北京大学学报(自然科学版), 第 50 卷, 第 4 期, 2014 年 7 月
Acta Scientiarum Naturalium Universitatis Pekinensis, Vol. 50, No. 4 (July 2014)
doi: 10.13209/j.0479-8023.2014.133

Low-Cost VLSI Implementation of Motion Estimation for H.264/AVC Encoders

WANG Teng[1], WANG Xin'an[1,†], XIE Zheng[1], HU Ziyi[2]

1. Key Lab of Integrated Micro-Systems, Shenzhen Graduate School of Peking University, Shenzhen 518055;
2. Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029;
† Corresponding author, E-mail: anxinwang@pku.edu.cn

Abstract A pipelined architecture with novel memory structure is proposed with several modifications of the ME algorithm. Fast motion estimation with low hardware cost and less memory access is achieved by proper search strategy, efficient rate distortion optimization (RDO) cost calculation and interpolation components, innovative memory structure and optimized dataflow scheduling. The proposed design is synthesized by SMIC 130 nm CMOS technology process with a clock frequency of 167 MHz and consumes 181.7 K logic gates and 13.8 KB memory, which shows great hardware efficiency compared with other designs. The proposed design was finally integrated within an H.264/AVC encoder for FPGA prototyping and VLSI implementation. The core area of the overall chip is 1.74 mm×1.74 mm with SMIC 65 nm CMOS technology which can support real-time HD(1080P@60fps) encoding with a clock frequency of 350 MHz.

Key words H.264/AVC; motion estimation; pipeline architecture; real-time HD encoding; VLSI implementation

H.264/AVC 编码器中运动估计的低代价 VLSI 实现

王腾[1] 王新安[1,†] 谢峥[1] 胡子一[2]

1. 北京大学深圳研究生院集成微系统科学工程与应用重点实验室, 深圳 518055;
2. 中国科学院微电子研究所, 北京 100029; † 通信作者, E-mail: anxinwang@pku.edu.cn

摘要 通过对运动估计算法进行优化, 提出一种应用新型存储结构的流水线实现结构。通过采用合适的搜索策略、高效的率失真优化代价计算和插值部件、创新的存储结构及优化的数据流调度, 实现具有低硬件代价和存储访问的快速运动估计。该设计在 SMIC 130 nm 工艺下综合, 时钟频率可达到 167 MHz, 消耗 181.7 K 逻辑门和 13.8 KB 存储, 相比同类设计具有更高的硬件效率。该设计集成在一个 H.264/AVC 编码器中进行 FPGA 原型验证和 VLSI 实现。SMIC 65 nm 工艺下, 整个芯片面积为 1.74 mm×1.74 mm, 工作频率为 350 MHz, 可以支持实时高清(1080P@60fps)编码。

关键词 H.264/AVC; 运动估计; 流水线结构; 实时高清编码; VLSI
中图分类号 TN47

While the resolution of videos becomes higher and higher, the bandwidth of networks in which video data is transmitted is still limited. Hard-wired video encoder with advanced compression methods is in urgent need. As an efficient video coding technique, H.264, announced by the Joint Video Team (JTV) in 2003, can reduce the bit-rate by up to 50% compared with MPEG-4 advanced simple profile[1], and achieve a compression ratio of more than 30:1, which significantly reduces the bandwidth requirement for real-time video transmission. In H.264, inter prediction (motion estimation and motion compen-

收稿日期: 2013–10–07; 修回日期: 2013–12–16; 网络出版日期: 2014–07–13

sation), intra prediction, integer DCT and Quantization, and entropy coding are adopted to eliminate temporal redundancy, spatial redundancy, visual redundancy and statistical redundancy, respectively. In our experiments, using an encoder implemented with inter prediction, the bit-rate can be reduced by 55%, compared with the encoder without it. When the video segment has a smooth and slow movement, this number can easily reach to 75%, which implies the vital role of inter prediction in H.264.

However, due to its complicated algorithm and high demand for data bandwidth, hard-wired inter prediction is not easy to implement. According to the research of Ostermann et al[2], the complexity of H.264 encoder is 10 times higher than MPEG-4 visual simple profile codec, of which inter prediction holds more than 50%[3], indicating the great cost of inter prediction operation. To avoid this exhausting computation, in certain researches of H.264 encoders, inter prediction is omitted and only intra prediction is conducted[4–6], while other researches on motion estimation mainly focus on algorithms that reduce the computation complexity and architectures that exploit the most parallelism[1,7–9].

In order to satisfy the demands of real-time video compression, after choosing a proper search strategy, designing and allocating computation components, and exploiting execution parallelism, an optimized pipelined architecture is proposed in this paper, which can achieve real-time motion estimation with HDTV specification.

1 Principle and Analysis of Motion Estimation

The inter prediction is highlighted in H.264 algorithm flow diagram, as shown in Fig. 1, which contains motion estimation and motion compensation (ME & MC). In this process, ME concentrates on searching for a block in the reference frames which matches the current encoding block best, and obtains a motion vector (MV) that points from the reference block to the current block, and then MC subtracts those two blocks to get the residual data. The operation of MC is so straightforward that it only consumes limited clock periods. In contrast, more than 52% power of H.264 encoder is consumed by ME[7], owing to its algorithmic complexity and heavy calculation burden. This comparison shows that ME is the key factor to determine the performance of inter prediction. In fact, inter prediction is often referred to as motion estimation, with MC incidentally included.

Motion estimation includes two operations, integer motion estimation and fraction motion estimation (IME & FME). IME is used to search for the best MV at actual pixels in reference frames, while FME aims at finding a better match at fraction pixels near the best integer MV. In the specification of H.264 standard, sub-sampling precision for luminance is 1/4-pixel. Fig. 2 shows the searching approach. The first step is looking for a best integer match based on the search strategy, after that, half-pixel positions are interpolated and evaluated, and eventually, a best quarter-pixel match is found, which concludes ME procedure.

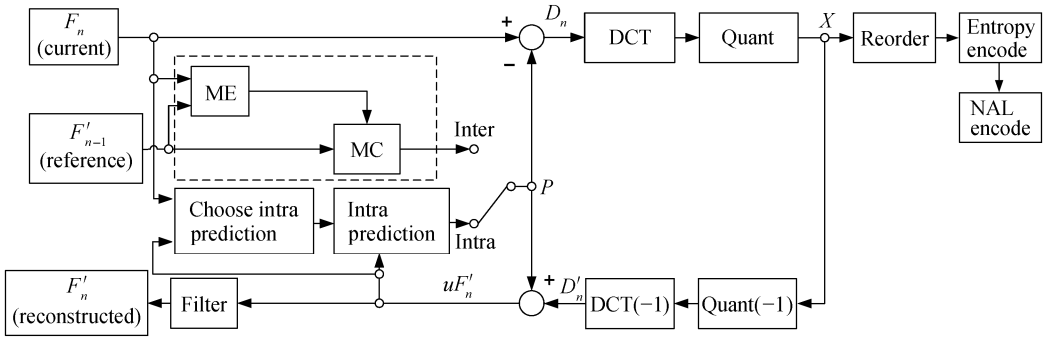


Fig. 1　Algorithm flow of H.264 encoder[2]

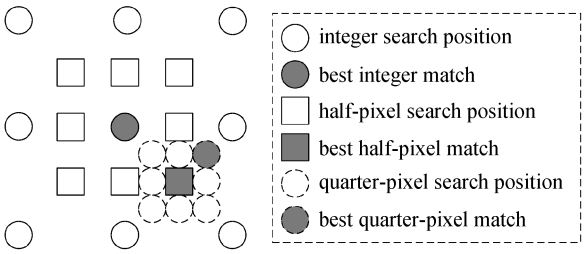


Fig. 2 Integer, half-pixel and quarter-pixel motion stimation[10]

1.1 Integer motion estimation

With the increasing of video resolution, the cost will be intolerable to search the best MV in the whole reference frames. Also, considering that the motion range between continuous frames is finite, it is unnecessary to search in the whole reference frames. Therefore, a search window with restricted size is often used to restrict the search range in practical applications and the common size of a search window is from 32×32 to 256×256 according to different search algorithms. Full Search (FS) can provide a higher compression ratio, because it will evaluate all possible pixels to find the best MV. However, it still suffers heavy burden of computation due to the massive computation, and can hardly achieve real-time encoding.

To eliminate this defect of FS, plenty of fast search algorithms have been presented, such as Three-Step Search (TSS)[11], Diamond Search (DS)[12], Hexagon Search (HS)[13], etc. Although those algorithms cannot always get the best MV, their search speed is much faster than FS, which makes them realizable. Recently, some more advanced algorithms, such as Predict Hexagon Base Search (PHS)[8] and Edge Detection technique[9], help to reduce the search time further. Nevertheless, these algorithms also come along with complicated control and irregular data access, which hold them back from hardware implementation.

Considering the real-time coding requirement and the complexity of implementation, in this paper, DS algorithm within a search window of 80×80 is chosen for motion estimation. As shown in Fig. 3, the searching approach is performed as follows.

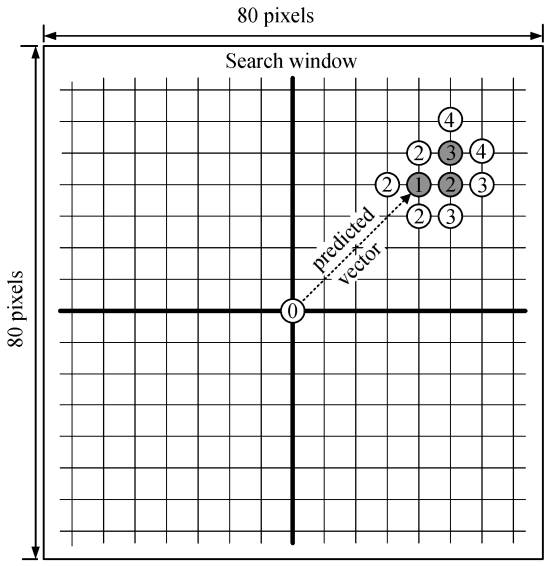


Fig. 3 Diamond search algorithm[10]

1) Locate the initial point based on predicted vector.

2) Search the surrounding points in a diamond shape, which is centered by the current point.

3) Repeat Step 2, until the SAD (sum of absolute differences) of the center point is the lowest one, or the searching approach reaches the boundary of the search window, or the search times reaches the limitation.

1.2 Fraction motion estimation

Since the coding efficiency of IME is not enough for the limited bandwidth of networks, sub-sampling and FME are introduced to improve the compression ratio. The encoding efficiency of 1/4-pixel ME is significantly improved on that of 1/2-pixel ME, while 1/8-pixel ME shows much less improvement[10]. Additionally, the formula of 1/8-pixel interpolation is more complicated, which makes 1/4-pixel ME a better choice for encoding.

1) Half-pixel Interpolation: As illustrated in Fig. 4(a), half-pixel samples are interpolated using a six tap FIR with the weights of (1/32, –5/32, 5/8, –5/32, 1/32). Samples with half-integer x-coordinate are generated by six horizontal pixels, while samples with half-integer y-coordinate are generated by vertical pixels. Once those are done, the remaining samples are calculated by interpolating between six vertical

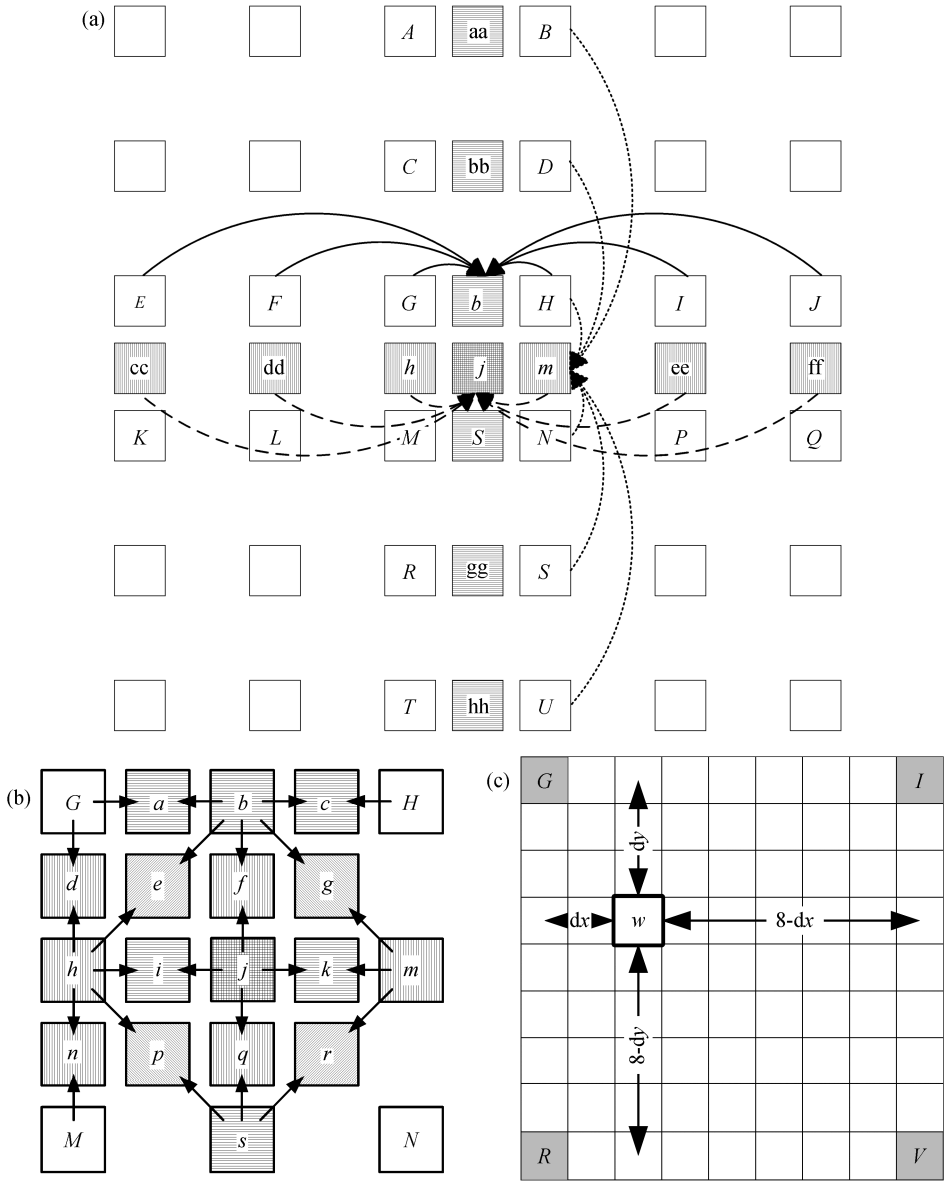


Fig. 4　Half-pixel interpolation (a), quarter-pixel interpolation (b) and 1/8 interpolation for chroma (c)

half-pixel samples from the first set of operations, or between horizontal half-pixel samples, which will get the same result. The formula of half-pixel interpolation is illustrated in Eq. (1).

$$b = \text{round}((E - 5F + 20G + 20H - 5I + J))/32. \quad (1)$$

The total number of half-pixel samples to be interpolated is $22 \times 17 + 17 \times 18 + 17 \times 17 = 969$ and SADs of 8 predicted macro-blocks needs to be calculated to obtain the best predicted macro-blocks with half-pixel sampling.

2) Quarter-pixel Interpolation: Once all half pixels are available and the best half-pixel position is determined, quarter-pixel samples are produced using linear interpolation between two adjacent integer or half-integer pixels, as shown in Fig. 4(b). Sample *a*, *c*, *l*, *k* are generated horizontally, sample *d*, *n* vertically, and sample *e*, *g*, *p*, *r* are generated diagonally. The formula is presented in Eq. (2).

$$a = \text{round}((G + b)/2). \quad (2)$$

There are totally 2048 quarter-pixel samples to be interpolated and also SADs of 8 predicted macro-blocks to be calculated. Considering the procedure of motion compensation, another 256 quarter-pixel

samples are interpolated, which build up the final predicted macro-block.

3) Chroma Interpolation: Since the resolution of the chroma samples is half of that of the luma samples, the motion vector for chroma, which is the same with that of the luma, is of one-eighth accuracy. The inter-prediction of the chrominance in H.264 baseline profile is realized by bilinear interpolation of four full-sample chroma pixels, as shown in Fig. 4(c). The formula is as follows:

$$w(\mathrm{d}x,\mathrm{d}y)=\mathrm{round}\{[(8-\mathrm{d}x)(8-\mathrm{d}y)G+\mathrm{d}x(8-\mathrm{d}y)I+(8-\mathrm{d}x)\mathrm{d}yR+\mathrm{d}x\mathrm{d}yV]/64\,. \quad (3)$$

As there are two color components cb and cr, the total number of chroma sub-samples to be interpolated is 128.

2 Proposed Architecture

The overall block diagram of the proposed architecture is demonstrated in Fig. 5. Considering that the operations of IME and FME are both exhausting, two-stage pipelining architecture is adopted to exploit operation parallelism and reduce hardware cost.

The first stage of pipeline is IME, in which CUR_L1 is the memory to store current block, and

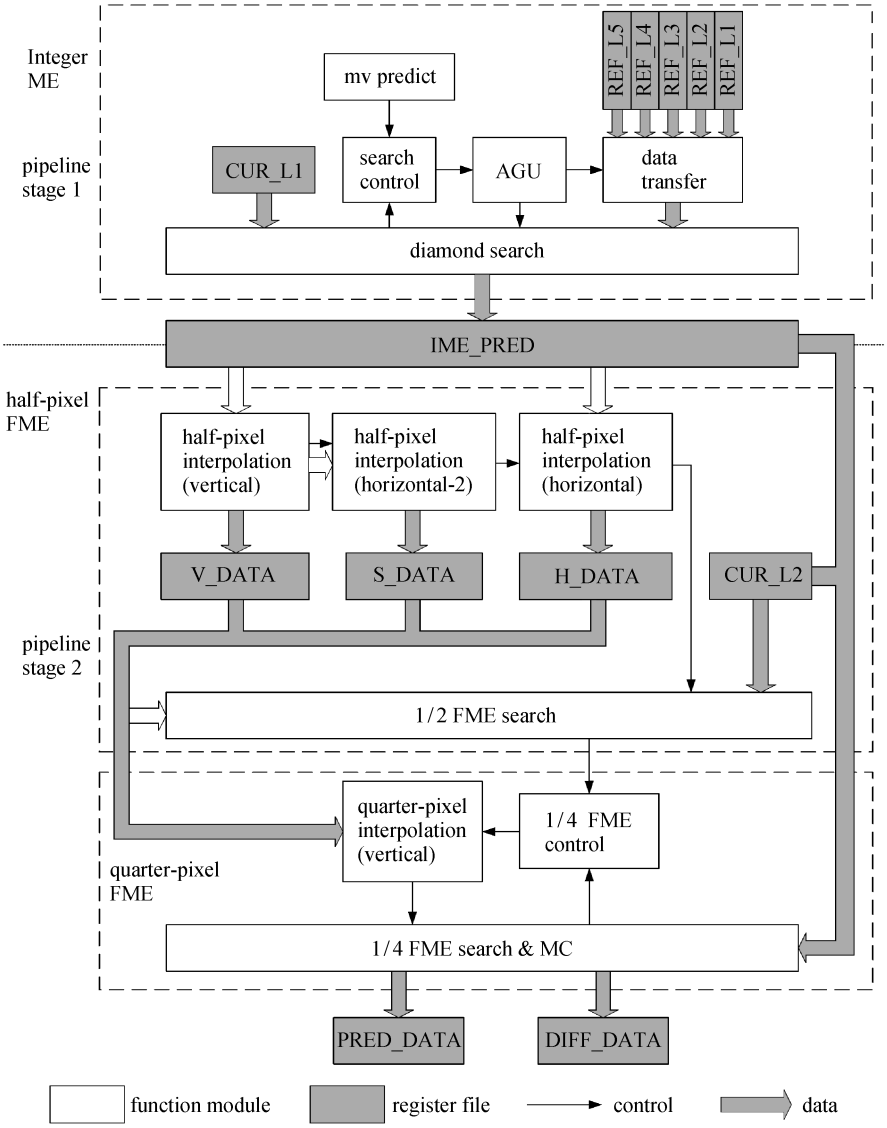


Fig. 5 Block diagram of proposed architecture for motion estimation

REF_L1–L5 are used to store the search window. The function of AGU (Address Generation Unit) is to generate and deliver the address of reference block to search module. The searching process starts at the pixel where the predicted MV points, and searches with DS strategy, outputting the best integer MV and a predicted 22×22 block for half-pixel interpolation, which is stored in IME_PRED according to the addresses from AGU.

The second stage is FME. CUR_L2 stores the current block for FME to process. The predicted block in IME_PRED passes through the first and second set of vertical half-pixel interpolation, and then horizontal interpolation, with interpolated samples stored in V_DATA, T_DATA and H_DATA, respectively. Meanwhile, 1/2 FME search module is activated to detect the best half-pixel position. After that, quarter-pixel samples are produced and evaluated, to get the final MV and predicted block, which is buffered in PRED_DATA. Motion compensation is included, too, outputting the residual data to DIFF_DATA.

2.1 Prediction of motion vector

The standard procedure of inter prediction requires three steps of operation, as shown in Fig. 6(a). After IME and FME, it also requires the result of inter/intra prediction selection to calculate the final MV, which is then passed back to IME, so that the MV of next block can be predicted. This procedure leads to a feedback circle, breaking the data pipelining.

The optimized structure is presented in Fig. 6(b). In order to remove the feedback, MV prediction is conducted twice in stage 1 and stage 3. The first prediction is used to predict the MV of next block, and the other one is to calculate the actual predicted MV, which will be subtracted from final MV, generating the to-be-encoded vector difference. The side effect of this modification is that, the predicted MV could be warped a little from the original one, but it can be corrected by motion estimation. Besides, the predicted MV before modification is not definitely better, since it's also simply guessed based on the motion vectors of nearby encoded blocks.

2.2 SAD structure

When searching with DS strategy, SADs of three macro-blocks need to be calculated at each step. If the maximum of searching step is defined as 16, IME needs to calculate 48 SADs within the constrained time, which makes SAD calculation the bottleneck of IME. Furthermore, FME also contains the calculation of 16 SADs. Encountering with this problem, SAD structure is improved, such as Propagate Partial SAD (PPSAD)[14], or replaced by new algorithms[15] in recent researches.

The SAD structure used in this paper is demonstrated in Fig. 7. With the input bandwidth of 256 bits, it reads 16 predicted pixels and 16 to-be-predicted pixels of the current macro-block at

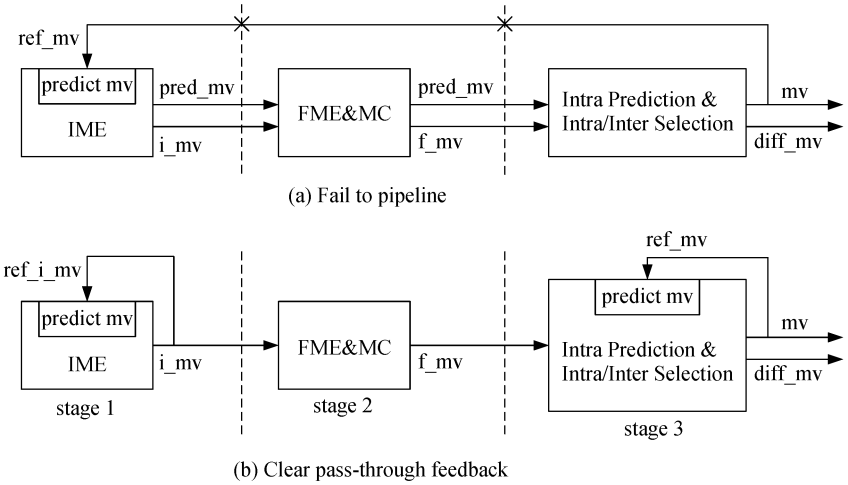


Fig. 6　Motion vector prediction implementation

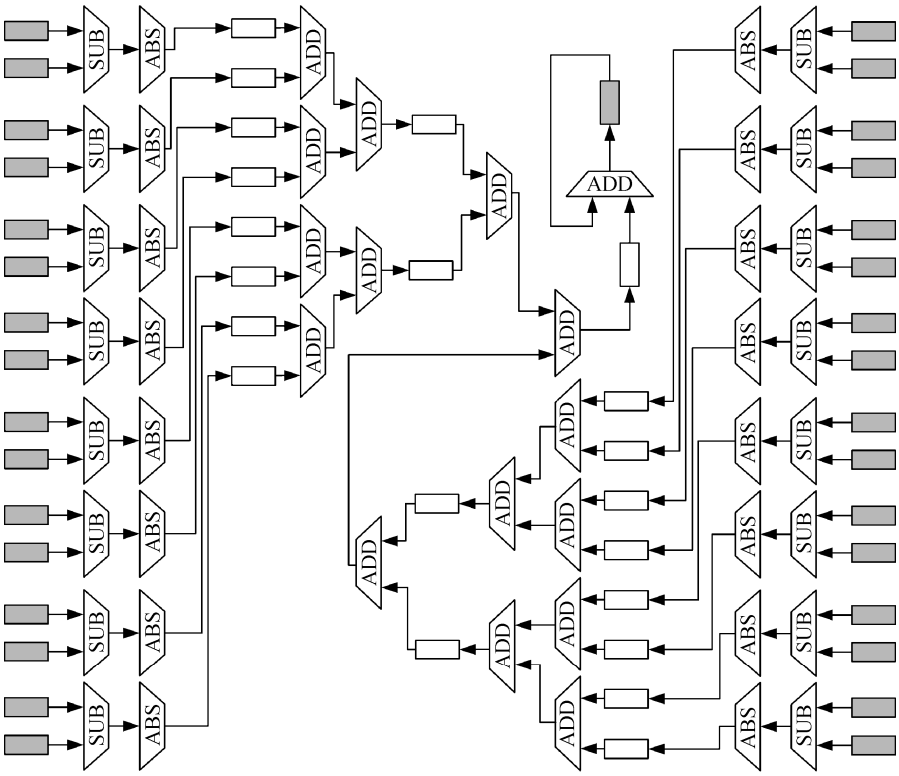


Fig. 7 SAD structure

each clock period, calculates the absolute values of subtraction, and then adds them up by 4-input adders, which are implemented with a 4-to-2 compressor and 2-input adder. The final SAD of a block is generated after 16 times of accumulation. Registers are inserted to decrease the combinational delay. With full pipelining, SAD calculation of a macro-block costs only 16 periods. Using this structure, SAD will cost no more than 772 periods when the search depth is 16, which fulfils the timing requirement of IME.

2.3 Interpolation components

969 half-pixels and 2048 quarter-pixels need to be generated at FME stage. For hard-wired implementation, shifting is more efficient than multiplying. Thus Eq. (1) and Eq. (2) can be transformed as following:

$$b = \{(E+J)-[(F+1)\ll 2+(F+1)]+[(G+H)\ll 4+(G+H)\ll 2]+16\}\gg 5, \quad (4)$$

$$a = (G+b+1)\gg 1. \quad (5)$$

Fig. 8(a) and 8(b) present the hardware structure of half-pixel and quarter-pixel interpolation, respectively. The half-pixel interpolation component is accordant to Eq. (4), with an input bandwidth of 6 bytes and output bandwidth of 1 byte. Two sets of this component are implemented, working in parallel, to meet the timing requirement.

The quarter-pixel interpolation component contains 16 structures accordant to Eq. (5), with an input bandwidth of 32 bytes and output bandwidth of 16 bytes. It costs only 144 clock cycles to fulfil the interpolation of quarter-pixel samples within FME and MC, which is fast enough for the real-time demand.

The interpolation of chroma sub-pixel makes a lot use of multipliers, which can also be conducted with adders and shift operations. Take the interpolation for chroma sub-pixel at position (2, 3) as an example:

$$w(2,3) = [(30G+10I+18R+6V+32)/64] = [5(3G+I)+3(3R+V)+16]\gg 5. \quad (6)$$

In Eq. (6), two expressions of $3x+y$ and $5x+3y$ can be abstracted, which can be realized with no multipliers as:

$$3x+y = (y-x)+x\ll 2, \quad (7)$$

$$5x+3y = (x-y)+(x+y)\ll 2. \quad (8)$$

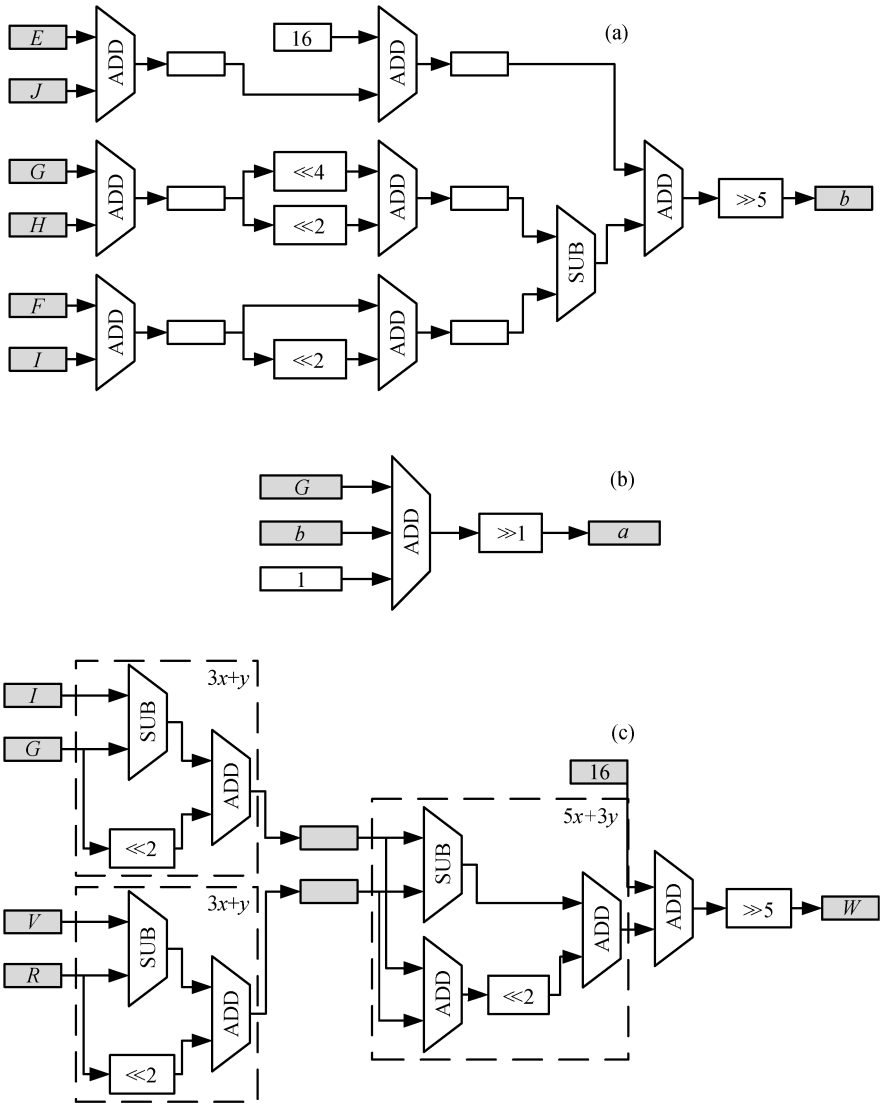
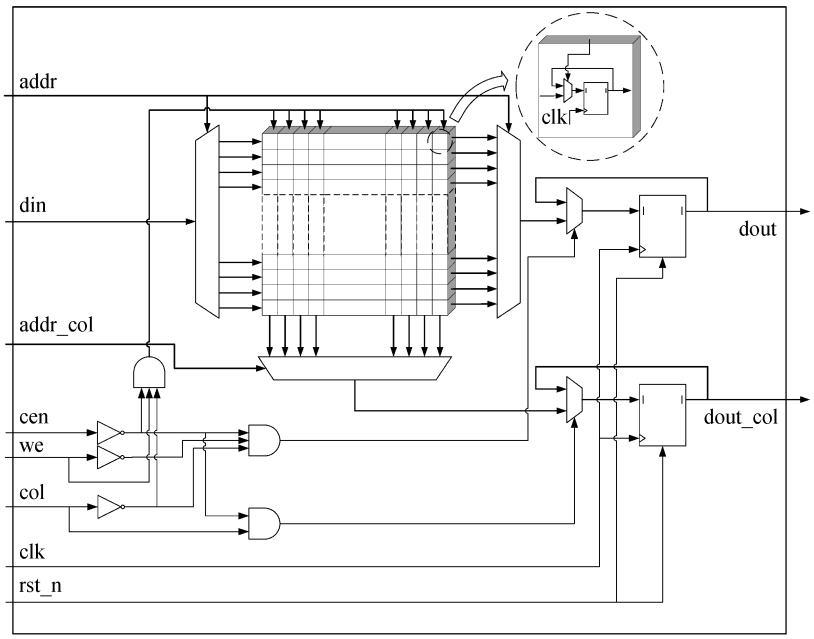


Fig. 8　Hardware structure for half-pixel (a), Quarter-pixel (b) and 1/8 chroma interpolation (c)

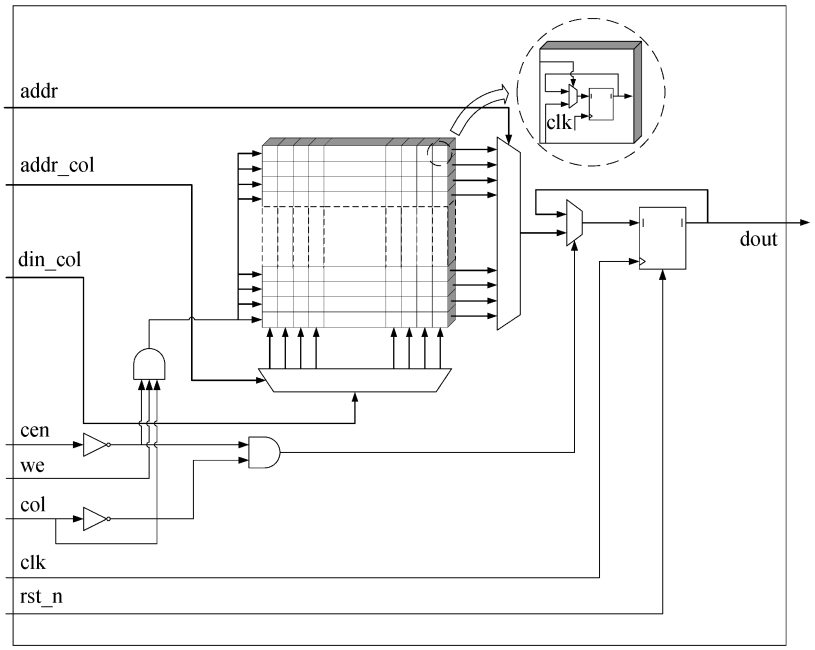
The supposed hardware structure of the interpolation for chroma sub-pixel at position (2, 3) is demonstrated in Fig. 8(c). Actually, as we have proposed in Ref. [16], four arithmetic elements with only adders and shift operations can fully support the calculation of chroma sub-pixels at any position with a two-stage pipelining structure.

2.4　Memory structure and pipeline within FME

In the operation of horizontal interpolation for half-pixel samples, 6 continuous pixels in a row are read from memory IME_PRED, while in the operation of vertical interpolation, pixels adjacent in a column are read. Using a conventional two-port register file module that can be read only by row, it will take at least 6 clock cycles to obtain the input data for vertical interpolation, which is even longer than the interpolation operation itself. This will lead to an intermitted pipeline, which is unacceptable to the timing constraint. In this case, a manual-build memory module IME_PRED is introduced to the architecture, with the details shown in Fig. 9(a). At the center of IME_PRED lies a $22 \times 22 \times 8$ bit register array, which stores the predicted data from integer motion search for half-pixel interpolation. IME_PRED is written by IME through port din. Unlike conventional memory



(a) IME_PRED



(b) V_DATA

Fig. 9 Novel structures for efficient memory access

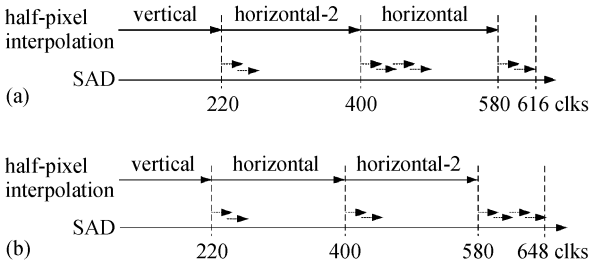
modules, data can be read out through two output ports, dout and dout_col, in which dout is connected to horizontal interpolation and dout_col is connected to vertical interpolation. With this structure, memory would be accessed only once for vertical interpolation, which will enhance the parallelism of the interpolation procedure largely. Compared with the original memory structure, this will greatly accelerate the process of FME.

For vertical interpolation, not only its inputs are lined in column, its outputs are also arranged vertically, which are then stored in V_DATA. However, the format of data in V_DATA cannot be changed because it will be delivered to SAD module and quarter-pixel interpolation module in rows. Thus, to avoid wasting time in writing V_DATA, a similar

modification has been applied to V_DATA as well. Differ from IME_PRED, V_DATA doesn't need the extra dout_col output, but requires an input port din_col that can write the register array by column, which means V_DATA has an input port of din_col and an output port of dout, as shown in Fig. 9(b).

Considering the secondary interpolation for half-pixel samples, i.e., position *j* in Fig. 4(a), they can be calculated by interpolating either between six vertical or horizontal half-pixel samples from the first set of operations. However, the better choice is horizontal interpolation between the vertical half-pixel samples from the first set of operations. In this case, the data in V_DATA will be accessed in rows, while if the horizontal half-pixels samples from the first set of operations are used for the secondary interpolations, the data in H_DATA will have to be accessed in columns, which requires an extra modification of H_DATA. Thus, there are 17×22=374 vertical half-pixel samples, 17×18 = 306 horizontal half-pixel samples and 17×17=289 secondary half-pixel samples to be interpolated and the capacity of V_DATA, H_DATA, S_DATA is 374 bytes, 306 bytes and 289 bytes, respectively.

After the memory modifications for vertical interpolation, taking SAD calculation into consideration, the pipeline schedule of half-pixel interpolation is illustrated in Fig. 10. Clock cycles spent on the step of vertical interpolation is not much more than the other two steps, because of the improved memory structure. To make the process more efficient, interpolation and SAD calculation are pipelined. Since four SADs need to be conducted after the secondary interpolation, the secondary interpolation should be conducted before the horizontal interpolation. In this case, the four SADs after secondary interpolation can be calculated in parallel with the horizontal interpolation, as shown in Fig. 10(a), which will save 32 clock cycles. That is, samples of half-integer *y*-coordinate are generated by vertical interpolation first, and then are used immediately in the secondary interpolation. Finally



(a) Secondary interpolation first; (b) horizontal interpolation first

Fig. 10　Pipeline scheduling

samples of half-integer *x*-coordinate are produced by horizontal interpolation. This scheduling can also reduce memory accesses, which helps to accelerate the interpolation process even further.

3　Implementation Results

The proposed architecture for ME is integrated in our H.264/AVC encoder, the specification of which is baseline profile with level up to 3.1. The processing capability of the encoder is real-time coding HDTV 720P 60 fps video with one reference frame.

For FPGA prototyping, the design is mapped to a XC6VLX240T FPGA device on the DN-DualV6-PCIe4 Xilinx Virtex6 Logic Emulation Board with a clock frequency of 52.7 MHz. Six video sequences, some with slow and regular motion (Stockholm_ter and Shields_ter) and others with fierce and irregular movements, are encoded with and without ME, and the simulation results are compared in Table 1. As shown in Table 1, with the slow-motion video, inter prediction is selected in more than 80% of all macroblocks, while less macroblocks within the fierce-motion video are encoded with inter prediction. Table 1 also shows that inter prediction brings a 75% reduction of bitstream length for the slow-motion video (Stockholm_ter), costing only 1.2 db decrease of PSNR. The compression ratio can actually reach 1:43 with a quantization parameter (QP) of 26. Even dealing with the more fierce video with plenty details (Ducks_take_off), the bit-rate saving can still reach 55%, and the PSNR drop is limited within 1.4 dB. This proves that ME can truly improve the encoding

Table 1 Simulation results with different video sequences

Video sequence	MBs with inter prediction/%	Bitstream length/(KB·frame^{-1})			PSNR/dB		
		Without ME	With ME	Decrease by/%	Without ME	With ME	Decrease by/%
Stockholm_ter	81.5	131.01	32.05	75.54	37.09	35.88	3.26
Ducks_take_off	71.8	204.26	90.40	55.74	37.24	35.84	3.76
Shields_ter	84.6	154.48	31.20	79.80	38.10	37.03	2.81
In_to_tree	76.4	100.32	21.96	78.11	38.13	37.37	1.99
Park_joy	72.7	223.64	102.48	54.18	37.58	36.22	2.62
Sintel_trailer	64.0	7.00	3.28	53.14	57.55	54.85	4.69

efficiency significantly, by lowering the bit-rate without evident quality loss, which is essential for real-time video transmission, and thus is not dispensable despite its complexity. Fig. 11 presents one of the original frames from the slow-motion video sequence before encoding and the residual data of the same frame after inter prediction.

To compare with other VLSI architectures, the proposed design of IME and FME has been synthesized with SMIC 130 nm CMOS technology with a clock frequency of 200 MHz, consuming 202 K logic gates and 13.8 K bytes SRAM, as shown in Table 2. As compared with Chen et al[17], our design can support a 2 times larger video specification with less hardware cost. Furthermore, only IME is fulfilled in the work of Chen et al[17]. As compared with Koziri et al[15], the proposed architecture can support a much larger search range with only half of the logic gates and lower operating frequency for the same video specification except that the maximum number of reference frames of Koziri's work is 2. Although Tsai et al.[1] proposed the architecture with a large search range and very low operation frequency, the gate count is 1.5 times larger than that of ours, without considering that the fast algorithm used in Tsai's work may lead to distortion and PSNR drop. Since diamond search algorithm is taken while conducting integer motion estimation, the proposed design achieves a hardware efficiency of 7608 compared with that of 6521 in Ref. [18]. Considering that only IME is conducted in Ref. [18], the hardware efficiency of our design is even higher.

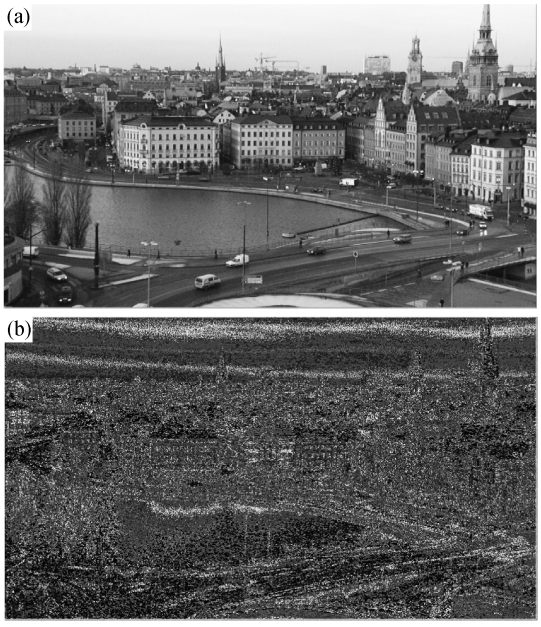


Fig. 11 Original frame before encoding (a) and residual frame after inter prediction (b)

According to Table 3, the proposed design achieves the HDTV(720P@60fps) encoding with the lowest hardware cost. Actually, the logic gate count for SAD calculation and sub-pixel interpolation is even smaller than 202 K, since the optimized memory structure for IME_PRED and V_DATA is build up with separated flip-flops, which consumes 36.3 K and 19.4 K logic gates, respectively. The SAD structure comprised of compressors, interpolation components build up by adders only and the optimized memory structure for full-pipeline operation are the key factors that contribute to the reduced hardware cost.

The overall H.264/AVC baseline profile encoder chip with the proposed motion estimation module is finally implemented with SMIC 65 nm 1P8M high-

Table 2　Performance comparison with different VLSI architectures

Items	Proposed	Ref. [17]*	Ref. [15]	Ref. [1]	Ref. [18]*
Profile	Baseline	Baseline	Baseline	Baseline	Baseline
Process/μm	SMIC 0.13	UMC 0.18	UMC 0.13	UMC 0.13	UMC 0.09
Gate count/K	181.7	305	370	300	30
Frequency/MHz	167	108	241	116.6	250
Max Spec.	720P@60fps	720P@30fps	720P@60fps	3840x2160@30fps	720P@30fps
Max SR.	80 × 80	128 × 64	16 × 16	256 × 256	49 × 49
Max Ref.	1	1	2	1	1

* Only IME is conducted in Ref. [17] and [18].

Table 3　Specification of the overall H.264/AVC encoder chip with the proposed ME module

Technology	SMIC 65 nm 1P8M CMOS(RVT, HS)
Pad/core voltage	2.5/1.0 V
Core area	1.74 mm×1.74 mm
Gate count	790.5 K
Memory	25.3 KB
Max frequency	350 MHz
Work frequency	200 MHz
Encoding features	Baseline profile, Level 1-3.1
Spec.	720P@60 fps
# of Ref. frame	1
Max SR.	H[–40, 39] V[–40, 39]

speed (HS) regular-threshold-voltage (RVT) CMOS technology process. As shown in Table 3, the whole chip consumes 790.5 K logic gates and 25.3 K bytes SRAMs with a maximum frequency of 350 MHz, in which the proposed motion estimation module takes 276.5 K of the logic gates and 13.8 K bytes of the memories. It should be noted that besides de-blocking (DB) filter and CAVLC entropy coding[19], both 16×16 intra prediction and 4×4 intra prediction[20] as well as network abstraction layer (NAL) coding are implemented within the proposed encoder, which makes IME and FME takes only 1/3 of the logic gates of the overall chip. With the proposed H.264/AVC encoder, an operating frequency of 200 MHz can fulfil the real-time HDTV (720P@60fps) video encoding. Fig. 12 presents the layout photo of the encoder, in which IME, FME and the related memories, especially IME_PRED and V_DATA, are highlighted.

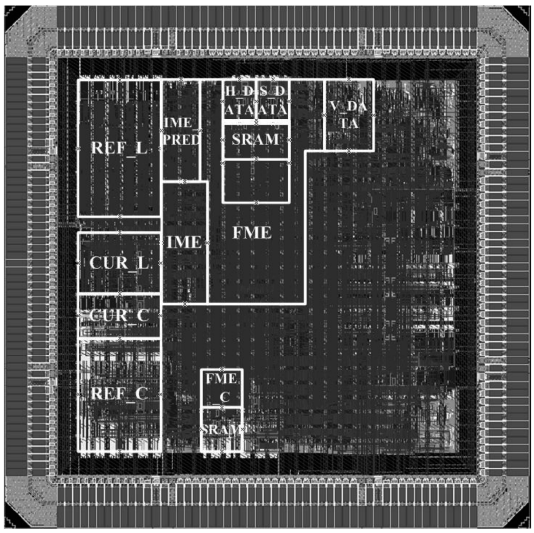


Fig. 12　Layout photo of the proposed encoder

4　Conclusion

As an important part of H.264/AVC standard, motion estimation plays a key role in eliminating temporal redundancy and improving encoding efficiency. However, due to the complexity of algorithm and the exhaustiveness of computation, it's not easy to be integrated into real-time encoders. In this paper, by modifying the procedure of motion vector prediction, pipelining integer and fraction motion estimation, choosing the appropriate searching strategy, designing fast and efficient interpolation components, and optimizing data storage structure, an hardware efficient architecture for motion estimation in H.264/AVC encoders is proposed, which can fulfil the real-time HD (1080P@60fps) video encoding with a clock frequency of 350 MHz. The proposed design has

been integrated within an H.264/AVC encoder for FPGA prototyping and VLSI implementation. The core area of the overall chip is 1.74 mm×1.74 mm with SMIC 65 nm CMOS technology while the proposed motion estimation module takes 276.5 K of the logic gates and 13.8 K bytes of the memories.

References

[1] Tsai T H, Pan Y N. High efficiency architecture design of real-time QFHD for H.264 fast block motion estimation. IEEE Trans Circuits Syst Video Technol, 2011, 21(11): 1646–1658

[2] Ostermann J, Bormans J, List P, et al. Video coding with H.264/AVC: tools, performance, and complexity. IEEE Circuits Syst Mag, 2004, 4(1): 7–28

[3] Choi W I, Jeon B, Jeong J. Fast motion estimation with modified diamond search for variable motion block sizes // Proc of ICIP'03. Catalonia, 2003, 2: 371–374

[4] Li D, Ku C, Cheng C, et al. A 61 MHz 72 k gates 1280×720 30 fps H.264 intra encoder // Proc of ICASSP. Honolulu, 2007: 801–804

[5] Lin Y K, Ku C W, Li D W, et al. A 140-MHz 94 k gates HD 1080p 30-frames/s intra-only profile H.264 encoder. IEEE Trans Circuits Syst Video Technol, 2009, 19(3): 432–436

[6] Loukil H, Atitallah A B, Kadionik P, et al. Design implementation on fpga of H.264/avc intra decision frame // Conf Des Technol Integr Syst Nanoscale Era (DTIS'10). Hammamet, 2010: 1–4

[7] Huang Y W, Hsieh B Y, Chien S Y, et al. Analysis and complexity reduction of multiple reference frames motion estimation in H.264/avc. IEEE Trans Circuits Syst Video Technol, 2006, 16(4): 507–522

[8] Tsai T H, Pan Y N. A novel 3-d predict hexagon search algorithm for fast block motion estimation on H.264 video coding. IEEE Trans Circuits Syst Video Technol, 2006, 16(12): 1542–1549

[9] Pan Y N, Tsai T H. Fast motion estimation and edge information inter-mode decision on H.264 video coding // Proc of ICIP'07. San Antonio, 2007, 2: 473–476

[10] Bi H J. New video compression standard-H.264/AVC. 2nd ed. Beijing: Posts & Telecom Press, 2009: 34

[11] Li R, Zeng B, Liou M L. A new three-step search algorithm for block motion estimation. IEEE Trans Circuits Syst Video Technol, 1994, 4(4): 438–442

[12] Zhu S, Ma K K. A new diamond search algorithm for fast block-matching motion estimation // Proc of ICICS'07. Singapore, 1997: 292–296

[13] Zhu C, Lin X, Chau L P, et al. A novel hexagon-based search algorithm for fast block motion estimation // Proc of ICASSP'01. Salt Lake City, 2001, 3: 1593–1596

[14] Huang Y, Liu Z, Goto S, et al. Cost efficient propagate partial sad architecture for integer motion estimation in H.264/avc // Proc of ASICON'07. Guilin, 2007: 782–785

[15] Koziri M G, Dadaliaris A N, Stamoulis G I, et al. A novel low-power motion estimation design for H.264 // Proc of ASAP'07. Montreal, 2007: 247–253

[16] Wang T, Zhao L, Hu Z Y, et al. A hardware efficient implementation of chroma interpolator for H.264 encoders // Proc of EDSSC'11. Tianjin, 2011: 1–2

[17] Chen T C, Chien S Y, Huang Y W, et al. Analysis and architecture design of an HDTV 720p 30 frames/s H.264/avc encoder. IEEE Trans Circuits Syst Video Technol, 2006, 16(6): 673–688

[18] Chen Y B, Chen Z D, Guo L, et al. Architecture design of low-power motion estimation based on DHS-NPDS for H.264/avc. Sci China, Ser F: Inf Sci, 2012, 55(10): 2234–2242

[19] Hu Z Y, Chen K L, Wang X A. Operator design methodology and implementation for H.264 entropy encoder // Proc of ICIECS'10. Wuhan, 2010: 1–4

[20] Hu Z Y, Peng J H, Zhang X, et al. Implementation of intra prediction in H.264 based on a novel design methodology // Proc of CSAE'11. Shanghai, 2011: 650–655