An Efficient Hardware Architecture of Intra Prediction and TQ/IQIT Module for H.264 Encoder

Kibum Suh, Seongmo Park, and Hanjin Cho

In this paper, we propose a novel hardware architecture for an intra-prediction, integer transform, quantization, inverse integer transform, inverse quantization, and mode decision module for the macroblock engine of a new video coding standard, H.264. To reduce the cycle of intra prediction, transform/quantization, and inverse quantization/inverse transform of H.264, a reduction method for cycle overhead in the case of I16MB mode is proposed. This method can process one macroblock for 927 cycles for all cases of macroblock type by processing 4×4 Hadamard transform and quantization during 16×16 prediction. This module was designed using Verilog Hardware Description Language (HDL) and operates with a 54 MHz clock using the Hynix 0.35 µm TLM (triple layer metal) library.

Keywords: Intra prediction, integer transform, quantization, inverse integer transform, inverse quantization, H.264.

Manuscript received Jan. 31, 2005; revised July 20, 2005.
The material in this work was presented in part at IT-SoC 2004, Seoul, Korea, Oct. 2004.

Kibum Suh (phone: +82 42 630 9705, email: kbsuh@wsu.ac.kr), is with the Department of Electronic Engineering, Woosong University, Daejeon, Korea.
Seongmo Park (email: smpark@etri.re.kr) and Hanjin Cho (email: hjcho@etri.re.kr) are with Basic Research Laboratory, ETRI, Daejeon, Korea.

I. Introduction

Nowadays, H.264 is drawing considerable attention because it can encode video with approximately three times fewer bits than the comparable MPEG-2 algorithm. It also offers several features such as squeezing more television programs into a given channel bandwidth, delivering quality video over bandwidth-constrained networks (for example, 3G mobile), and fitting a high-definition movie feature onto a standard DVD [1].

The major features of H.264 compared with MPEG-4 (ISO/IEC 14496-2) are as follows: H.264 contains improved inter-prediction and motion compensator techniques, improved intra spatial prediction, integer transform, a deblocking filter, a context adaptive entropy coding method, and so on.

If a block or macroblock is encoded in intra mode, a prediction block is formed based on previously encoded and reconstructed (but un-filtered) blocks. This prediction block P is subtracted from the current block prior to encoding. For the luminance (luma) samples, P may be formed for each 4×4 sub-block or for a 16×16 macroblock. There are a total of nine optional prediction modes for each 4×4 luma block; four optional modes for a 16×16 luma block; and four modes for a 4×4 chroma intra block.

If a macroblock is in p-slice or b-slice, the rd_cost (distortion + λ•rate) value of motion estimation is compared with the smallest rd_cost value among the intra prediction (16×16 luma prediction/4×4 luma prediction). The mode decision of a macroblock is determined by selecting the mode of the smallest rd_cost value among the probable modes.

For the H.264 encoder, after the mode decision of a macroblock, each residual macroblock is quantized and transformed. Whereas previous standards such as MPEG-1, MPEG-2, MPEG-4, and H.263 made use of the 8 × 8 discrete cosine transform (DCT) as the basic transform, H.264 uses three transforms depending on the type of residual data that is coded in the bitstream: a transform for the 4×4 array of luma DC coefficients in intra macroblocks (predicted in 16×16 mode), a transform for the 2×2 array of chroma DC coefficients (in any macroblock), and a transform for all other 4×4 blocks in the residual data.

Therefore, for the implementation of transform and quantization (TQ) and inverse quantization and inverse transform (IQIT), the control flow is dependent on the macroblock type and is more complex than the previous standards such as MPEG-1, MPEG-2, and MPEG-4 [2], [3].

Several researchers have been published on H.264 encoder hardware architecture. The contributions related to the motion estimator [4]-[7] and transformation [8], [9] are note worthy. For the hardware implementation of intra-prediction and transformation, the contribution of Chen and colleagues [10], [11] is a good solution for a D1 resolution intra frame coder. Here, they have used approximately 1280 cycles for one macroblock for the D1 resolution intra frame encoder. Using the DCT distortion measure, reconfigurable processor element (PE), and interleaved I4MB/I16MB prediction scheduling, they can reduce the gate count. Another paper by Chen and colleagues [12] on the H.264 level 3 inter-prediction encoder is an excellent contribution, but much of the intra prediction and transformation is not revealed.

In this paper, we propose an intra prediction hardware architecture and TQ/IQIT module architecture in an H.264 encoder that can process one macroblock for 927 cycles. We explain the operation of intra prediction and the TQ/IQIT module, and a reduction method for cycle overhead in the case of the I16MB mode in section II. The proposed architecture of intra prediction and the TQ/IQIT module are discussed in section III. Finally, experimental results and conclusions drawn are presented in sections IV and V, respectively.

II. Intra Prediction, TQ/IQIT Module Operation and the Cycle Overhead for I16 MB Mode

1. 4 × 4 Intra Prediction

The data dependency of 4×4 intra prediction mode is shown in Fig. 1. The pixels from a to p are predicted from A-L and M. Pixels labeled in upper case are reconstructed pixels. Because there are 16 4×4 blocks in a macroblock, the predictor cannot get the reconstructed pixels when previous blocks are not coded. JM [13] uses a two-pass algorithm to code these blocks. It requires all the blocks passing the transform, quantization, dequantization, and inverse transformation loop to do a 4×4 intra prediction, which is too complex for the hardware implementation. For this reason, Chen and others [14] proposed a modified algorithm, where original frame pixels instead of reconstructed pixels are used as predictors, and devised an error term to compensate for the estimation inaccuracy. But this modification results in quality degradation for the sequence, which needs many intra MBs. Therefore, we prefer the JM algorithm for hardware implementation. In our proposed architecture, intra prediction and the TQ/IQIT module are fully compatible with JM 8.5 software.

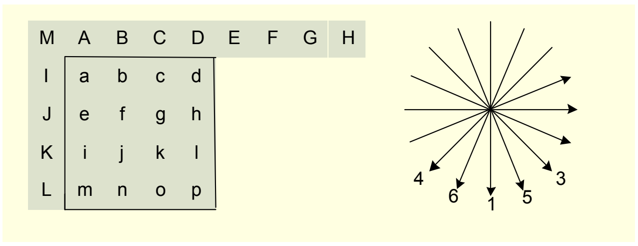


Fig. 1. 4×4 intra prediction.

2. 16 × 16 Intra Prediction

As an alternative to the 4×4 intra prediction mode, the entire 16×16 intra prediction component of a macroblock may be predicted in one operation. Figure 2 shows the data dependency of 16×16 intra prediction. The current macroblock is predicted by the 17 pixels from upper macroblocks and 16 pixels from the left macroblock. The 16×16 intra prediction has

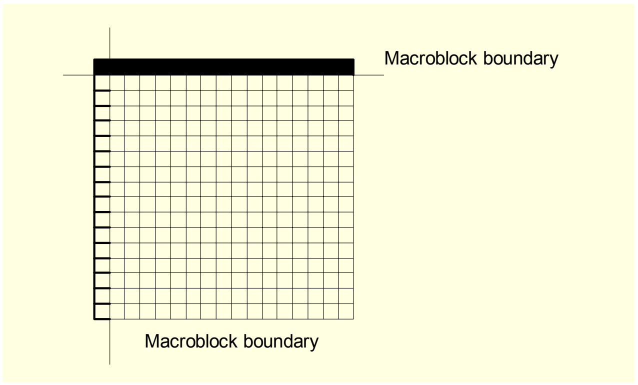


Fig. 2. Data dependency of 16 × 16 intra prediction.

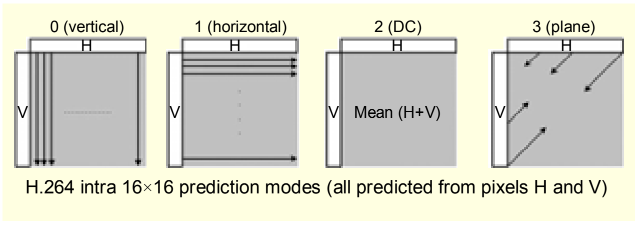

0 (vertical)
1 (horizontal)
2 (DC)
3 (plane)
H
V
Mean (H+V)
H.264 intra 16×16 prediction modes (all predicted from pixels H and V)

Fig. 3. 16 × 16 intra prediction in four modes.

four modes that are calculated for a 16×16 block and is shown in Fig. 3.

3. 8 × 8 Intra Prediction

Each 8×8 intra prediction component is predicted from previously encoded chroma samples above and/or to the left, and both Cb, Cr components always use the same prediction mode. The four prediction modes for 8×8 prediction are very similar to the 16×16 intra prediction modes. As the prediction mode decision method for 8×8 intra prediction is similar to the16×16 prediction mode, a 16×16 intra prediction module is used for 8×8 intra prediction for our implementation.

4. TQ/IQIT Operation and Cycle Overhead for I16MB Mode

Data within a macroblock are transmitted in the order shown

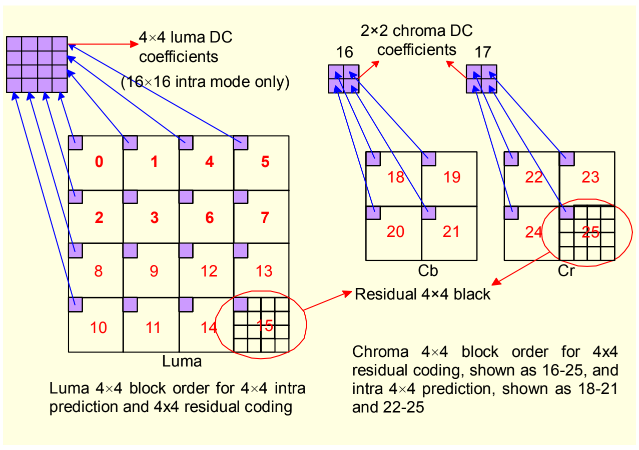

4×4 luma DC coefficients (16×16 intra mode only)
0 1 4 5 2 3 6 7 8 9 12 13 10 11 14 15
Luma
Luma 4×4 block order for 4×4 intra prediction and 4x4 residual coding
2×2 chroma DC coefficients
16
17
18 19 20 21
22 23 24 25
Cb
Cr
Residual 4×4 black
Chroma 4×4 block order for 4x4 residual coding, shown as 16-25, and intra 4×4 prediction, shown as 18-21 and 22-25

Fig. 4. Scanning order of residual blocks within a macroblock.

in Fig. 4. If the macroblock is coded in 16×16 intra mode (I16MB), then the block labeled “-1” is transmitted first, containing the DC coefficients of each 4×4 luma block. Next, the luma residual blocks 0 to 15 are transmitted in the order shown (with the DC coefficients set to zeros in a 16×16 intra macroblock). Blocks 16 and 17 contain a 2×2 array of DC coefficients from the Cb and Cr chroma components, respectively. Finally, chroma residual blocks 18 to 25 (with zero DC coefficients) are sent. If the macroblock is not coded in 16×16 intra mode, then the block labeled “-1” is not transmitted, and the luma residual blocks 0 to 15 are transmitted in the order shown (with the DC coefficients). The chroma coefficients are sent with the same methodology as in 16×16 intra mode.

Figure 5 shows the timing cycle when intra 16×16 mode is selected. This is our first version for the macroblock processing cycle. The operation cycle of IQIT can start after 769 cycles, and the total number of cycles for the I16MB processing is 1105. Each cycle as given in Fig. 5 is explained in the section III.

When in intra 16×16 mode, the transformation method uses both 4×4 integer DCT and a Hadamard transform. To code a macroblock in I16MB mode, 16 blocks have to be 4×4 integer DCT transformed and quantized, and the DC values obtained from a 4×4 integer DCT have to be Hadamard transformed and quantized. Since the DC coefficients of DCT-transformed data can be obtained after a DCT operation of 16 blocks, IQIT operation can start after completion of the Hadamard transform and quantization of DC coefficients (start of IQIT@I16MB). This fact makes the 178-cycle overhead for I16MB mode compared with non-I16MB mode. For the encoding cycle of non-I16MB mode (I4MB, INTER), the operation of IQIT can

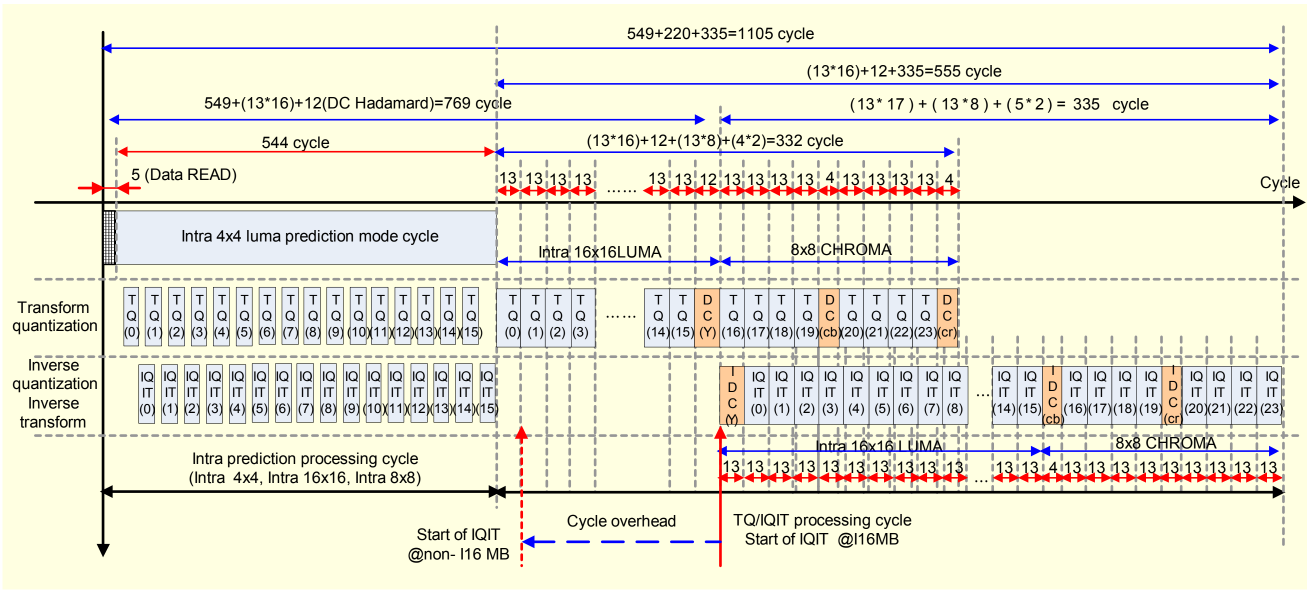

549+220+335=1105 cycle
(13*16)+12+335=555 cycle
549+(13*16)+12(DC Hadamard)=769 cycle
(13* 17) + (13 *8) + (5 * 2) = 335 cycle
544 cycle
(13*16)+12+(13*8)+(4*2)=332 cycle
5 (Data READ)
Cycle
Intra 4x4 luma prediction mode cycle
Intra 16x16LUMA
8x8 CHROMA
Transform quantization
Inverse quantization Inverse transform
TQ(0) … TQ(15)
DC(Y) DC(cb) DC(cr)
IQIT(0) … IQIT(23)
IDC(Y)
Intra prediction processing cycle (Intra 4x4, Intra 16x16, Intra 8x8)
Start of IQIT @non- I16 MB
Cycle overhead
TQ/IQIT processing cycle Start of IQIT @I16MB
Intra 16x16 LUMA
8x8 CHROMA

Fig. 5. The cycle overhead for I16MB mode.

start at the point "start of IQIT @non-I16MB." To reduce the cycle overhead in I16MB mode, we have to move the starting point of IQIT to "start of IQIT @non-I16MB." If we can predict the quantized value of Hadamard transformed DC coefficients, we can reduce the cycle overhead for I16MB mode.

The distortion calculation method for 16×16 prediction uses the same method to code an I16MB macroblock, but while calculating the distortion, the distortion measure is the sum of absolute error (SAE) value of 2D-Hadamard transformed results of the prediction error (the difference between coded pixel and predicted pel).

Here, we will verify that DC values of the 2D-Hadamard transform and integer 2D-discrete cosine transform are basically the same. From this fact, we will confirm that a Hadamard DC coefficient can be predicted in the 16×16 intra prediction process. During the 16×16 intra prediction process, the value of a DC coefficient can be obtained by modifying the intra prediction module.

A. 4×4 Residual Luma and Chroma Forward Integer Transform

In H.264, a 4×4 forward DCT transform is modified [8] as

$$Y = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right). \quad (1)$$

From (1), we can obtain the DC value Y_{00} as shown in (2):

$$\begin{aligned} Y_{00} = (&x_{00} + x_{01} + x_{02} + x_{03} + x_{10} + x_{11} + x_{12} + x_{13} \\ &+ x_{20} + x_{21} + x_{22} + x_{23} + x_{30} + x_{31} + x_{32} + x_{33}). \end{aligned} \quad (2)$$

B. Hadamard Transform

The 4×4 luma DC coefficients of a 16×16 block are grouped into a 4×4 block and further transformed, for intra frames, to improve compression.

The input matrix X is formed by picking out DC coefficients from the 16 transformed 4 × 4 blocks. DC coefficients are then transformed using a symmetric Hadamard transform in the form

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2. \quad (3)$$

From (3), the DC value Y_{00} of a 2D-Hadamard transform can be obtained as

$$\begin{aligned} Y_{00} = (&x_{00} + x_{01} + x_{02} + x_{03} + x_{10} + x_{11} + x_{12} + x_{13} \\ &+ x_{20} + x_{21} + x_{22} + x_{23} + x_{30} + x_{31} + x_{32} + x_{33}) / 2. \end{aligned} \quad (4)$$

From this, it is clear that the DC values of a Hadamard transform and integer DCT are basically the same. As we use the Hadamard distortion measure for the intra prediction process, the DC value of 16 block integer DCT can be obtained during the distortion calculating process.

As the DC values obtained from the Hadamard transform and 4×4 integer transform are basically the same, we can obtain the quantized DC coefficients by collecting the DC values of the selected I16MB mode among the vertical, horizontal, DC, and plane modes, and by applying a Hadamard transform followed by a quantization step. Therefore, we propose a hardware architecture for the 16×16 prediction module and quantization module, which can obtain quantized DC coefficients. This is discussed in section III. The start of an IQIT operation in the case of I16MB is performed at the point "start of IQIT @non-I16MB ."

5. Processing of Inter Macroblock and Coded Block Pattern Consideration

For the 4×4 intra prediction mode or inter prediction mode, a Hadamard transform of the luma component is not required. Here, the operation of IQIT can start after the completion cycle of the first 4×4 block TQ. But in the inter prediction mode, the coeff_cost value, which is used for coded block pattern (CBP) calculation, has to be considered. The CBP processing method, considering the coefficient cost, is incorporated into JM to improve coding efficiency for inter prediction and chroma components.

$$cbp_{8\times 8block} = \left(\left(\sum_{i=0}^{3}\sum_{j=0}^{15} c(i,j)\right) \le 4\right) ? 0 : 1 , \quad (5)$$

where $x ? y : z$ returns y if x is true and z if x is false.

$$c(i,j) = \begin{cases} \infty & \text{if } \ coeff(i,j) > 1 \\ 3 & \text{if } \ j = 0 \ \text{ and } \ coeff(i,j) = 1 \\ 2 & \text{if } \ 1 \le j \le 2 \ \text{ and } \ coeff(i,j) = 1 \\ 1 & \text{if } \ 3 \le j \le 5 \ \text{ and } \ coeff(i,j) = 1 \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

where j means a zero run value.

$$Coeff\_C_{all} = \sum_{block8\times 8=0}^{3} \left\{ \left(\sum_{i=0}^{3}\sum_{j=0}^{15} c(i,j)\right) \le 4\right) ? 0 : \sum_{i=0}^{3}\sum_{j=0}^{15} c(i,j) \right\}, \quad (7)$$

$$Cbp_{luma} = 0 \quad \text{if} \quad Coeff\_C_{all} \leq 5. \qquad (8)$$

Equations (5) and (6) indicate that when the quantized coefficients (level) of four 4×4 blocks are -1, 0, 1, and the summation of coeff_cost is less than 4, this 8×8 block is not coded in the bitstream. In this case, the reconstructed data for the 8×8 block is the prediction data. This means that the IDCT value of the 8×8 block has to be processed to all zeros. Equations (7) and (8) indicate that when the conditional sum of coefficient costs for four 8×8 blocks is less than 5, all of the luma components are not coded in the bitstream. For this reason, if we start an operation of IQIT at the "start of IQIT@non-I16MB", we cannot obtain a correct reconstructed frame. But in the real architecture, we start the IQ operation at the "start of IQIT@non-I16MB", and use the temporal storage memory 'IDCT value SRAM' to store the IDCT results. After determination of the CBP, the selected value from the IDCT value and 0s according to the CBP will be fed into the summation logic. Hence, there is no cycle overhead in inter macroblock.

III. Proposed Hardware Architecture

Figure 6 shows the hardware architecture of intra prediction, TQ/IQIT, and mode decision blocks. As the intra prediction module intrinsically uses neighboring pixels to predict the value of the current pixel, the neighboring pixels have to be stored. This pixel is stored in intra prediction SRAM as shown in Fig. 6. After the mode decision of the macroblock, this pixel is updated using the summated data of the IDCT value and prediction data for the chosen mode. These values can be determined only after one macroblock processing. But in the case of 4×4 intra prediction, the neighboring pixels for a 4×4 block depend on the block number position as shown in Fig. 4. To get the left neighboring pixels of block number 1, block 0 has to be processed using the subtraction operation, TQ/IQIT, and summation operation. Thus, before the determination of the macroblock mode, a TQ/IQIT operation has to be performed. That is, for the 4×4 block prediction mode, because there are sixteen 4×4 blocks in a macroblock, the predictor can't obtain the reconstructed pixels (unfiltered) when the previous blocks are not coded. Therefore, the TQ/IQIT operation is required for calculating the distortion (rd_cost) values of intra 4×4 prediction. Thus, we devised the intra prediction flip-flops to store the prediction data for 4×4 blocks. During the 4×4 prediction, neighboring pixels for each 4×4 block have to be updated in the prediction flip-flops. That is, intra prediction flip-flops are updated with summation data of the IDCT value and prediction value during the 4×4 prediction mode, whereas intra prediction RAM is updated after the decision of macroblock type.

As mentioned before, the 16×16 prediction and 8×8

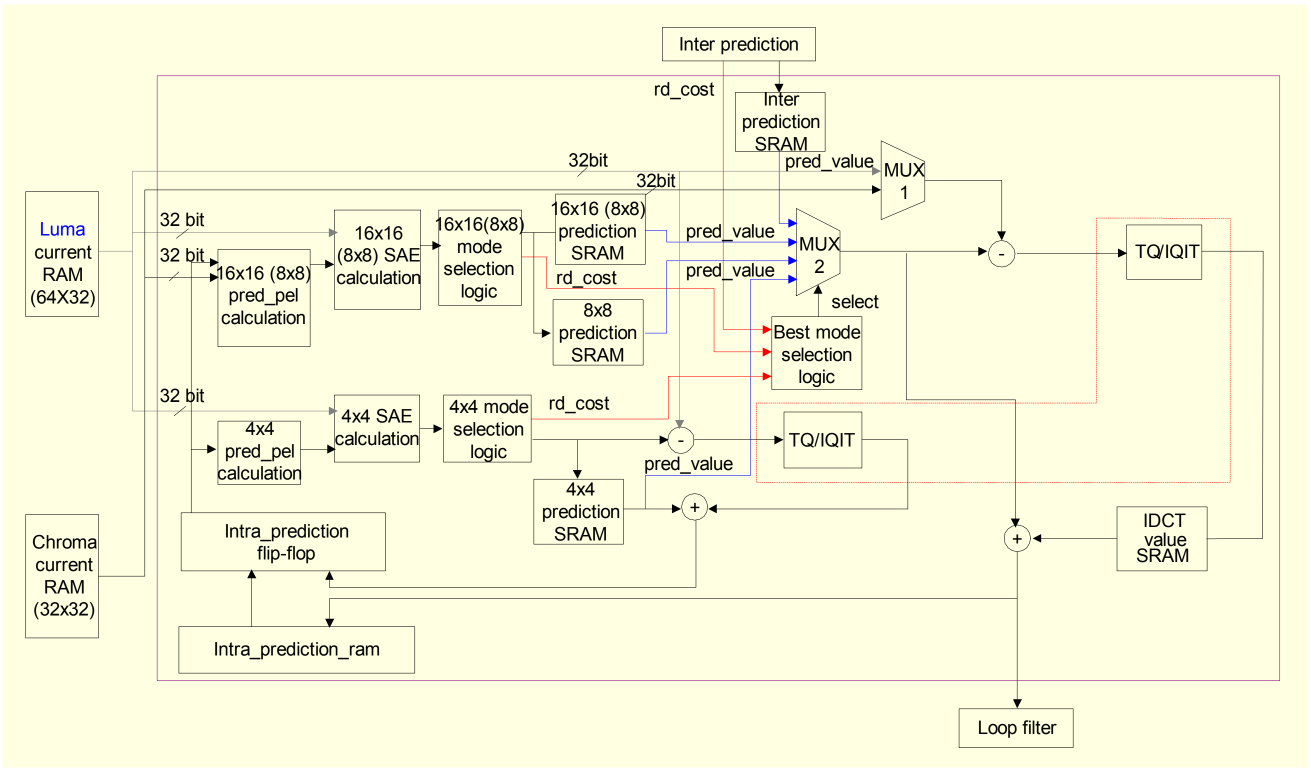


Fig. 6. Intra prediction and TQ/IQIT block diagram.

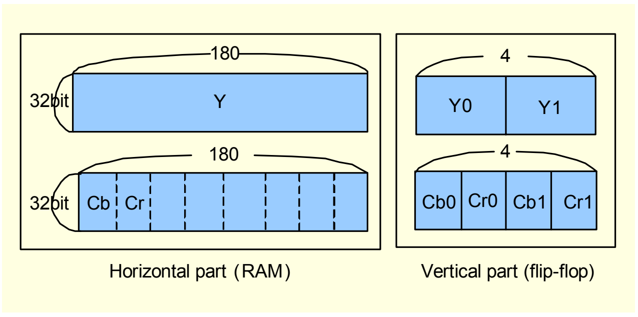


Fig. 7. Intra prediction RAM memory.

prediction use a similar method for prediction; therefore, we use a design for the 16×16 prediction module that can also be used for the 8×8 prediction module.

The neighboring pixel values for each macroblock in the prediction RAM are transferred to the prediction flip-flops before the start of intra-prediction for each macroblock.

In Fig. 6, the TQ/IQIT module is shown twice. One is for 4×4 prediction timing and the other is for timing after the macroblock mode decision. But in a real application, only one TQ/IQIT module is implemented because the operation timing of two TQ/ IQITs is different.

The memory architecture of intra prediction RAM is shown in Fig. 7. The prediction RAM is separated into horizontal and vertical parts. Each part represents the neighboring pixel of the macroblock boundary. Because the application of our encoder is for D1 resolution, we use 32 bit × 180 (720 pixel) horizontal SRAM for both luminance data and chrominance data. As 16 vertical prediction data are needed for luminance data, we use a vertical prediction 128-bit flip-flops for the vertical part. For the chrominance data, eight pixels are required for vertical prediction data of both Cb and Cr.

First, the intra 4×4 prediction mode is dealt here. Using the reconstructed pixel value of the intra prediction flip-flops, the 4×4 prediction pel calculation module computes the prediction values of each mode. Using the current pixel and the prediction values, the 4×4 SAE calculation module outputs the SAE values of 9 modes. By comparing the rd_cost (SAE+λ·rate) value of 9 modes, 4×4 mode selection logic selects the best mode with the minimum rd_cost and outputs the prediction values to be fed into the 4×4 prediction SRAM and TQ. By using the same method, 16×16 and 8×8 intra prediction outputs each selected values after choosing the best mode.

Each prediction value of the best mode selected by 16×16 and 4×4 mode selection logic are stored in 16×16 and 4×4 prediction SRAMs, each having a size of 256 bytes. The prediction values of the best mode selected in the case of 8×8 prediction are stored in 8×8 prediction SRAM, which is 128 bytes in size. Meanwhile, the prediction values of the best mode selected by the inter prediction module have to be stored in inter prediction SRAM, which is 384 bytes in size. The rd_cost value of inter prediction, which is calculated by the motion estimation module, is transferred to this module using AMBA (Advanced Microcontroller Bus Architecture) bus.

The best mode selection logic chooses the best mode among the 4×4 intra prediction, 16×16 intra prediction, and inter prediction with the best rd_cost calculated by each mode selection logic.

Using the selected mode from the best mode selection logic, MUX2 selects the stored values in 16×16, 4×4, and 8×8 inter prediction SRAM and transfers the values to the subtraction logic.

Output values of MUX2 have to be subtracted from the current value, and its subtracted values are transferred to the TQ module. After the IQIT operation is complete for each DPCMed value, the addition of the output values of IQIT and the predicted values in selected prediction SRAM is done, and its added values are stored in intra prediction RAM and a 384-byte sized buffer of the loop filter. The IDCT value SRAM has a size of 256 × 16 bits and is used for storing IDCT results for luma components for the inter prediction macroblock. For the inter prediction macroblock, the coefficient cost value has to be considered for the reconstruction of a macroblock. Until the TQ operation for 16 blocks is completed, the cbp (coded block pattern) value will not be determined. But in our architecture, because the IQIT operation starts after the first TQ cycle in order to reduce the cycle of one macroblock cycle, the reconstruction of a pixel will be mismatched. So, IDCT values are stored in the RAM and are used for summation after the 16 blocks' TQ operation considering the cbp_{luma} value.

1. Intra Prediction Flip-Flop Architecture

For intra prediction processing for one macroblock, 37 neighboring pixels are required for the luminance component and 34 neighboring pixels are required for the chrominance components. Figure 8 shows the required pixels for the luminance component. For the horizontal direction, 21 neighboring pixels are required because the intra 4×4 prediction needs an M value and 16 pixels in the upper side and 4 pixels on the right upper side. For the upper side of the chroma components, nine upper neighboring pixels are required for each component. For the vertical direction, 16 left neighboring pixels are needed for the luma components, and eight left neighboring pixels are needed for Cb and Cr, respectively.

Thus, we map these neighboring pixels into intra prediction

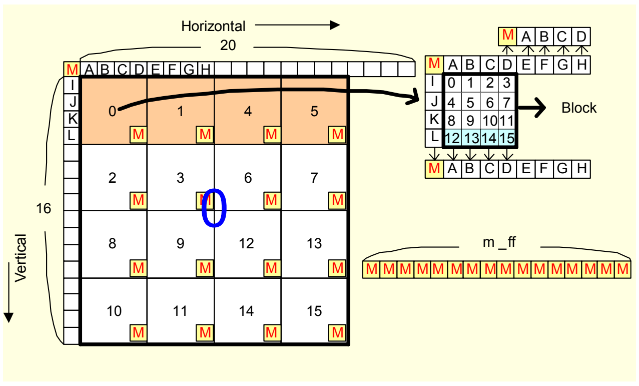


Fig. 8. Intra prediction flip-flops.

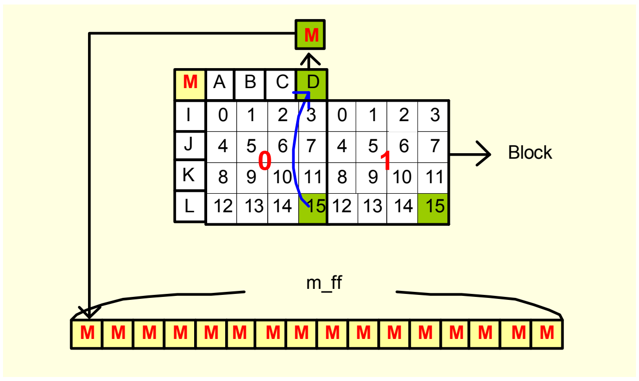


Fig. 9. M flip-flops.

flip-flops. These flip-flops have to be read from the intra prediction SRAM before the start of a macroblock and are updated during the operation of 4×4 intra prediction. The intra 4×4 prediction block executes a prediction operation after reading eight pixel values in the horizontal part and four pixel values in the vertical part, corresponding to the position of the current block.

In the case of 4×4 intra prediction, the result of adding IDCT and prediction values has to be stored in the intra prediction flip-flops for the prediction of the neighboring pixel of the next block.

In the case of the vertical neighboring pixel, the 3rd, 7th, 11th, and 15th pixel data of one block are stored in the vertical prediction flip-flops, and for the horizontal neighboring pixel, the 12th, 13th, 14th, and 15th pixel data are stored in the horizontal prediction flip-flops. These stored pixel values in the flip-flops are used for the prediction of the next block. Figure 9 shows the M flip-flops. The M flip-flops indicate the upper-left pixels which are only used for 4×4 intra prediction.

In Fig. 9, the M value of block 1 exists as D in the horizontal prediction flip-flop while block 0 is processing. But, after updating the flip-flop of position D with the value of the 15th pixel of block 0, the D pixel data value no longer exists because of overwriting.

Here, before updating the flip-flop of position D with the 15th pixel, the value of D is transferred to the M flip-flop with index 0. Because there are 16 blocks in macroblock luminance components, 16 storage positions are needed.

Table 1 shows the read index of M for each block. When block 1 is processing, the M value of block 1 is read from the M flip-flop with index 0. From Fig. 9, we can see that the M value of block 1 is the D value of block 0.

By introducing the intra prediction flip-flops and M flip-flops, the 4×4 prediction process can be done without memory access. Therefore, we can access the neighboring pixel at any time to generate the predicted pels for 9 modes. Since the neighboring pixel finding method for JM is different from our method, we modified the neighboring pixel finding method described above in our reference software and confirmed the correctness of our method.

Table 1. Read index of M for each block.

Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Index of M	5	0	7	2	1	4	3	6	13	8	15	10	9	12	11	14

2. 4×4 Intra Prediction Hardware Architecture

The 4×4 intra prediction module consists of the prediction pel calculation module, 4×4 SAE calculation module, and 4×4 mode selection logic, as shown in Fig. 10.

Using the pixels read from the intra prediction flip-flops, the prediction pel calculation module generates 36 pixel values simultaneously for 9 mode. In the 4×4 SAE calculation module, the differences of the current pixel and prediction pel value are computed, and their computed differences are Hadamard-transformed.

This module computes the SAE value by addition of the absolute value of the transformed difference. Table 2 shows the computed SAE calculation time cycle using the 2D-Hadamard transform. The SAE calculation time takes eleven cycles because of the parallel processing of 4 pixel data. The 4×4 mode selection logic finds the best mode with the smallest rd_cost among the 9 mode. The rd_cost is calculated by addition of the calculated value of 4×4 SAE and the value in proportion to the amount of bits to be generated for the syntax element in intra 4×4 pred_mode.

Figure 11 shows the 4×4 intra prediction processing cycle of a 4×4 block. As shown in Fig., 1 the cycle is required to read the pixel from the luma current RAM and to calculate the

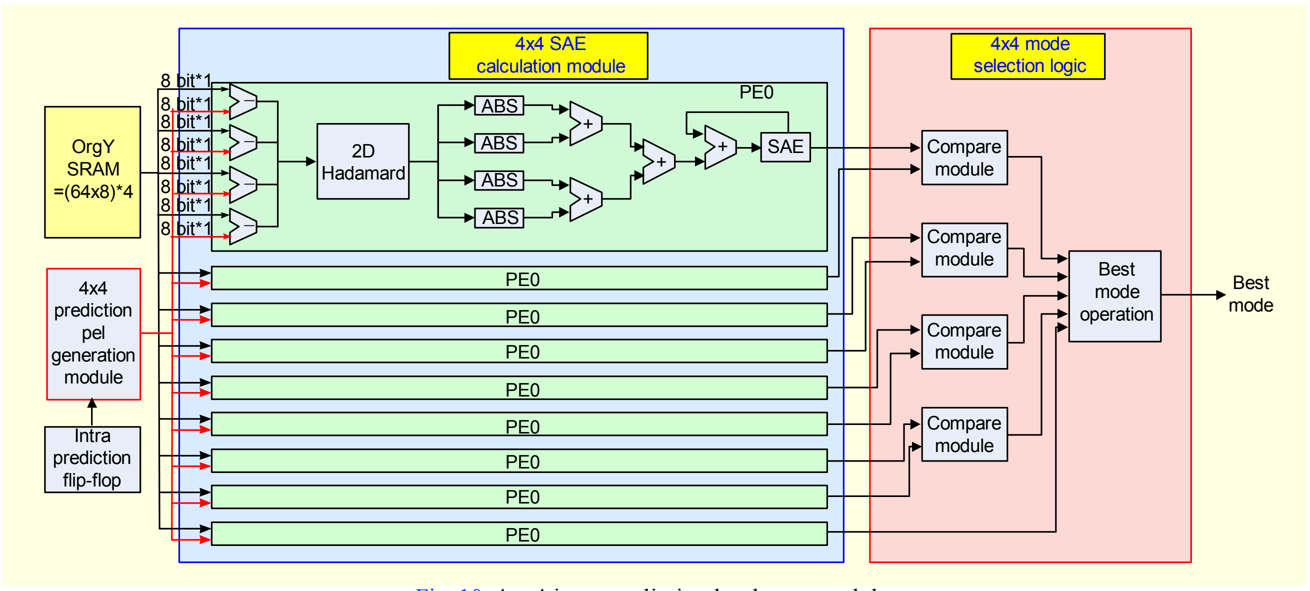


Fig. 10. 4 × 4 intra prediction hardware module.

Table 2. Hadamard distortion calculation time.

Count	Address	Hadamard input	Transpose register file	Hadamard output	Org-pred	TQ	IQIT
1	(0, 1, 2, 3)						
2	(4, 5, 6, 7)	(0, 1, 2, 3)					
3	(8, 9, 10, 11)	(4, 5, 6, 7)	(0, 1, 2, 3)				
4	(12, 13, 14, 15)	(8, 9, 10, 11)	(4, 5, 6, 7)				
5		(12, 13, 14, 15)	(8, 9, 10, 11)				
6			(12, 13, 14, 15)				
7		(0, 4, 8, 12)					
8		(1, 5, 9, 13)		(0, 4, 8, 12)			
9		(2, 6, 10, 14)		(1, 5, 9, 13)			
10		(3, 7, 11, 15)		(2, 6, 10, 14)			
11				(3, 7, 11, 15)			
12				Cal_rdcost 4×4			
13				Mode_sel			
14				Prediction 4×4 SRAM write	(0, 1, 2, 3)		
15					(4, 5, 6, 7)	TQ operation	
16					(8, 9, 10, 11)		
17					(12, 13, 14, 15)		
18							
19							
20							
21							
22							
23							IQIT operation
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							

prediction value of each mode.

The SAE value that has been Hadamard transformed takes eleven cycles. The selection of the best mode takes one cycle.

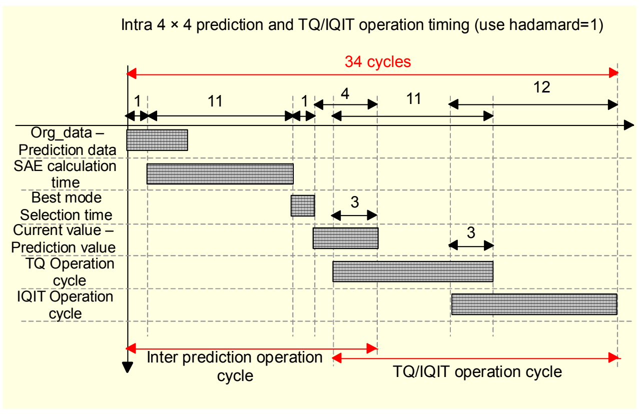


Fig. 11. Intra 4 × 4 prediction hardware cycles.

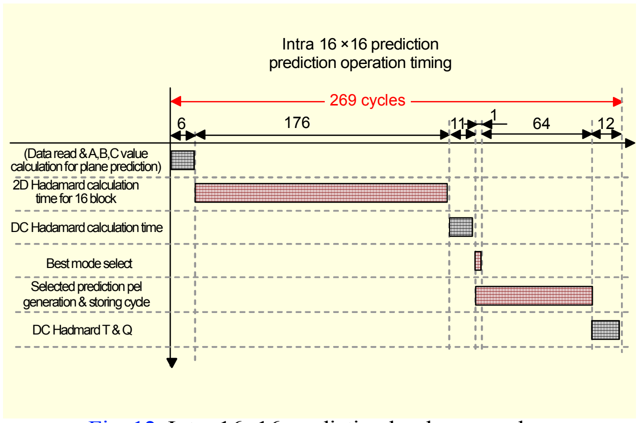


Fig. 12. Intra 16×16 prediction hardware cycles.

Also, it takes one cycle to compute the difference of the current values and prediction value. Thirteen cycles are required for the transformation and quantization cycle. For the inverse quantization and inverse transform case, thirteen cycles are needed. We will explain each case in sub-sections.

So, since 34 cycles are needed for 4×4 intra prediction, and as one macroblock has sixteen blocks, the total prediction takes 544 cycles. Since the prediction flip-flops have to be updated before the start of intra 4×4 prediction, five cycles are required to read 20 pixels in the upper horizontal SRAM. So, there are a total of 549 cycles for 4×4 prediction.

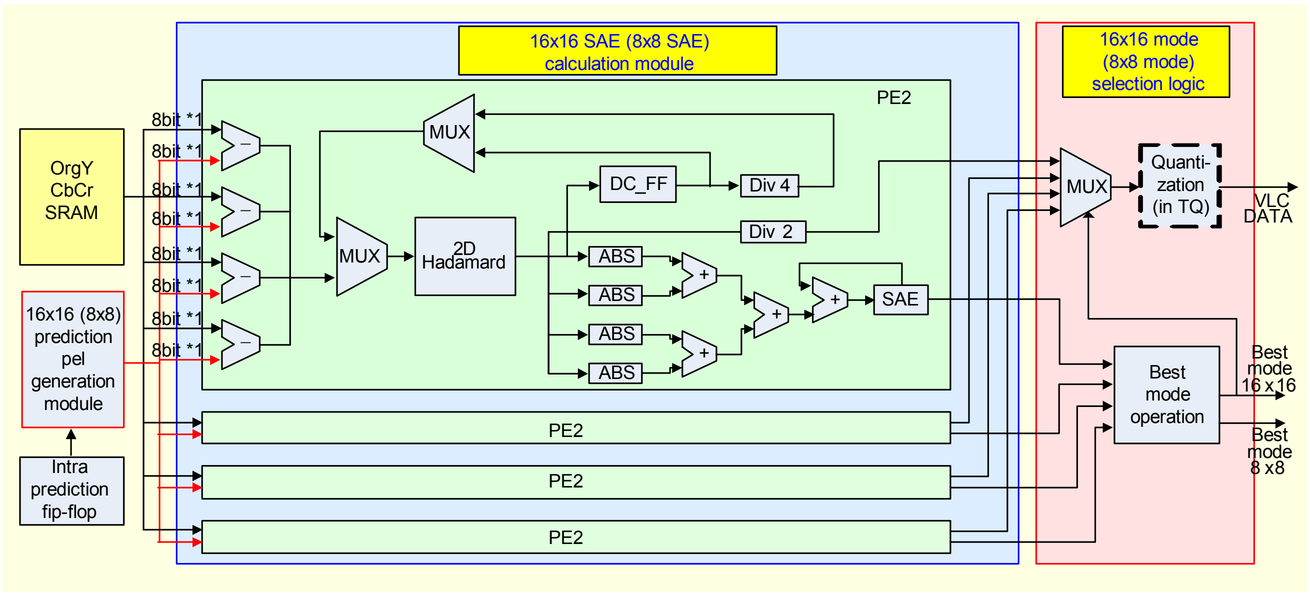
3. 16×16 (8×8) Intra Prediction Hardware Architecture

The 16×16 intra prediction is similar to the 4×4 intra prediction in its architecture and processing method, but it has a different mode and processing timing cycle.

Referring to Fig. 12, six cycles are required to read the neighboring data (4 cycles) and generate A,B,C values for the plane prediction (2 cycles) of 16×16 prediction.

The SAE value that has to be Hadamard transformed takes187 cycles, which can be calculated by 17 (16 block +DC)* 11. The selection of the best mode requires one cycle, and the selected prediction mode pel transferring 16×16 prediction SRAM requires 64 cycles. So, there is a nominal total of 257 cycles for 16×16 intra prediction for one macroblock.

To reduce the number of processing cycles for one macroblock, we predicted the quantized value of Hadamard-transformed DC coefficients in the prediction process. Figure 13 shows the 16×16 intra prediction module architecture,



16x16 SAE (8x8 SAE) calculation module
16x16 mode (8x8 mode) selection logic
PE2
OrgY CbCr SRAM
16x16 (8x8) prediction pel generation module
Intra prediction fip-flop
8bit *1
MUX
DC_FF
Div 4
Div 2
2D Hadamard
ABS
SAE
Quanti-zation (in TQ)
VLC DATA
Best mode operation
Best mode 16 x16
Best mode 8 x8

Fig. 13. 16 × 16 intra prediction module integrating Hadamard DC transform.

Table 3. Timing cycle for 4 × 4 DC coefficients block.

Cycle	Hadamard input	Hadamard f/f enable	DC coeff. register file	Quantization	Ext RAM
0	DC0, DC1, DC4, DC5				
1	DC2, DC3, DC6, DC7	En	DC0, DC1, DC4, DC5		
2	DC8, DC9, DC12, DC13	En	DC2, DC3, DC6, DC7		
3	DC10, DC11, DC14, DC15	En	DC8, DC9, DC12, DC13		
4		En	DC10, DC11, DC14, DC15		
5	DC0, DC2, DC8, DC10				
6	DC1, DC3, DC9, DC11	En			
7	DC4, DC6, DC12, DC14	En		DC0, DC2, DC8, DC10	
8	DC5, DC7, DC13, DC15	En		DC1, DC3, DC9, DC11	DC0, DC2, DC8, DC10
9		En		DC4, DC6, DC12, DC14	DC1, DC3, DC9, DC11
10				DC5, DC7, DC13, DC15	DC4, DC6, DC12, DC14
11					DC5, DC7, DC13, DC15

which can predict 2D-Hadamard DC transform results. Since we use a Hadamard distortion measure for intra prediction, each process element (PE2) has a 2D-Hadamard transform and DC_FF (sixteen Hadamard-transformed DC coefficients). During the calculation of distortion for the DC Hadamard calculation time, DC coefficients divided by 4 are fed into the 2D Hadamard.

After mode selection of 16×16 prediction, each DC_FF retains sixteen Hadamard-transformed DC coefficients for each mode. Therefore, if we apply the Hadamard transform to these coefficients directly and select its output by the best selected mode, we can get the 2D-Hadamard results of DC coefficients. Since the quantization module is in the TQ module, the output results are fed into the TQ module.

Table 3 shows the data processing cycle of a Hadamard transform and quantization for luma DC coefficients. Since there are twelve processing cycles for the 4×4 DC coefficients block as shown in Table 3, the 2D Hadamard transformation and quantization step is processed in the last twelve cycles as shown in Fig. 12.

As mentioned before, the 8×8 intra prediction mode is very similar for 16×16 intra prediction except for the size of the prediction. So the 8×8 prediction mode is processed in the 16×16 prediction module. It uses same logic for SAE calculation and best mode selection logic. There are a total of 122 cycles for 8×8 prediction including neighboring pixel reading, SAE calculation, and prediction pel storing. Since the total number of cycles for processing 16×16 and 8×8 prediction is 397, which is less than the number of cycles required for processing 4×4 prediction, sharing logic can be accomplished.

4. TQ/IQIT Processing Timing

For the 4×4 block prediction mode, as there are sixteen 4×4 blocks in a macroblock, the predictor cannot obtain the reconstructed pixels (unfiltered) when the previous blocks are not coded. To make the reconstructed pixels, a TQ/IQIT operation is required for the 4×4 prediction mode. So the reduction in the number of operation cycles of TQ/IQIT is a crucial point in implementing the H.264 encoder. In the proposed architecture, we used four parallel input data sets and processed four data sets, simultaneously, to reduce the cycle of TQ/IQIT. For implementing the quantization module, four quantization modules are used for the parallel processing. Figure 14 shows the hardware architecture of a TQ module. As the 4×4 Hadamard transform and DC coefficient register have been moved to the intra 16×16 prediction module, there are no 1D-Hadamard and DC coefficient register files for luminance. In Fig. 14, the DC coefficient register represents the coefficient value for chroma components. In the TQ module, the CBP process block generates the coeff_cost and all_coeff_cost to be used for inter-prediction mode, and generates CBP and CBP_BLK information to be used for variable-length-coding and the deblocking filter, respectively.

For the quantization of DC Hadamard output, the data input from the 16×16 prediction module is selected by mux and transferred into the quantization module. The IQIT module has a similar architecture as TQ except for the inverse 4×4 Hadamard transform.

Table 4 shows the data processing cycle of the integer DCT and quantization module for a 4×4 block. The first cycle indicates the address generation cycle for the selected SRAM

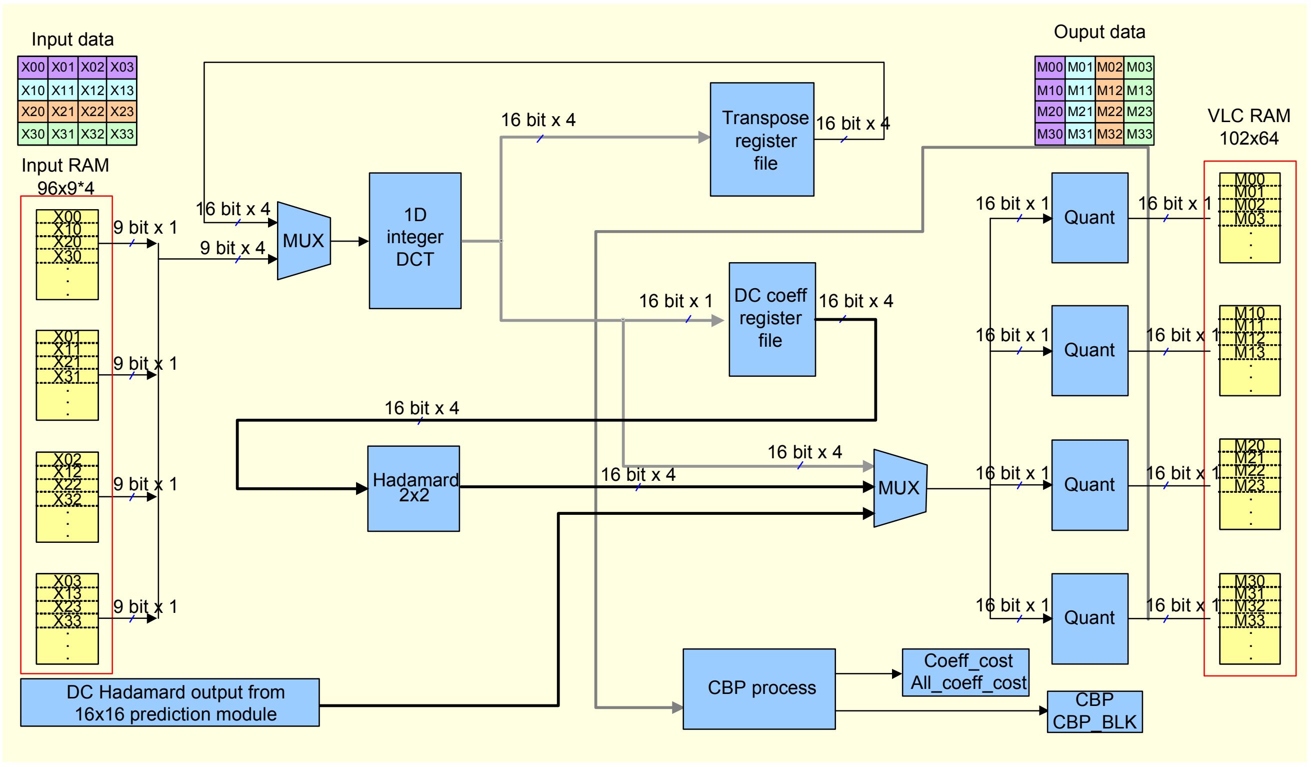


Fig. 14. TQ hardware architecture.

Table 4. Timing cycle for a 4 × 4 block.

Cycle	Address	Transform input	Transform f/f enable	Transform register file	Quantizatiom	Ext RAM
0	X00, X01, X02, X03					
1	X10, X11, X12, X13	X00, X01, X02, X03				
2	X20, X21, X22, X23	X10, X11, X12, X13	En	X00, X01, X02, X03		
3	X30, X31, X32, X33	X20, X21, X22, X23	En	X10, X11, X12, X13		
4		X30, X31, X32, X33	En	X20, X21, X22, X23		
5			En	X30, X31, X32, X33		
6		M00, M10, M20, M30				
7		M01, M11, M21, M31	En			
8		M02, M12, M22, M32	En		M00, M10, M20, M30	
9		M03, M13, M23, M33	En		M01, M11, M21, M31	M00, M10, M20, M30
10			En		M02, M12, M22, M32	M01, M11, M21, M31
11					M03, M13, M23, M33	M02, M12, M22, M32
12						M03, M13, M23, M33

(among the 16×16 prediction SRAM, 4×4 prediction SRAM, inter prediction SRAM, and 8×8 prediction SRAM) according to the chosen macroblock mode.

From cycles 1 to 4, the 4×4 block is 1D-transformed, and from cycles 6 to 9, the second 1D-integer DCT is applied. A quantization process is applied one clock after the output of 2D DCT-transformed data. There is a total of 13 cycles for a 4×4 block. Table 5 shows the data processing cycle of 2×2 chroma Hadamard transformations. A 2×2 Hadamard transformation requires four cycles.

Table 5. Timing cycle for 2 × 2 DC coefficients block.

Cycle	Hadamard input	Hadamard f/f enable	Quantization	Ext RAM
0	DC16, DC17, DC18, DC19			
1		En		
2			DC16, DC18, DC17, DC19	
3				DC16, DC18, DC17, DC19

5. The Number of Cycles for One Macroblock Processing in an H.264 Encoder

For the processing of the D1 resolution (720×480×30 frame) encoder, the processing time of one macroblock is calculated into about 1300 cycles when the operating frequency of the encoder is 54 MHz.

In H.264, there are two types of prediction modes for an intra macroblock, 4×4 prediction mode and 16×16 prediction mode. For the calculation of the rd_cost of intra 4×4 prediction, the TQ/IQIT has to be operated before the selection of the prediction mode (inter, intra16×16, intra 4×4).

For the rd_cost calculation of the other prediction modes except for the 4×4 intra prediction mode, we need no TQ/IQIT operation. After the rd_cost calculation of each prediction mode, the mode-select logic selects the mode that has the smallest value of rd_cost. After the selection of the prediction mode, the TQ/IQIT module can start the operation. As the 4×4 block prediction process is the longest cycle for the intra prediction, 549 cycles are needed for the intra prediction processing cycle.

After the selection of prediction mode, TQ and IQIT operations have to be performed. The allowable number of cycles of TQ and IQIT operations for one macroblock is 751, which is derived from 1300 cycles minus 549 cycles. Figure 15 shows one macroblock processing cycle using the prediction method as described before. As the quantized DC Hadamard value is pre-calculated in the 16×16 prediction process, the inverse Hadamard and quantization for the quantized DC Hadamard value of the I16MB mode can be performed in the same cycle of the first TQ cycle. So we obtain a total of 927 cycles for one macroblock processing and reduce it by 178 cycles compared with the original number of timing cycles as shown in Fig. 5. Since the reducing method of overhead for I16MB is generic, it can be applied to other architectures. If we apply this method to the DCT-based mode decision method in [11], we can reduce this architecture by approximately 64 cycles .

IV. Experimental Results

For experimental verification, we made a C-language reference model of hardware modified from the JM8.5 [13]. We compared the output results of our reference-C model with that of JM 8.5 model, and confirmed the correctness of our model. From the reference model, we extracted the test vector for the intra prediction, TQ/IQIT, and mode decision module and confirmed the correctness of each hardware module using Mentor Graphics ModelSim. The proposed architecture was

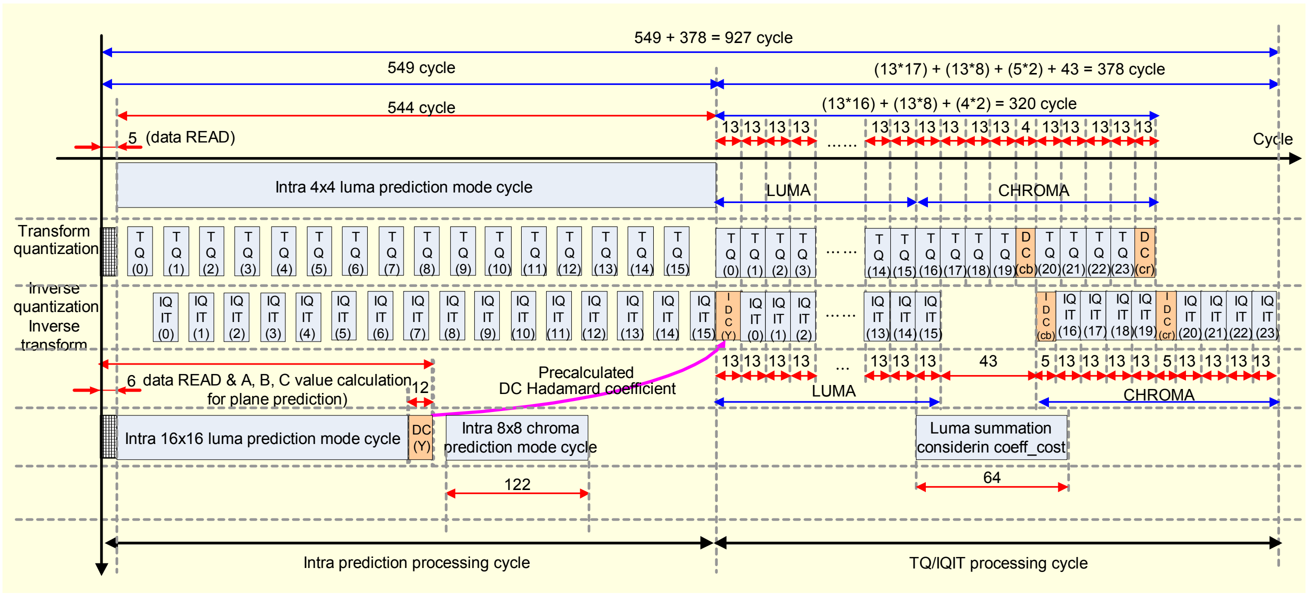

549 + 378 = 927 cycle
549 cycle
(13*17) + (13*8) + (5*2) + 43 = 378 cycle
544 cycle
(13*16) + (13*8) + (4*2) = 320 cycle
5 (data READ)
Cycle
Intra 4x4 luma prediction mode cycle
LUMA
CHROMA
Transform quantization
Inverse quantization Inverse transform
6 data READ & A, B, C value calculation for plane prediction)
12
Precalculated DC Hadamard coefficient
43
Intra 16x16 luma prediction mode cycle
DC (Y)
Intra 8x8 chroma prediction mode cycle
Luma summation considerin coeff_cost
122
64
Intra prediction processing cycle
TQ/IQIT processing cycle

Fig. 15. Reduction of overhead cycle for I16MB mode using the pre-calculated Hadamard coefficient.

Table 6. RAM sizes used.

RAM	D/W	Size (bit)	Pixel
Luma current RAM	64×32 bit	2,048	256
Chroma current RAM	32×32 bit	1,024	128
4×4 prediction RAM	64×32 bit	2,048	256
16×16 prediction RAM	64×32 bit	2,048	256
8×8 prediction RAM	32×32 bit	1,024	128
IDCT_value SRAM	64×64 bit	4,096	
Inter prediction RAM	96×32 bit	3,072	384
Intra prediction RAM(Y)	180×32 bit	5,760	720
Intra prediction RAM(Cb,Cr)	180×32 bit	5,760	720
Most probable mode RAM	180×4 bit	720	
Sum		27,600	

Table 7. Synthesized gate.

Blocks	Gate
4×4 intra prediction	74,955 gate
16×16 (8×8) intra prediction	39,058 gate
TQ/IQIT	70,321 gate
Others	8,102 gate
Sum	192,436 gate

Table 8. Comparison results with the method of Huang and others.

Architecture	MB pipelining [11]	MB pipelining and DCT-based decision [11]	Our method (dedicated)
Decision method	Hadamard	DCT-based	Hadamard
Cycle/MB	1280	1280	927
Required frequency	52.7 MHz	52.7 MHz	38.17MHz
Inter slice support	No	No	Yes
Required bus bandwidth	20Mbyte/s	20Mbyte/s	15.6Mbyte/s
Gate size	Unknown	74,000	192,436

designed with verilog HDL code and synthesized with Synopsys Design Compiler using the Hynix 0.35 μm TLM library, cb35os142d. Tables 6 and 7 summarize the RAM size and results of synthesis. In Table 6, the sizes of the last three SRAMs depend on the resolution of the picture to be encoded.

The differences from the previous work [11] are shown Table 8. Our results have a good point in the number of processing cycles needed, but have a larger gate size compared with the results of [11]. This is because [11] uses the DCT-based mode decision, reconfigurable PE, and interleaved I4MB/I16MB prediction schedule, whereas our architecture uses dedicated logic. But our architecture incorporates inter prediction processing and does not require the external SDRAM access to acquire the neighboring information, which can generate the memory bottleneck problem considering the large memory access cycle of motion estimation for inter prediction. The external SDRAM memory access of our architecture requires only the original 384 pixels to be coded for one macroblock.

V. Conclusion

In this paper, we propose an efficient architecture for H.264 intra prediction, transform and quantization (TQ) and inverse quantization and inverse transform (IQIT), and mode decision modules. It supports sum of absolute error (SAE) calculation using Hadamard and inter macroblock processing considering the coefficient cost. It can process one macroblock in 549 cycles for 4×4 intra prediction, which is the worst case method of intra prediction. Our main contribution is a reduction method of cycle overhead for I16MB mode. We predict the quantized value of Hadamard transformed DC coefficients in the prediction process, and use these values for IQIT in the cycle of the first TQ cycle. Therefore, we can reduce 178 cycles in our architecture. Our reduction method of cycle overhead in I16MB can be applied to other architectures. If we applied this method to the DCT-based mode decision method [11], we can estimate the reduction to be approximately 64 cycles. It can process one macroblock for 927 cycles for all cases of macroblock type, and it has the capability to process a D1 resolution picture at 42 frame/s for a 54 MHz clock. It can be directly extended to a 1280×720 resolution at 30 frame/s if we use a 108 MHz clock.

References

[1] ISO/IEC 14496-10 International Standard (ITU-T Rec. H.264).

[2] Seong-Min Kim, Ju-Hyun Park, etc. "Hardware-Software Implementation of MPEG-4 Video Codec," *ETRI J.*, vol.25, no.6, Dec. 2003, pp.489-502.

[3] Bong-Ho Lee, Kyu-Tae Yang, Young Kwon Hahm, Soo In Lee, and Chieteuk Ahn, "A Framework for MPEG-4 Contents Delivery over DMB," *ETRI J.*, vol.26, no.2, Apr. 2004, pp.112-121.

[4] Jae Hun Lee and Nam Suk Lee, "Variable Block Size Motion Estimation Algorithm and Its Hardware Architecture for H.264/AVC," *Proc. IEEE Int'l Symp. Circuits and Systems,* vol.3, 2004, pp.741-744.

[5] Yueh-Yi Wang, Yang-Tsung Peng, and Chun-Jen Tsai, "VLSI

Architecture Design of Motion Estimator and In-Loop Filter for MPEG-4 AVC/H.264 Encoders," *Proc. IEEE Int'l Symp. Circuits and Systems*, vol.2, 2004, pp.149-152.

[6] Yu-Wen Huang, Tu-Chih Wang, Bing-Yu Hsieh, and Liang-Gee Chen, "Hardware Architecture Design for Variable Block Size Motion Estimation in MPEG-4 AVC/JVT/ITU-T H.264," *Proc. IEEE Int'l Symp. Circuits and Systems,* 2003, pp.II-796-II-799.

[7] Tung-Chien Chen, Yu-Wen Huang, and Liang-Gee Chen; "Fully Utilized and Reusable Architecture for Fractional Motion Estimation of H.264/AVC," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* vol.5, 2004, pp.9-12.

[8] H.S. Malvar, A Hallapuro, M. Karczewicz, and Louis Kerosfsky, "Low Complexity Transform and Quantization in H.264/AVC," *IEEE Trans. CSVT*, vol.13, no.7, pp.598-603.

[9] Tu-Chih Wang, Yu-Wen Huang, Hung-Chi Fang and Liang-Gee Chen, "Parallel 4x4 2D Transform and Inverse Transform Architecture for MPEG-4 AVC/H.264," *Proc. IEEE Int'l Symp. Circuits and Systems*, vol.3 2003, pp.800-803.

[10] Yu-Wen Huang, Bing-Yu Hsieh, Tung-Chien Chen, and Liang-Gee Chen, "Hardware Architecture Design for H.264/AVC Intra Frame Coder," *Proc. ISCAS*, vol.2, 2004, pp.269-272.

[11] Yu-Wen Huang, Bing-Yu Hsieh, Tung-Chien Chen, and L.G Chen, "Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder," *IEEE Trans. Circuit and Systems for Video Technology,* vol.15, no.3, Mar. 2005, pp.378-401.

[12] Tung-Chien Chen, Yu-Wen Huang, and Liang-Gee Chen, "Analysis and Design of Macroblock Pipelining for H.264/AVC VLSI Architecture," *Proc. IEEE Int'l Symp. Circuits and Systems,* vol.2, 2004, pp.273-276.

[13] JVT H.264 Reference Software Version JM8.5, ftp://ftp.imtc-files.org/jvt-ecperts/

[14] T. C. Wang, Y. W. Huang, H. C. Fang, and L.G Chen, "Performance Analysis of Hardware Oriented Algorithm Modifications in H.264" *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, Hong-Kong, vol.2, April 2003, pp.493-496.

Kibum Suh received the BS, MS, PhD degrees in electronics engineering from Hanyang University in Seoul, Korea in 1989, 1991, and 2000. In 2000, he joined Electronics and Telecommunications Research Institute (ETRI) in Daejeon, Korea. He was engaged in the development of MPEG-4 ASIC design, image compression algorithms and VLSI architecture for video codecs. He is currently in the Department of Electronics at Woosong University in Daejeon, Korea. He is currently engaged in research on H.264 SOC design, image compression algorithms, and SOC architecture design.

Seongmo Park received the BS and MS degrees in electronics engineering from Kyungpook National University, Taegu, Korea, in 1985 and 1987. From 1987 to 1992, he was with LG Semiconductor Company, Gumi, Korea, where he worked on ASIC design and Mask ROM design. In 1992, he was with ETRI and joined in the development of ASIC design. Currently, he is working toward the PhD degree in electronics engineering from Kyungpook National University, Korea. He is also engaged in research on SOC design, image compression algorithms, and SOC architecture design. His main research interests are in video coding, image compression, and low power SOC architecture design.

Hanjin Cho was born in Seoul, Korea on July 8, 1960. He received the BS degree in electronic engineering from Hanyang University in 1982. He received the MS and PhD degrees in electrical engineering from New Jersey Institute of Technology in 1987, and from University of Florida in 1992, respectively. He joined ETRI in 1992, where he currently works in SOC design methodology development and wireless multimedia SOC design as a project manager.