J. Vis. Commun. Image R. 23 (2012) 245–253

Contents lists available at SciVerse ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Modification and implementation of an edge-based fast intra prediction mode decision algorithm for H.264/AVC high resolution real-time systems

Kuen-Cheng Chiang *, Ming Feng Wu, Jean Jyh-Jiun Shann

Department of Computer Science and Information Engineering, College of Electrical Engineering and Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC

ARTICLE INFO

Article history:
Received 1 August 2010
Accepted 27 October 2011
Available online 31 October 2011

Keywords:
Mode decision
Direction decision
Vector calculation
Intra prediction
H.264/AVC
Five-value MB sorting
Butterfly architecture
Histogram strength accumulation

ABSTRACT

In this paper, we propose an architecture for H.264/AVC fast intra-prediction-mode decision making in high resolution real-time applications. Intra-prediction-mode decision making requires many computations of H.264/AVC video coding, and also extra time for mode generation for intra prediction mode decisions. Hence, there exists a bottleneck in the execution of high resolution real-time applications. To improve the operation of intra prediction mode decision, we use an algorithm which, based on the edge information of an object, will reduce estimations of mode predictions by 66%; with negligible loss of video quality and a small increase in bit-rate of video stream. We propose a low cost architecture, with gate counts reduced by 50% compared with former design. The total gate count is 86,671 and the maximum operating frequency is 250 MHz using TSMC 0.18 μm cell-based technology. The experimental results show our design is a strong competitor with most modern high resolution, real-time video processing.

1. Introduction

High-resolution video is one of the most attractive and accessible visual mediums. In order to deliver more video content over limited transmission bandwidth, techniques of video encoding have been developed to dramatically reduce the storage size of the content. This is achieved by simplifying unnecessary information that is indistinguishable to the human eye. In 2001, a new standard of video encoding technology called H.264/AVC (Advanced Video Coding) was developed by the ITU-T VCEG (Video Coding Experts Group) and ISO/IEC 14496-10 AVC MPEG (Moving Picture Experts Group) [1]. MPEG-2, H.264/AVC can achieve a bit-rate reduction of about 39% to 64% over older video encoding standards such as MPEG-4, H.263 and also maintain about twice the video quality of the older standards. These improvements in coding performance are due to improvements in the prediction components, including inter-prediction and intra-prediction.

The high coding efficiency of H.264/AVC is achieved due to the rate distortion optimization (RDO) technique. This approach examines all mode combinations of inter-prediction and intra-prediction and is therefore computationally expensive. If we perform intra-prediction by a full search, i.e. completely executing all possible modes, the coding will suffer from wasted mode calculations. This motivates us to explore an efficient intra-prediction solution for H.264/AVC video coding.

Intra-prediction is a coding method for reducing spatial redundancy between frames; inter-frame and intra-frame computation is performed to determine the block type. Indeed, the intra-prediction algorithm for video coding has the same influence on encoded video quality and encoded bit-stream size as inter-prediction. And since fast intra-prediction is a new development in H.264/AVC video coding, new research is published every year. One such method is to use the edge information of an object [2–5]. Other methods include transformation of the domain [6–8]. Generally speaking, the latter is usually more complex than the former, and requires more hardware. Other techniques have also been developed [9–11].

In our fast intra-mode prediction method, we use the same algorithm as Li et al. [4] but utilize a different hardware design. We began by designing a different memory access scheme that generates virtual pixels cycle by cycle that can be pipelined easily without extra registers. We have proposed a modified architecture for gradient-vector calculations and direction detection. Both architectures control the detection of intension and direction for each virtual pixel. Finally, we developed a sorting architecture to process 4×4 blocks and macro blocks, with a design that can pick up the top three maximum values during each cycle. Our experimental results indicate that these direction decisions and vector calculation mechanisms allow the design of hardware with faster execution speed and at lower cost.

* Corresponding author.
E-mail address: kcc.chiang@msa.hinet.net (K.-C. Chiang).

1047-3203/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2011.10.007

2. Background and related works

H.264/AVC supports the encoding of high resolution video, but has bottlenecks affecting real-time applications since compression involves long calculation times. To achieve the goals of a real-time application, we focused our efforts on intra-prediction. High performance hardware that accelerates intra-prediction would be useful contribution towards real-time H.264/AVC systems.

2.1. H.264/AVC video coding

H.264/AVC is the newest international standard for video coding and emphasizes efficiency and reliability. Many new technologies are used in this standard, such as variable block size motion estimation (VBSME), multiple reference frames (MRF), 4×4 inter transform, etc. [12]. Using these new techniques for video compression, H.264/AVC achieves higher coding efficiency and provides higher quality video streaming compared to previous standards. To achieve such coding efficiency, H.264/AVC uses a complex mode-decision method based on rate-distortion optimization (RDO), which is computationally expensive due to the long calculation time required by the prediction process. Until now, time-consuming RDO calculations have been the bottleneck for high resolution real-time systems.

2.2. Introduction to fast intra mode prediction

Although inter-prediction and intra-prediction are key techniques for improving video compression in H.264/AVC [13], they introduce a performance bottleneck for real-time high resolution applications. Fig. 1 indicates that intra predictor generation and transformation, and mode decision, the two major computations account for 57% and 20% respectively of the total execution time in H.264/AVC encoding. Since both these steps are closely correlated with the intra-prediction process [14], we wish to reduce their computation time. Therefore, fast intra-prediction mode decision making technology has been developed to speed up the process of intra-prediction mode decisions.

H.264/AVC intra mode prediction is a spatial video compression technology for choosing the best mode with minimum difference between each prediction block. In this scheme, each block is constructed by the previously encoded and reconstructed block, and subtracted from the current block prior to encoding. For luma prediction, there are nine prediction modes for each 4×4 block and four prediction modes for a 16×16 block. For chroma prediction, only one block size of 8×8 is supported by four prediction modes. Figs. 2 and 3 show nine prediction modes for each 4×4 luma component and four prediction modes for a 16×16 component. The blocks in row I or column A of the grid are samples from the previously reconstructed block and the arrows indicate the direction of prediction for each block. The four prediction modes used for 8×8 chroma components are similar to 16×16 luma prediction modes, except that the labels of the modes are as follows: DC (mode 0), horizontal (mode 1), vertical (mode 2) and plane (mode 3). The prediction mode chosen by the encoder should have minimum

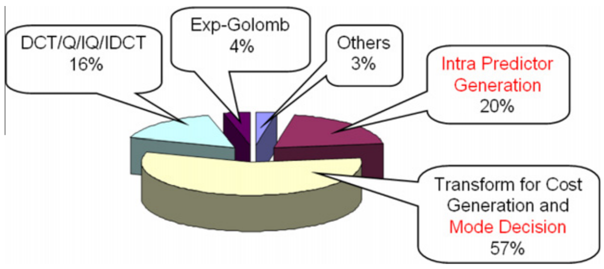

Fig. 1. Computation distribution for in H.264/AVC.

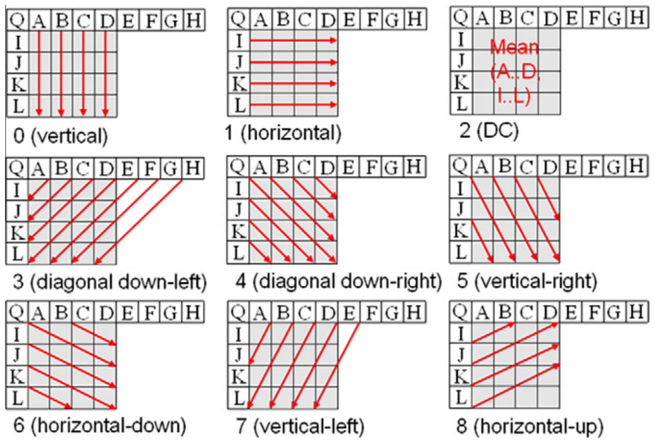

Fig. 2. Nine modes of 4×4 luma intra prediction.

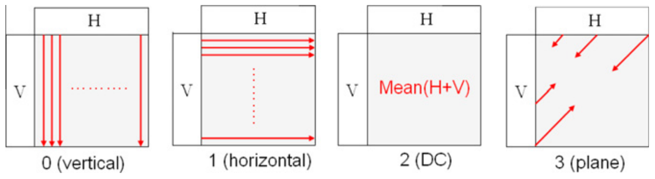

Fig. 3. Four modes of 16×16 luma intra prediction.

difference between a prediction block and the currently processed block.

The rate-distortion performance of video coding systems describes the tradeoff between the quality and the compressed bit rate. The execution flowchart is shown in Fig. 4. To identify the best prediction mode for video coding performance, three variables need to be taken into consideration: quality, compressed bit rate and computational cost. The function for the cost of the rate-distortion (RDcost) computation is given in Eq. (2.1) where S_k, I_k, Q, λ_{MODE} represent the macro block number, mode number, quantization value and Lagrangian parameter, respectively. D_{REC} is an estimate of the distortion and is measured by calculating the sum of square values between the constructed and the original macro block pixels. R_{REC} is the rate which the result of entropy coding. After estimating predictions for a macro block, the mode with minimum J is chosen as the suitable prediction.

$$J_{MODE}(S_k, I_k|Q, \lambda_{MODE}) = D_{REC}(S_k, I_k|Q) + \lambda_{MODE} R_{REC}(S_k, I_k|Q) \qquad (2.1)$$

Methods for fast intra-prediction mode decisions can further reduce the total time taken for difference calculations. These can be classified into three categories. The first classification uses characteristics extracted from the block to avoid unnecessary computation. The second classification divides the prediction modes into groups as per the corrections extracted from them. The third approach tries to maintain a parallel and pipelined intra-prediction operation. Pan et al. [2] proposes edge based fast intra-mode

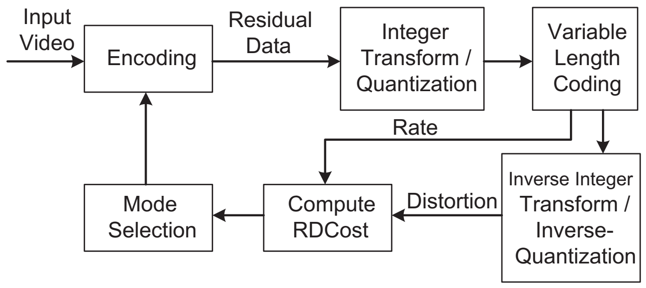

Fig. 4. The computation flow of RDcost.

prediction, which uses the information of an objective edge to find a suitable mode for a block. Unlike other methods, this technique only requires spatial analysis and can be easily implemented with fewer hardware resources. Experimental results also show that this method improves coding speed significantly with negligible loss in quality.

The edge based fast intra-mode prediction method is based on observation of locations. Pixels that are located along the direction of a local edge usually have similar values for both luma and chroma components. This method can be broken down into two steps. The first step involves generating values that can represent the features, amplitude and direction of a small part of a block. This information is stored or accumulated whereas the second step involves processing a block base.

2.3. Related works

Pan et al. [2] proposed a method to find the edge information in which the adjacency of the intra-block and the edge map of an image are produced by the Sobel edge operators. The edge map is the edge vector, including its edge direction and amplitude, for each pixel. The Sobel operator contains two convolution cores to calculate the difference in both vertical and horizontal directions. The Sobel operator is described in Eqs. (2.2) and (2.3) where $dx_{i,j}$ and $dy_{i,j}$ represent the difference in vertical and horizontal directions respectively, and $p_{i,j}$ is the pixel value in the i-th column and j-th row.

$$dx_{i,j} = p_{i-1,j+1} + 2 \times p_{i,j+1} + p_{i+1,j+1} - p_{i-1,j-1} - 2 \times p_{i,j-1} - p_{i+1,j-1} \tag{2.2}$$

$$dy_{i,j} = p_{i+1,j-1} + 2 \times p_{i+1,j} + p_{i+1,j+1} - p_{i-1,j-1} - 2 \times p_{i-1,j} - p_{i-1,j+1} \tag{2.3}$$

If an edge vector is defined as $\vec{D}_{i,j} = \{dx_{i,j}, dy_{i,j}\}$, then the amplitude and direction of the edge vector can be estimated by the following formulae.

$$Amp(\vec{D}_{i,j}) = |dx_{i,j}| + |dy_{i,j}| \tag{2.4}$$

$$Ang(\vec{D}_{i,j}) = \frac{180^\circ}{\pi} \times \arctan\left(\frac{dy_{i,j}}{dx_{i,j}}\right), \quad |Ang(\vec{D}_{i,j})| < 90^\circ \tag{2.5}$$

To decide on the proper modes for prediction, an edge direction histogram is created from the pixel map of the block by accumulating the amplitude of the edges with similar directions in the block. There are eight cells in a 4 × 4 block histogram and three cells in a macro block histogram, because the DC mode is always chosen in accordance with experiment. For 4 × 4 block intra-prediction mode decision, the cell with the maximum amplitude and its two adjacent cells are chosen with the DC mode to perform further RDO calculations. For the macro block intra predication mode decision, only two modes are required in the RDO calculation. Therefore, the total calculations of intra-prediction mode decision making can be reduced massively and the processing speed is also increased.

Wang et al. [3] try to simplify the decision of choosing a mode by unifying any image-block, including 4 × 4, 16 × 16 and 8 × 8 blocks, as a composition of four sub-blocks called Pseudo-blocks, as shown in Fig. 5. We can get a 2 × 2 pseudo-block with a0, a1, a2 and a3 pixel values, which are produced by averaging all pixels of each sub-block. Fig. 6 shows that the five types of directional filters operating with the pseudo-block are vertical edge, horizontal edge, the 45° edge, the 135° edge, and no-directional edge. Filter coefficients associated with each type are also shown. Each computational result represents the strength of the pseudo-block in that direction and is labeled as $S^v, S^h, S^{45^\circ}, S^{135^\circ}$ and S^{nd}. To find the strongest direction of the pseudo-block, the modes which are related

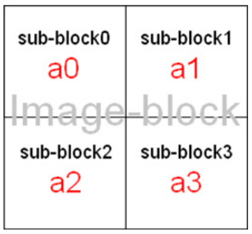

Fig. 5. Definition of a pseudo-block.

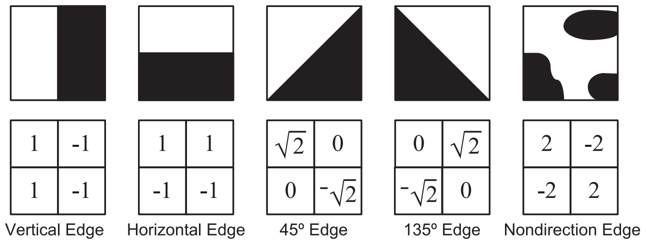

Fig. 6. Five types of directional filters and coefficients.

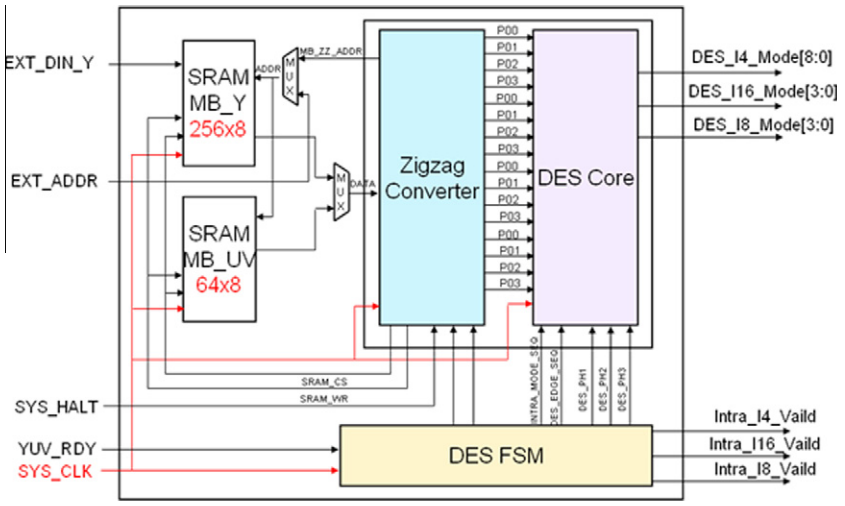

Fig. 7. Architecture of the DES fast intra prediction.

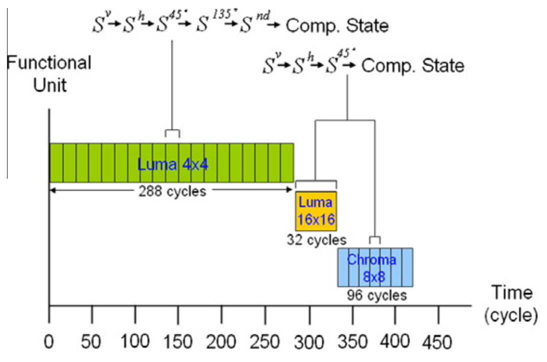

Fig. 8. Timing schedule of the DES engine.

with this direction will be analyzed further during the RDO computations. For example, if S^v is the maximum, mode 0, mode 2, mode 5 and mode 7 are chosen for RDO calculation of the 4 × 4 block.

$$Ep = \arg \max_{v,h,45^\circ,135^\circ,nd} \{S^v, S^h, S^{45^\circ}, S^{135^\circ}, S^{nd}\} \tag{2.6}$$

Figs. 7 and 8 depict the hardware architecture and timing schedule of this design. The zigzag converter is connected to memory to arrange the sequence of image data from raster scan order into zigzag scan order. The dominate edge strength (DES) [3] core is a popular design responsible for pseudo-block computation, edge filtering and DES extraction. The DES finite state machine (FSM) is in charge of generating a timing schedule. The time spent for one macro block, including both luma and chroma components, is 416 cycles. The DES core is in charge of finding the dominate edge according the five directional edges as shown in Fig. 6 and the predicted modes for 4 × 4, 8 × 8 and 16 × 16 blocks are generated based on the five strength values.

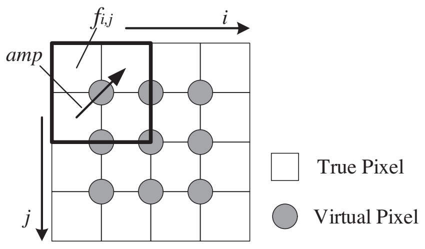

Fig. 9. A 2 × 2 filter for gradient vector calculation.

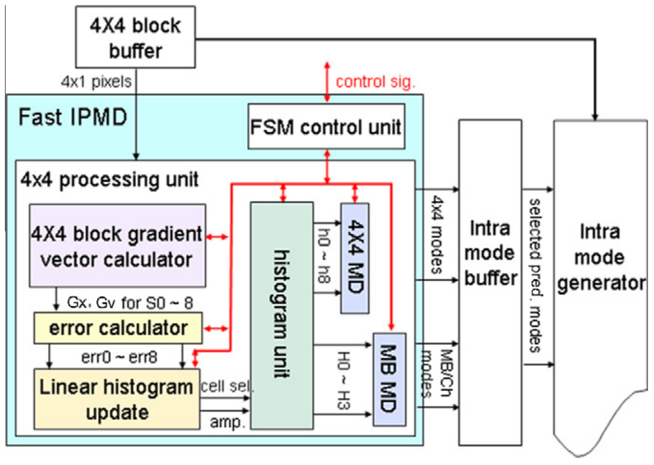

Fig. 10. Architecture of fast IPMD.

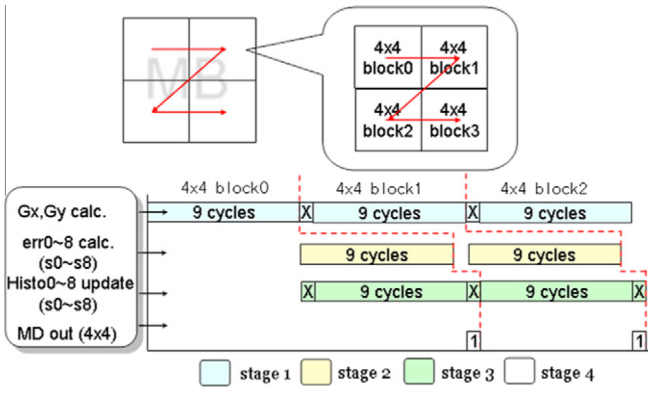

Fig. 11. Timing diagram of the fast IPMD.

The third method proposed by Li et al. [4] is a simple and easily implemented solution based on a filter of 2 × 2 block size to process the 4 × 4 blocks. As shown in Fig. 9, there are only four pixels included in the thick block to generate one virtual pixel. $\vec{G}_{i,j}$ is the gradient vector and it represents the direction of the virtual pixel. It contains the vertical component $Gx_{i,j}$ and the horizontal component $Gy_{i,j}$. Both components are described in Eqs. (2.7) and (2.8).

$$Gx_{i,j} = (f_{i,j+1} - f_{i+1,j+1}) + (f_{i,j} - f_{i+1,j}) \quad i,j = 0,1,2 \tag{2.7}$$
$$Gy_{i,j} = (f_{i+1,j} - f_{i+1,j+1}) + (f_{i,j} - f_{i,j+1}) \quad i,j = 0,1,2 \tag{2.8}$$

The amplitude of the virtual pixel $Amp(\vec{G}_{i,j})$ is the absolute sum of Gx and Gy as calculated in Eq. (2.9). Other important virtual pixel information, such as the direction, can be calculated by the modified algorithm listed in Eq. (2.10). The result err is the difference of $|Gx|$ and $|Gy|$ in different weights. The minimum err is the proper direction of this virtual pixel and should be kept in the edge direction histogram as discussed earlier in Pan's work. Finally, suitable modes for processing 4 × 4 blocks are chosen according to the results of RDO computation. It requires four modes for 4 × 4 blocks and three modes for 16 × 16 and 8 × 8 blocks.

Fig. 10 shows the hardware architecture of fast intra prediction mode decision (fast IPMD) proposed by Li et al. [4] and the execution timing diagram is shown in Fig. 11. The 4 × 4 block gradient vector calculator, error calculator and linear histogram update are responsible for computing the mode information of each virtual pixel. A histogram unit stores the information of each virtual pixel of 4 × 4 blocks to find the proper mode for each block. Because information is reused for 4 × 4 blocks, the total time spent on one macro block is only 170 cycles.

$$Amp(\vec{G}_{i,j}) = +|Gx_{i,j}| + |Gy_{i,j}| \tag{2.9}$$

$$\begin{aligned}
err_0 &= ||Gy_{i,j}| - 8|Gx_{i,j}|| \\
err_1 &= ||Gy_{i,j}| - 0.125|Gx_{i,j}|| \\
err_3 &= \begin{cases} ||Gy_{i,j}| - 1|Gx_{i,j}|| & (sign(Gy_{i,j}) = sign(Gx_{i,j})) \\ \infty & else \end{cases} \\
err_4 &= \begin{cases} ||Gy_{i,j}| - 1|Gx_{i,j}|| & (sign(Gy_{i,j}) \neq sign(Gx_{i,j})) \\ \infty & else \end{cases} \\
err_5 &= \begin{cases} ||Gy_{i,j}| - 2|Gx_{i,j}|| & (sign(Gy_{i,j}) \neq sign(Gx_{i,j})) \\ \infty & else \end{cases} \\
err_6 &= \begin{cases} ||Gy_{i,j}| - 0.5|Gx_{i,j}|| & (sign(Gy_{i,j}) \neq sign(Gx_{i,j})) \\ \infty & else \end{cases} \\
err_7 &= \begin{cases} ||Gy_{i,j}| - 2|Gx_{i,j}|| & (sign(Gy_{i,j}) = sign(Gx_{i,j})) \\ \infty & else \end{cases} \\
err_8 &= \begin{cases} ||Gy_{i,j}| - 0.5|Gx_{i,j}|| & (sign(Gy_{i,j}) = sign(Gx_{i,j})) \\ \infty & else \end{cases}
\end{aligned} \tag{2.10}$$

Table 1
Comparisons between related works.

	Peng et al. [2]	Wang et al. [3]	Li et al. [4]
Algorithm complexity	High	Middle	Low
Hardware requirements (gate count)	NA	10.3 k	15.8 k
Max. △Bitrate (%)	3.39	4.437	5.57
Max. △PSNR (db)	−0.177	−0.294	−0.28

Table 1 gives a short summary of three related works in algorithm complexity, hardware requirements, maximum △Bitrate and maximum △PSNR. Our goal is to propose a design that not only speeds up intra-prediction mode decision making in H.264/AVC, but is also less complex. Therefore, the design proposed by Peng et al. [2] cannot be utilized because of its high complexity, which conflicts with our design goal. In Wang's [3] research, a zigzag converter is used for the re-arrangement of data and it also reduces the complexity and hardware requirement of the DES core. Based on the requirements of real time intra-prediction, we choose the design proposed by Li et al. [4] for a reference design of ours because of the lowest algorithm complexity and negligible loss in bit rate and PSNR.

3. Design

The modified structure for edge based, fast intra-mode decision making is shown in Fig. 12. The architecture is composed of five components: the 4×4 block gradient vector calculator, direction decider, histogram unit, 4×4 block mode decider and micro block mode decider. The 4×4 block pixels are stored in the 4×4 block buffer and can be used by the intra mode generator. The intra mode buffer stores the candidate modes generated by the fast intra-prediction mode decisions. The finite state machine (FSM) control unit is responsible for maintaining the timing schedule of the mode decision.

In the 4×4 processing core, both the 4×4 block gradient vector calculator and direction detector are responsible for generating the virtual pixels and computing the strength and direction of each virtual pixel. The histogram unit is an accumulator, and it stores the strength according to the directional information. Finally, the 4×4 block mode decisions and the macro block mode decisions are suitably processed for 4×4 block and macro block, respectively.

3.1. Algorithm description

At the start of the execution flow, four pixels are given to a 2×2 filter, the thick block in Fig. 9, to calculate the value of each virtual pixel. There are a total of nine virtual pixels for each 4×4 block to

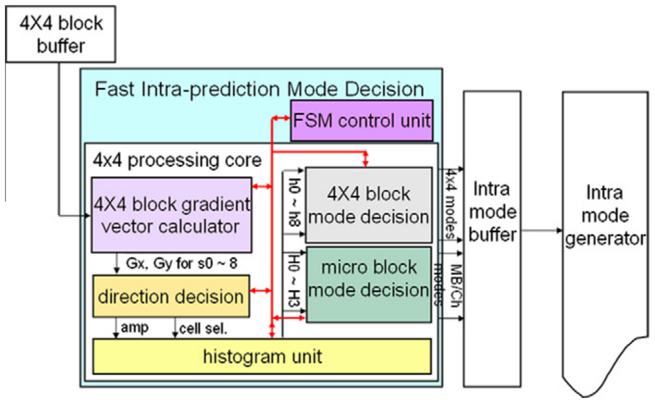

Fig. 12. Proposed architecture for fast intra mode decision.

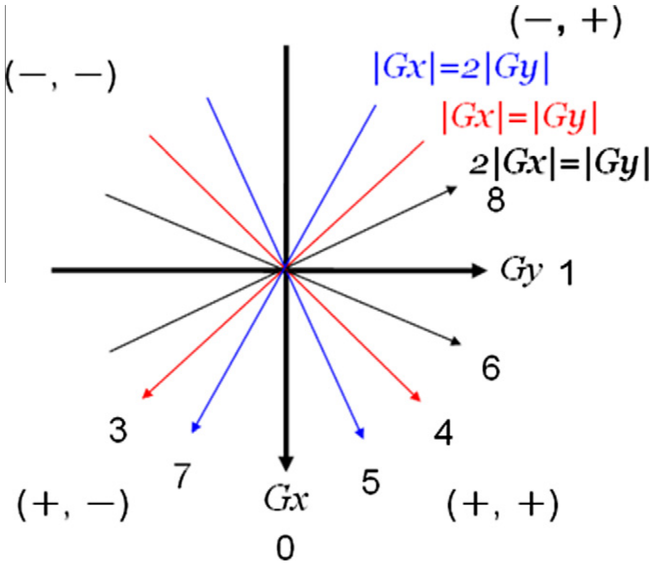

Fig. 13. The modified intra prediction mode direction.

decide the proper mode, and each virtual pixel includes the vertical value *Gx* and horizontal value *Gy* derived from Eqs. (2.7) and (2.8). The strength (amp) of the virtual pixel is the sum of the absolute value of *Gx* and *Gy*, as shown in Eq. (3.1). The direction is decided by the sign information of *Gx* and *Gy* by applying Eqs. (3.2), (3.3) and (3.4).

$$amp = +|Gx_{i,j}| + |Gy_{i,j}| \tag{3.1}$$

$$Gx = |Gy| \tag{3.2}$$

$$Gx = 2|Gy| \tag{3.3}$$

$$Gy = 2|Gx| \tag{3.4}$$

As shown in Fig. 13, the sign information of *Gx* and *Gy* divides the plane to four parts. If the sign of *Gx* is the same as *Gy*, the direction would cross the quadrant of first and third. Otherwise, it cross the quadrant of second and fourth; and further, Eqs. (3.2), (3.3) and (3.4) can judge the final answer for the current 4×4 block. There are only eight modes be taken into consideration because the DC mode is always chosen for the RDcost calculation and therefore we can neglect it.

In the final part of our design, the top three modes that are strongly correlated in their direction with mode 2 are chosen for the RDcost calculation for each 4×4 block. Obviously, two adjacent modes will always be decided for one virtual pixel of a 4×4 block in every cycle. To reduce size, we can't sort the directional values to find the proper modes until all virtual pixels are processed, because of the huge hardware requirements. Therefore, we do the sorting processing every cycle with only five data points, including two new generating mode values and three mode values produced in the former cycle. The suitable modes for each 4×4 block will be given in a period of nine cycles. For a macro block, we need 144 cycles to get the proper modes.

3.2. Execution scheduling

The timing schedule of our design is shown in Fig. 14; the scan order of the 4×4 block is still performed in a zigzag manner, as in the specification of H.264/AVC, and processed sequentially. The first stage is responsible for generating the virtual pixel and each virtual pixel is sent to next stage one by one as the cycle proceeds. In general, only nine cycles are needed to process one 4×4 block.

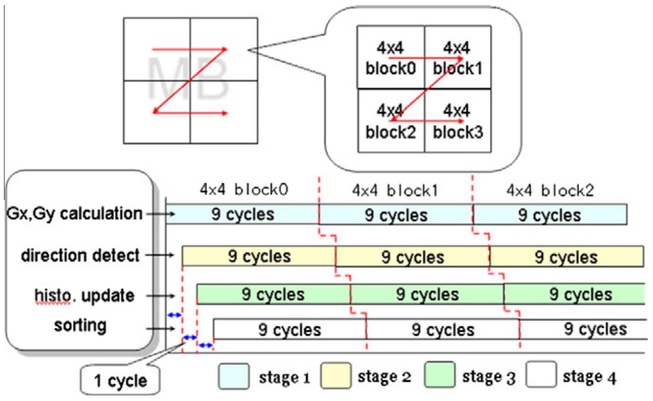

Fig. 14. Execution timing chart of the proposed architecture.

3.3. Memory access scheme

To achieve the pixel-by-pixel schedule, we should modify the memory access scheme in the beginning. Fig. 15 shows the access method used in our design. In the first three cycles, row 1 and row 2 are fetched and held; only the pixels belonged to column 1 and column 2 are used for generating the virtual pixels (i.e. pixels inside the black box in Fig. 15). In the next cycle, we move the black box one step to the right and choose pixels included in it (i.e. pixel 1, 2, 5 and 6) to generate the second virtual pixel. This procedure needs nine cycles to produce all virtual pixels for one 4×4 block.

3.4. 4×4 gradient vector calculator

Eqs. (3.5) and (3.6) shown below are modified forms of Eqs. (2.7) and (2.8). Both these formulae have the same value inside the parenthesis but have different signs in the first expression. As a result, only four computations are needed in order to get *Gx* and *Gy*. Our design of a gradient vector calculator is composed of four adders and separated into two levels to calculate the absolute value and sign information of *Gx* and *Gy*.

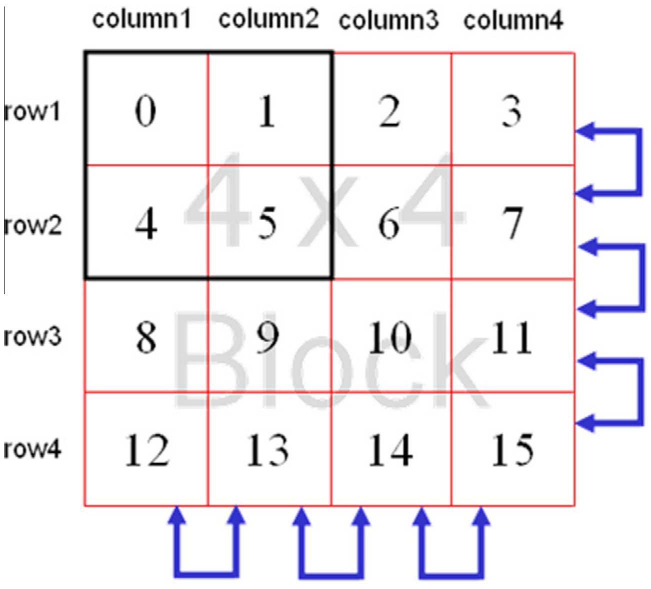

Fig. 15. Memory access scheme.

Table 2
Relation between sign bits, XNOR and direction of virtual pixel.

Gx_{sign}	Gy_{sign}	XNOR	Direction of virtual pixel
0	0	1	1, 3 quadrant
0	1	0	2, 4 quadrant
1	0	0	2, 4 quadrant
1	1	1	1, 3 quadrant

Table 3
Mode selection scheme.

Direction information bit	Selected mode (4×4)	Selected mode (MB)
000	0, 7	0, 3
001	7, 3	0, 3
010	3, 8	1, 3
011	8, 1	1, 3
111	6, 4	1, 3
101	4, 5	0, 3
101	4, 5	0, 3
110	5, 0	0, 3

$$Gx_{i,j} = -(f_{i+1,j} - f_{i,j+1}) + (f_{i+1,j+1} - f_{i,j}) \tag{3.5}$$
$$Gy_{i,j} = +(f_{i+1,j} - f_{i,j+1}) + (f_{i+1,j+1} - f_{i,j}) \tag{3.6}$$

3.5. Direction detector

According to the algorithm described in Section 3.1, we can implement the modified intra-prediction mode decision making model without needing any multiplication. The sign bit of *Gx* and *Gy* is used to decide which quadrants are crossed by the direction of a virtual pixel. The relationship between the signs of *Gx* and *Gy*(Gx_{sign} *and* Gy_{sign}) is shown in the logical output of XNOR, and the direction of the virtual pixels is shown in Table 2. The range information is composed of three bits and is used to select the proper modes for the current virtual pixel. Each data point picks up two adjacent modes. Table 3 lists all possible range information bits and their relative modes.

3.6. Histogram accumulator

The histogram unit is responsible for accumulations as shown in Fig. 16. The right part is used for 4×4 block processes while the left part is used for macro block processes. Registers r1 to r11 are the default mode used in H.264/AVC intra-prediction apart from the DC mode (mode 2 in 4×4 block and macro block). In the beginning of the 4×4 block process, based on range information given by the previous unit, the multiplexers pick up two registers from r1 to r8. The selected registers are adjacent modes (or directions) in Fig. 13. Then the amplitude (amp) of the current virtual pixel will be added to these two registers. As well as being used in the next stage, the outputs of the adders are sent back to update the old values of the selected registers, as described by Eq. (3.7), where *i* is the cycle number and *r* is the selected register. All the register values are cleared to zero in periods of nine cycles because there are nine virtual pixels in a 4×4 block. The macro block is processed in a similar way except that the values of registers are set to zero when all virtual pixels in a macro block are processed and the total number of cycles of the MB process is 144.

$$r_{i+1} = r_i + amp_i \quad (i = 1, 2, \ldots 8) \tag{3.7}$$

3.7. The sorting hardware for 4×4 block and macro block

Fig. 17 depicts the structure of the 4×4 block mode decider, which is divided into three parts: the zero setting comparator,

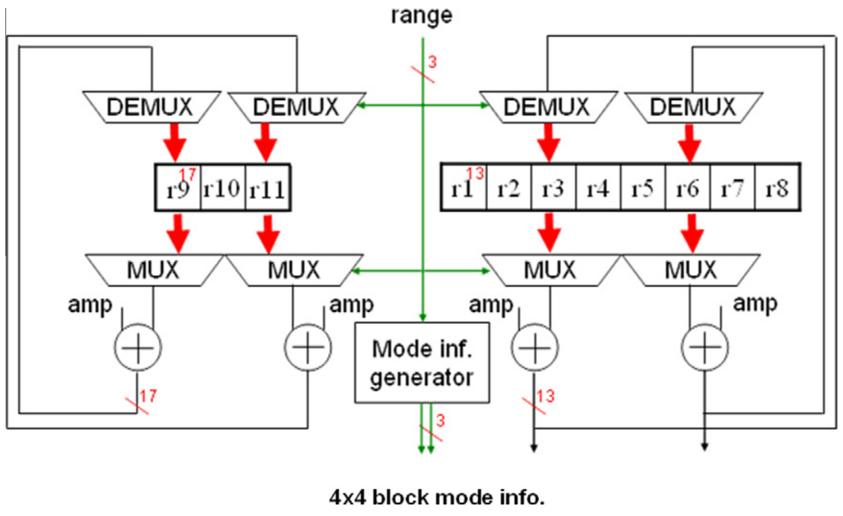

Fig. 16. Architecture of the proposed histogram unit.

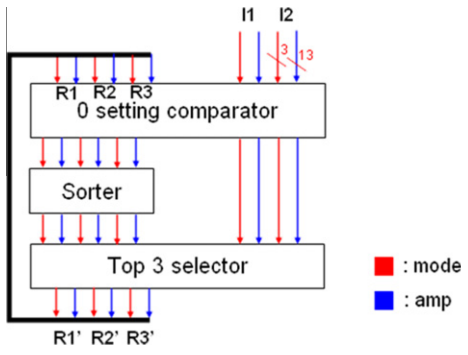

Fig. 17. Structure of the 4 × 4 block mode decision.

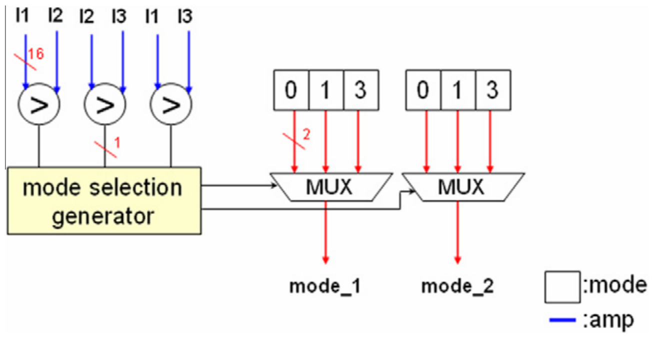

Fig. 18. Structure of the MB mode decision.

sorter, and top-three selector. R1, R2 and R3 are registers of the zero setting comparator; I1 and I2 represent input mode information which comes from the histogram unit. Each register or input consists of two values: mode and amplitude. In the first stage, the modes of inputs are checked to confirm whether the input modes are equal to the modes of the registers or not. If the mode of a register is the same as one of the inputs, the amplitude of the register is set to zero because the mode value of the input is already accumulated into the histogram unit. Stage two and stage three both arrange the information in decreasing order according the amplitudes. The top three R1′, R2′ and R3′ are stored back to replace R1, R2 and R3 respectively for the next iteration. As described in Section 2 and Fig. 9, there are nine virtual pixels generated by a 2 × 2 filter for gradient vector calculation for a 4 × 4 block. Finally, after the completion of nine iterations,

Table 4
Selecting algorithm of MB mode decision.

Comparator 1	Comparator 2	Comparator 3	Output	
			O1	O2
1	1	1	Mode_0	Mode_1
1	0	1	Mode_0	Mode_3
0	1	1	Mode_1	Mode_0
0	1	0	Mode_1	Mode_3
1	0	0	Mode_3	Mode_0
0	0	0	Mode_3	Mode_1

the top three amplitudes and the associated modes represent the result of 4 × 4 mode decision.

The architecture of the macro block mode decision is shown in Fig. 18. For a macro block, there are 16 4 × 4 blocks, and each of them takes 9 cycles to be processed. Hence, the macro block mode decision module takes 144 cycles to process the decision. The selecting algorithm of the macro block mode decision is shown in Table 4. After the processes of 4 × 4 block mode decision and MB mode decision are finished, the selected modes of them are stored into intra mode buffer. The intra mode generator fetches the resulting modes from intra mode buffer and generates selected prediction modes to intra frame coder for coding the video frames.

4. Experimental results

The proposed fast mode decision algorithm was implemented on JM9.3 [15] provided by JVT. In the experiments conducted, RD optimization is enabled and intra period is set to one. The test sequences used included Bus, Coastguard, Container, Football, Husky Mobile, Panzoom, Paris, Silent, Stockholm, Table, Tempete in the format of CIF 4:2:0 and City, Crew, Harbour, Kinghtshields, Parkrun in 1280 × 720 4:2:0 progressive format. The length of each test sequence was 200 frames and all the frames were encoded using I-frame coding. In order to implement the proposed design in hardware, the circuit was written in Verilog1995 register transfer level language and the logic synthesis was performed using DesignCompiler™ produced by Synopsys™ Inc. The design was implemented using the TSCM 0.18 μm technology cell library in order to evaluate gate counts, power dissipation and the maximum operating frequency of the proposed design. In order to verify the correctness of the design, a circuit simulation tool, ModelSim™ SE 6.1 by Mentor graphic corporation was used to evaluate the behavior of the circuit.

4.1. Comparison with related works

We proposed an improved hardware circuit with smoothly data feeding scheme to reduce hardware requirements in Intra-prediction mode decision. Compared with Li's [4] method, we always made the same results of mode decisions and thereby the performance of bit rate and PSNR can be maintained as well as Li's [4]. The results listed in Table 5 are the bit rate and PSNR deviations compared with our design and the H.264 reference software JM 9.3 [15]. The total number of RDO computations is reduced by about 66%, from 592 to 198, with a small bit rate overhead and an almost negligible PSNR loss is observed in our design. From the results, the maximum bit rate overhead is 5.57% and the maximum PSNR drop is 0.28db for the sequence 'football'. However, we can dramatically reduce the number of computations required and the hardware requirements which are both very important features for real-time high resolution coding applications.

There are two important modifications in our design which contribute to lowering hardware costs and increasing processing efficiency. The details of original design and the modifications are discussed below. The original architecture of the gradient vector calculator is shown in Fig. 19 and its access scheme is illustrated in Fig. 15. It needs nine cycles to calculate all the gradient vectors of a 4 × 4 block. In each cycle, four pixels are loaded from the 4 × 4 block buffer with three subtractions and are processed simultaneously. In the first four cycles, pixels are fetched row by row, and the *Gx* components are calculated. In the next four cycles, the other components, *the Gy's*, are processed. We realize that we require a register array to store the partial results.

In our design, a concept similar to that described in section two of Wang et al. [3] is used i.e. arranging the input data efficiently for processing. In each cycle, one virtual pixel (*Gx* and *Gy*) is produced; therefore the register array can be removed. The original design uses the algorithm listed in Eq. (2.10). The implementation of this algorithm requires at least eight adders processing simultaneously and extra hardware to deal with the sign bits. The generated values (i.e. err0, err1, err3… and err8) are processed by the histogram cell update control, which is shown in Fig. 20. Since the mode with the second-smallest error must be adjacent to the minimum mode, the minimums are first calculated between err0, err1, err3 and err4, relating to mode 0, mode1, mode3 and mode4. The results are used to find the second-minimum error.

The architecture we propose uses the sign bits to determine directions, and so the components of *Gx* and *Gy* (which have the same data size as err) are processed directly. Therefore, we just need three comparators to perform this procedure. The adders and the histogram cell update control can be replaced. Table 6 shows a comparison of the hardware cost of our design and Li's design in for main module of the 4 × 4 block processing core. Clearly, the hardware cost is very low in our proposed design.

Another component is modified in our design for the mode decision part. Let us take the 4 × 4 block mode decision as an example. In the 4 × 4 block processing core, the histogram unit is in charge of holding the amplitudes of each mode. If we perform the procedure of mode decision making after all amplitudes of the virtual pixels have been processed, we need to re-arrange an eight-cell row in decreasing order with 28 comparisons, which is too expensive to implement. In our proposal, the procedure of mode decision is executed when a virtual pixel is produced. There are only five components which need to be re-arranged and therefore the comparisons can be reduced to 10. However we designed this module in three stages; the information generated in the first stage is used in the second stage i.e. the sorter. Hence the number of comparison operations can be decreased. Finally, the hardware used for the 4 × 4 block mode decision making in our proposal is six 3-bit equality comparators and seven 13-bit comparators, instead of ten 16-bit comparators.

Table 7 shows a comparison between our proposal and related previous work. All three designs are implemented in 0.18 μm

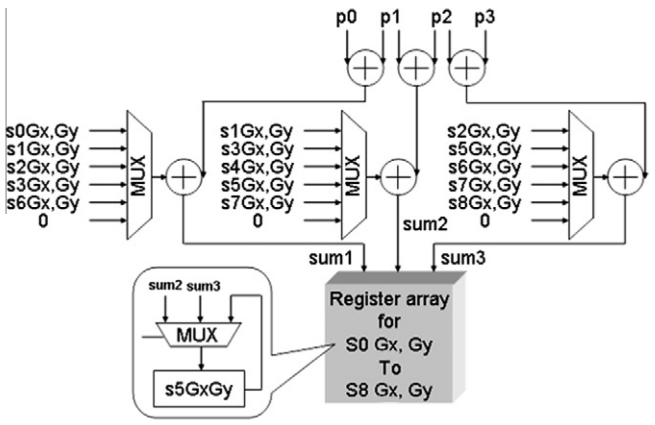

Fig. 19. Architecture of the gradient vector calculator.

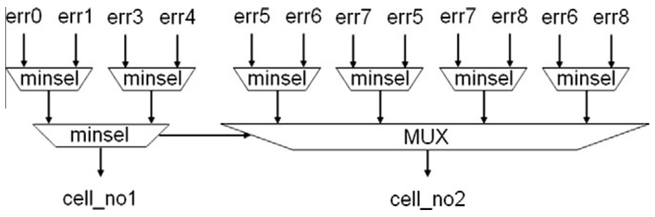

Fig. 20. Histogram cell update control.

Table 6
Comparison of hardware requirements in the main modules.

	Li's [4]		Proposed design	
	Vector calculation	Error calculation	Vector calculation	Direction decision
#Adder	6	8	4	1
#Comparator	0	7	0	3
#Register	18	0	0	0

Table 5
Coding performance on various video sequences.

Sequences	△Bitrate (%)	△PSNR (db)
Bus[a]	1.80	−0.08
Coastguard[a]	1.57	−0.06
Container[a]	0.57	−0.02
Football[a]	5.57	−0.28
Husky[a]	1.00	−0.08
Mobile[a]	2.29	−0.11
Panzoom[a]	1.97	−0.06
Paris[a]	2.52	−0.11
Silent[a]	2.86	−0.10
Stockholm[a]	2.21	−0.09
Table[a]	2.28	−0.09
Tempete[a]	2.29	−0.10
City[b]	0.54	−0.01
Crew[b]	2.23	−0.05
Harbour[b]	1.98	−0.07
Knightshields[b]	1.37	−0.03
Parkrun[b]	0.97	−0.05

[a] Means CIF video.
[b] Means 1280 × 720p video.

Table 7
Comparison of the implementation results.

	Wang's	Li's	Proposed #1	Proposed #2
Technology	UMC 0.18 μm	TSCM 0.18 μm	TSCM 0.18 μm	TSCM 0.18 μm
Cycle time	15 ns	5 ns	4 ns	20 ns
Cycle counts	416	210	183	183
Time/MB	6240 ns	1050 ns	732 ns	3660 ns
Gate counts	10.3 k	15.8 k	8.67 k	6.32 k
Power/MB	21,952 mW	NA	2436 mW	1189 mW

Table 8
Hardware resources of each module (250 MHz).

Module (250 MHz)	Gate counts	Power dissipation
4 × 4 block gradient vector calculation	21.1%	31.6%
Direction detection	19.5%	19.8%
Histogram strength accumulation	7.5%	4%
FSM control unit	3.9%	4%
4 × 4 block mode decision	28.3%	23.7%
Macro block mode decision	19.7%	16.9%

Table 9
Hardware resources of each module (50 MHz).

Module (50 MHz)	Gate counts	Power dissipation
4 × 4 block gradient vector calculation	11.3%	19%
Direction detection	21.7%	25%
Histogram strength accumulation	9.4%	6%
FSM control unit	4.9%	6%
4 × 4 block mode decision	29.9%	23%
Macro block mode decision	22.8%	21%

technology and our design has the lowest hardware costs (45.2% less than Li's [4] and 15.8% less than Wang's at maximum operating frequency), highest operating frequency and shortest processing time for one macro block. The power dissipation of our design is also lower than Wang's. These advantages make our proposal more desirable for H.264/AVC real-time systems as resolution increases. The hardware cost percentages of each module of our proposal are listed in Tables 8 and 9.

5. Conclusion

In this paper, low cost and high performance architecture for fast intra-mode prediction is explored and implemented. The improvements of the proposed design come from three components: the 4 × 4 block gradient vector calculator, direction decision making, and 4 × 4 block mode decision making. For block gradient vector calculations, butterfly architectures for the vector calculator and a block type memory access scheme are proposed. Both architectures generate the values of *Gx* and *Gy* of each visual pixel, cycle by cycle, and therefore we do not require space for saving incomplete values. Using this scheme, we reduce not only the hardware requirements but also the blocking time of the first 4 × 4 block. For direction decision making, we observe that the sign bit contains the direction information. By using the sign bit and the values of *Gx* and *Gy* immediately, we need only one processing cycle to find the proper direction of each 4 × 4 block. Only two processing stages of the 4 × 4 processing core are then required, which is one stage less than Li's design [4] and with less hardware complexity.

Finally, in mode decision making of 4 × 4 blocks, to replace the eight-value-sorting method, we design a five-value-sorting circuit with extra hardware for mode checking. The hardware picks up the top three values for every cycle and the exact result of each 4 × 4 block is found at the interval of nine cycles. This means that we do not require more time for processing between blocks and the corresponding architecture is not only smaller, but also faster than Li's.

As we see in the results, the architecture we designed for fast intra-prediction mode decision making has essentially the same video quality as Li's but implemented with almost half the hardware. Moreover, the maximum operating frequency and the processing cycle for one macro block of our hardware are also higher and shorter than Li's, respectively, and these advantages make our proposed design more suitable for high-resolution, real-time applications.

References

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC).
[2] Feng Pan, Xiao Lin, Susanto Rahardja, Keng Pang Lim, Z.G. Li, Dajun Wu, Si Wu, Fast mode decision algorithm for intraprediction in H.264/AVC video coding, in: Proceedings IEEE Transactions on circuits and system for video technology, (15)7, 2005.
[3] Jia-Ching Wang, Jhing-Fa Wang, Jar-Ferr Yang, Jang-Ting Chen, A fast mode decision algorithm and Its VLSI design for H.264/AVC intra-prediction, in: Proceedings IEEE Transactions on circuits and system for video technology 17(10), 2007.
[4] Shen Li, Xianghui Wei, Takeshi Ikenaga, Satoshi Goto, A VLSI architecture design of an edge based fast intra prediction mode decision algorithm for H.264/AVC, in: Great Lakes Symposium on VLSI and proceedings of the 17th great lakes symposium on Great lakes symposium on VLSI Stresa-Lago Maggiore, Italy SESSION: Architecture and memory Pages: 20–24 Year of Publication: 2007 ISBN:978-1-59593-605-9.
[5] Shen Li, Lingfeng Li, Takeshi Ikenaga, Shunichi Ishiwata, Masataka,Matsui Satoshi Goto, Content-based complexity reduction methods for MPEG-2 to H.264 transcoding, in: Proceedings IEICE Transactions on Information Systems (Inst. Electron Inf. Commun. Eng.), E90-D(1) 2007, pp. 90–98, ISSN:0916-8532.
[6] Kim Changsung et al., Fast H.264 intra prediction mode selection using joint spatial and transform domain features, in: Proceedings Journal of Visual Communication and Image Representation, 17(2) 2005, pp. 291–310.
[7] T. Tsukuba, I. Nagayoshi, T. Hanamura, H. Tominaga, H.264 fast intra prediction mode decision based on frequency characteristic, in: Proceedings European Signal Processing Conference (EUSIPCO), Antalya, Turkey.
[8] C. Kim, H.-H. Shih, C.-C.J. Kuo, Feature-based intra-prediction mode decision for H.264, in: Proceedings Image Processing, ICIP '04. 2004 International Conference Publication Date: 24-27 October 2004, vol. 2, pp. 769–772 ISSN: 1522-4880, ISBN: 0-7803-8554-3, INSPEC Accession Number: 8435983.
[9] Genhua Jin, Hyuk-Jae Lee, A parallel and pipelined execution of H.264/AVC intra prediction, computer and information technology, CIT apos; 06, in: The Sixth IEEE International Conference on September 2006, pp. 246–246.
[10] Liu Qiong, Hu Rui-min, Zhu LI, Zhang Xin-chen, Han Zhen, Improved fast intra prediction algorithm of H.264/AVC, in: Proceedings Journal of Zhejiang University – Science A, vol. 7, Supplement 1/2006.1, pp. 101–105, ISSN:1862-1775.
[11] Jun Sung Park, and Hyo Jung Song, Selective intra prediction mode decision for H.264/AVC encoders, in: Proceedings World Academy of Science, Engineering and Technology, vol. 13, May 2006, ISSN 1307-6884.
[12] H.264 and MPEG-4 VIDEO COMPRESSION Iain E. G. Richardson.
[13] Tung-Chien Chen, Yu-Wen Huang, Liang-Gee Chen, Analysis and design of macroblock pipelining for H.264/AVC VLSI architecture, in: Proceedings of the 2004 International Symposium on Circuits and Systems ISCAS apos II; 04, vol. 2, 23–26 May 2004 pp. 273–6.
[14] Y.W. Huang, B.Y. Hsieh, T.C. Chen, L.G. Chen, Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder, in: Proceedings IEEE Transactions on Circuits and System for Video Technology, 15(3) 2005, pp. 378–401.
[15] Joint Video Team (JVT) of ISO/IECMPEG and ITU-T VCEG, Reference Software to Comitee Draft. JVT-F 100 JM9.3.