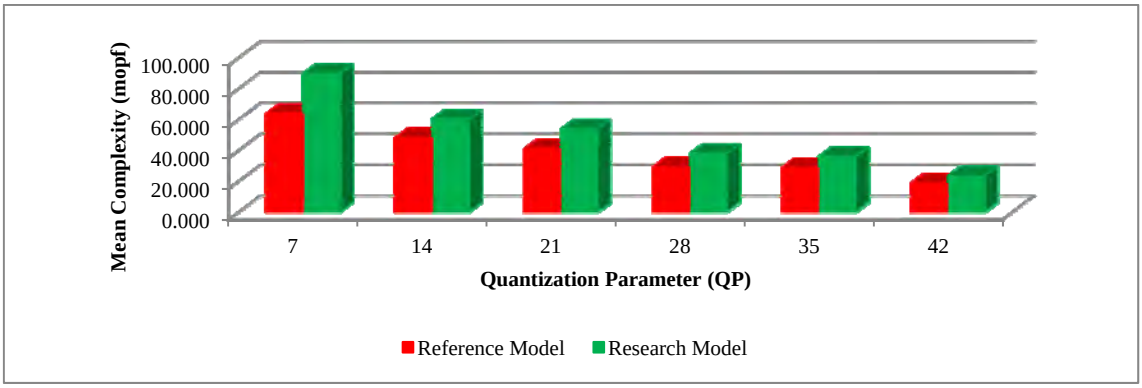

Fig. 25 Mean Complexity vs. QP for Mobile Sequence

6. PSNR vs. Bitrate (RD Plot)

Next the PSNR vs. bitrate plot (RD plot) is done to evaluate the quality. The Video quality vs. bitrate of the research model is very closer with the reference model. The comparisons for the above-said Akiyo, Foreman and Mobile sequences are shown in Fig. 26 to 28. It is clearly noted that the quality at different bitrates is on-par with that of reference software.

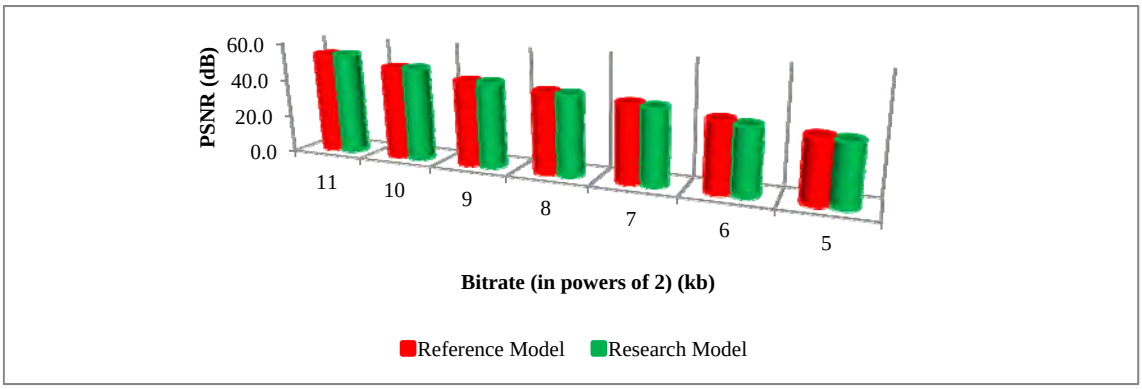

Fig. 26 Quality vs. Bitrate of Akiyo Sequence

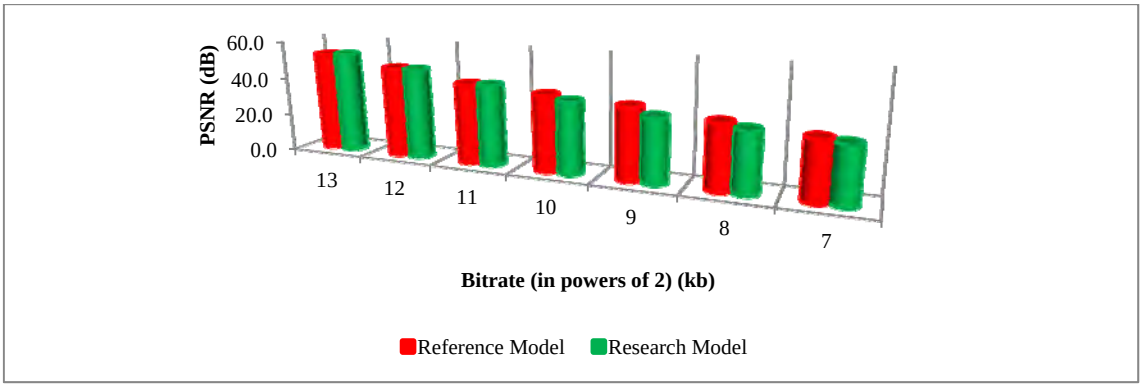

Fig. 27 Quality vs. Bitrate of Foreman Sequence

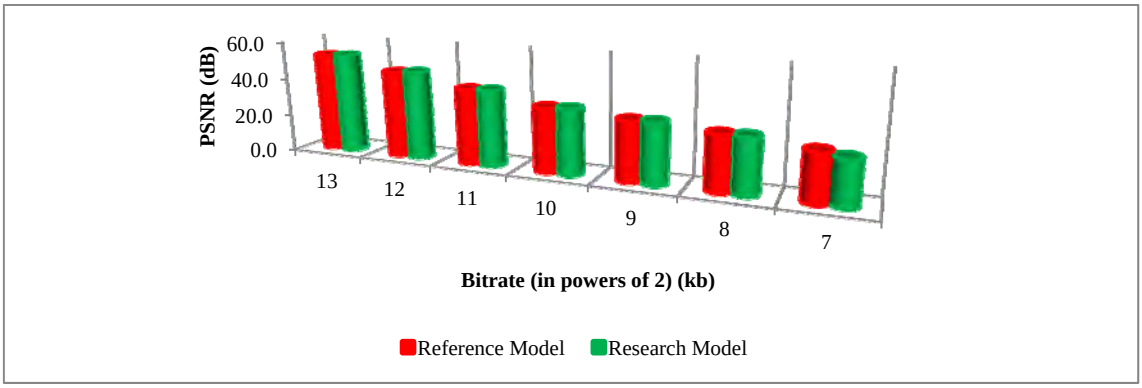

Fig. 28 Quality vs. Bitrate of Mobile Sequence

7. Rate-Distortion-Complexity (RDC) Plot

The new metric called RDC-plot is introduced in this research work. The combination of bits consumed, the quality in terms of SSE and complexity in mopf are combined as follows for each frame.

$$RDC = SSE + \lambda_A \times bits + \lambda_B \times mop \quad (28)$$

where

$$\lambda_A = \frac{\lambda_1(QP)}{2^{18}} \; and \; \lambda_B = \frac{\lambda_2(QP)}{2^{16} \times 10^2}$$

$\lambda_A \; and \; \lambda_B$ are Lagrangian Multipliers. The RDC-Plots for the same sequences are plotted in Fig. 29 to 31.

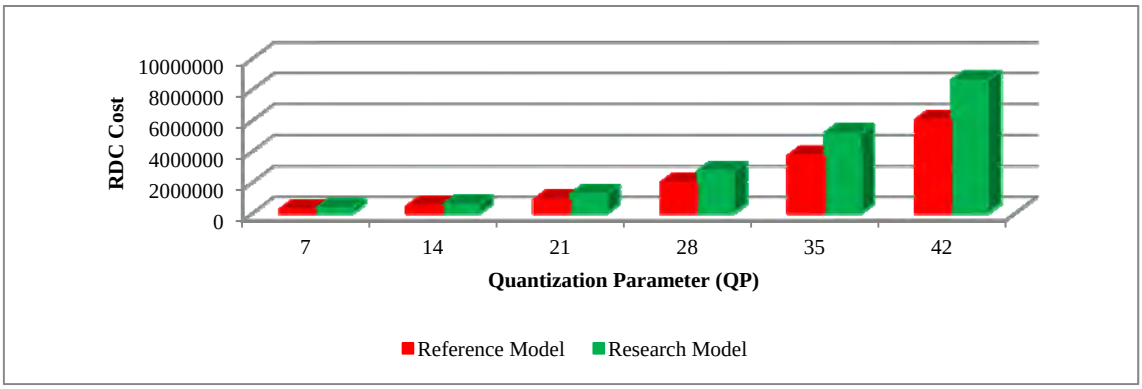

Fig. 29 Rate-Distortion-Complexity comparison for Akiyo Sequence

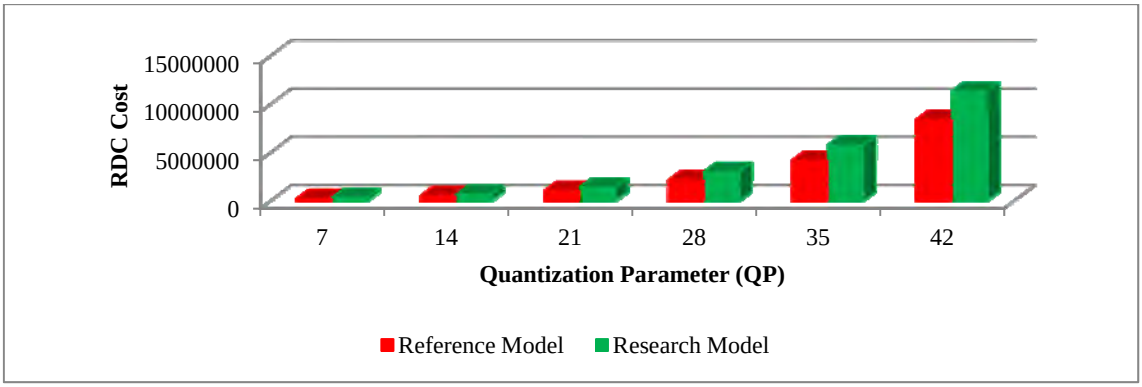

Fig. 30 Rate-Distortion-Complexity comparison for Foreman Sequence

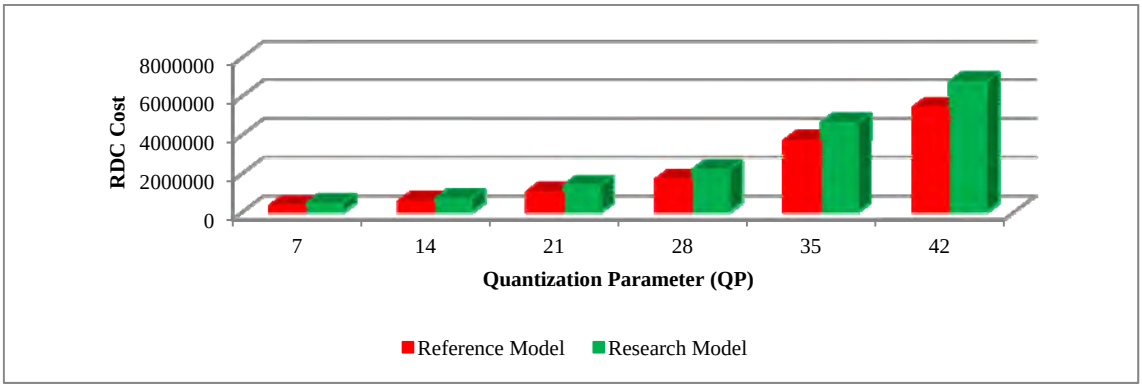

Fig. 31 Rate-Distortion-Complexity comparison for Mobile Sequence

V. CONCLUSION

Video transcoding is a core technology for providing universal multimedia access by the Internet users with different access links and devices. Video compression algorithms used in the standardizing H.264 are very different and difficult from that of in the previous traditional video compression standards. This research work proposed a new approach of compressed domain homogenous video encoder in the transcoder path using integer transform in compliance with H.264 Standard. The complete architecture of compressed domain encoder using Integer transform in compliance with H.264 Standard has been developed. Standard YUV sequences have been used and the efficacy of this algorithm has been demonstrated through various performance parameters. The compressed domain processes have been modified to suit the hardware implementation. This research work paved a way of using integer transform in video transcoding architectures. Another metric RDC has been proposed to measure the efficacy of encoder. The integer transform based processes definitely help implementing the algorithm in hardware.

REFERENCES

[1] Hung, J, Tihao, C & Ya-Qin, Z, 'Scalable Rate Control for MPEG-4 Video', IEEE Transactions on Circuits and Systems for Video Technology, vol. 10, no. 6 (2000) pp. 878-894.

[2] Jordi, R & Shawmin, L, 'Rate Control in DCT Video Coding for Low-Delay Communications', IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 1 (1999) pp. 172-185.

[3] Tihao, C & Ya-Qin, Z, 'A New Rate Control Scheme Using Quadratic Rate Distortion Model', IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 1 (1997) pp. 246-250.

[4] Ishfaq, A, Xiaohui, W, Yu, S & Ya-Qin, Z, 'Video Transcoding: An Overview of Various Techniques and Research Issues', IEEE Transactions on Multimedia, vol. 7, no. 5 (2005) pp. 793-804.

[5] Fernandez, G, E, Hari, K, Martínez, J, L, Cuenca P, Orozco-Barbosa, L & Garrido, A, 'An MPEG-2 to H.264 Video Transcoder in the Baseline Profile', IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 5 (2010) pp. 763-768.

[6] Martinez, J, L, Fernandez, E, G, Hari, K, Fernando W, A, C & Cuenca P, 'Wyner-Ziv to H.264 Video Transcoder for Low Cost Video Encoding', IEEE Transactions on Consumer Electronics, vol. 55, no. 3, (2009) pp. 1453-1461.

[7] Martinez, J, L, Fernandez, E, G, Hari, K, Cuenca, P, 'Motion vector refinement in a Wyner–Ziv to H.264 transcoder for mobile telephony', Institute of Engineering and Technology – Image Processing, vol. 3, no. 6, (2009) pp. 335-339.

[8] Rashad, J & Hari, K, 'Low Complexity Intra MB Encoding in AVC/H.264', IEEE Transactions on Consumer Electronics, vol. 55, no. 1 (2009) pp. 277-285.

[9] Hari, K & Phil, K, 'Dynamic Motion Estimation for Transcoding P Frames in H.264 to MPEG-2 Transcoders', IEEE Transactions on Consumer Electronics, vol. 54, no. 2 (2008) pp. 657-662.

[10] Gerardo, F, Hari, K, Pedro, C & Luis, O, 'RD-Optimization for MPEG-2 to H.264 Transcoding', IEEE International Conference on Multimedia and Expo, Toronto, (2006) pp. 309-312.

[11] Gerardo, F, Hari, K, Pedro, C & Luis, O, 'Reducing Motion Estimation Complexity in MPEG-2 to H.264 Transcoding', IEEE International Conference on Multimedia and Expo, Beijing, (2007) pp. 440-443.
[12] Gerardo, F, Jens, B, Jose, A, G, Hari, K, Pedro, C, Luis, O & Andre, K, 'Low-Complexity Heterogeneous Video Transcoding Using Data Mining', IEEE Transactions on Multimedia, vol. 10, no. 2 (2008) pp. 286-299.
[13] Gerardo, F, Hari, K, Pedro, C, Luis, O & Antonio, G, 'A Fast MB Mode Decision Algorithm for MPEG-2 to H.264 P-Frame Transcoding', IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no. 2 (2008) pp. 172-185.
[14] Branko, P & Hari, K, 'DCT Domain Intra MB Mode Decision for MPEG-2 to H.264 Transcoding', Digest of Technical Papers – International Conference on Consumer Electronics, Vietnam, (2006) pp. 419-420.
[15] Hari, K & Branko, P, 'Exploiting the Directional Features in MPEG-2 for H.264 Intra Transcoding', IEEE Transactions on Consumer Electronics, vol. 52, no. 2, (2006) pp. 706-711.
[16] Hari, K, 'Issues in H.264/MPEG-2 Video Transcoding', IEEE Consumer Communications and Networking Conference, Las Vegas, (2004) pp. 657-659.
[17] Siwei, M, Wen, G, Peng, G & Yan, L, 'Rate Control for Advance Video Coding (AVC) Standard', in Proceedings of the International Symposium on Circuits and Systems, Bangkok, vol. 2, (2003) pp. II-892 to II-895.
[18] Gao, C, Shouxun, L & Yongdong, Z, 'A Fast Coefficients Conversion Method for the Transform Domain MPEG-2 to H.264 Transcoding', International Conference on Digital Telecommunications, Cote de'Azur, (2006) pp. 17.
[19] Pedro, A & Mohammed, G, 'A Frequency-Domain Video Transcoder for Dynamic Bitrate Reduction of MPEG-2 Bit Streams', IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 8, (1998) pp. 953-967.
[20] Iain E, R, 'Whitepaper on 4x4 Transform and Quantization in H.264/AVC', (2009) http://www.vocodex.com.
[21] Gerardo, F, Pedro, C, Luis, O & Antonio, G, 'Computational Complexity Reduction of Intra-Frame Prediction in MPEG-2/H.264 Video Transcoders', IEEE International Conference on Multimedia and Expo, Amsterdam, (2005) pp. 707-710.
[22] Qiang, T, Panos, N & Rabab, W, 'An Efficient Re-quantization Error Compensation for MPEG-2 to H.264 Transcoding', IEEE International Symposium on Signal Processing and Information Technology, Columbia, (2006) pp. 530-535.
[23] Yu-Wen, H, Bing-Yu, Hh, Tung-Chien, C & Liang-Gee, C, 'Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder', IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 3, (2005) pp. 378 – 401.
[24] Chia-Wei, T, Heng-Yao, L, Bin-Da, L & Jar-Ferr, Y, 'Transform-Domain Partial Prediction Algorithm for Intra Prediction in H.264/AVC', IEEE International Symposium on Circuits and Systems, Taipei, (2009) pp. 1245-1248.
[25] Inchoon, C, Jeyun, L & Byeungwoo, J, 'Fast Coding Mode Selection With Rate-Distortion Optimization for MPEG-4 Part-10 AVC/H.264', IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 12, (2006) pp. 1557-1561.
[26] Jun, X, Anthony, V, Shun-ichi, S, Kazuo, S, 'MPEG-2 to H.264/AVC Transcoding for Efficient Storage of Broadcast Video Bitstreams', Digest of Technical Papers – International Conference on Consumer Electronics, Vietnam, (2006) pp. 417-418.
[27] Kan, C, Aidong, M & Wenhao, Z, 'Fast Intra-prediction Mode Decision for H.264/AVC', International Colloquium on Computing, Communication, Control, and Management, Sanya, (2009) pp. 69-73.
[28] Mohammed, G, S & Jonathan, W, 'Enhanced Low Complex Cost Function for H.264/AVC Intra Mode Decision', International Conference on Multimedia and Signal Processing, Guangxi, (2011) pp. 46-50.
[29] Chao-Hsuing, T, Hung-Ming, W & Jar-Ferr, Y, 'Enhanced Intra-4x4 Mode Decision for H.264/AVC Coders', IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 8, (2006) pp. 1027-1032.
[30] Essaki Muthu, P, Gemson R, M, O, 'A Computationally efficient method to find Transformed residue coefficients in Intra 4x4 mode decision in H.264 Encoder', International Journal of Electronics and Communication Engineering & Technology, vol. 3, no. 3, (2012) pp. 84-102.
[31] Bharanitharan, K, Bin-Da, L, Jar-Ferr, Y & Wen-Chih, T, 'A Low Complexity Detection of Discrete Cross Differences for Fast H.264/AVC Intra Prediction', IEEE Transactions on Multimedia, vol. 10, no. 7, (2008) pp. 1250-1260.
[32] Essaki Muthu, P, Gemson R, M, O, 'Estimation of Bitlength of Transformed-Quantized Residue Coefficients with Context Information and its Syntax Elements for mode decision in H.264 Baseline Encoder', International Journal of Computer Engineering Technology, vol. 3, no. 3, (2012) pp. 168-183.
[33] H.264 Standard – Advanced Video Coding for generic audio-visual services, International Telecommunication Union, November 2007.

AUTHOR PROFILE

P. Essaki Muthu received the Master's degree in Digital Electronics & Communication Engineering from Manipal Institute of Technology, Manipal in 2001. His passion is teaching and he is continuing pedagogy and research in Institutions and Industries. He is a research student of Dr. MGR Educational and Research Institute, Chennai, INDIA. His interests are in Video compression, Video transcoding, Video filters, Wavelet transform, Cryptography and Signal Processing. He designs and develops algorithms for various multimedia transmission pipelines.