doi:10.3772/j.issn.1006-6748.2014.03.015

Research on variable block size motion estimation algorithm for airborne image①

Wang Ke (王 科)②[*,**], Huang Dongshan[*], Ma Li[*,**], Zhang Yudong[*]
([*] China National Aeronautical Electronic Radio Research Institute, Shanghai 200233, P. R. China)
([**] School of Electronic, Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai 200233, P. R. China)

Abstract

A fast motion estimation algorithm for variable block-size using the "line scan and block merge procedure" is proposed for airborne image compression modules. Full hardware implementation via FPGA is discussed in detail. The proposed pipelined architecture based on the line scan algorithm is capable of calculating the required 41 motion vectors of various size blocks supported by H.264 within a 16×16 block in parallel. An adaptive rate distortion cost function is used for various size block decision. The motion vectors of adjacent small blocks are merged to predict the motion vectors of larger blocks for reducing computation. Experimental results show that our proposed method has lower computational complexity than full search algorithm with slight quality decrease and little bit rate increase. Due to the high real-time processing speed it can be easily realized in hardware.

Key words: H.264, motion estimation, line scan algorithm, rate distortion optimization (RDO), FPGA, block merge method

0 Introduction

H.264 is a video coding standard proposed by Join Video Team (JVT). Similar to old video coding standards, H.264 adopts a motion estimation algorithm based on the block matching method to eliminate redundant information between frames. Motion estimation module is still the most time-consuming during the process of H.264 coding. Work station SunBlade2000 with 1.015GHz processor and 8GB memory space is used for the H.264 coder software simulation. In order to get a real time processing target, the system needs to be of a computing capability of over 300 giga-instructions per second (GIPS), while data transmission bandwidth needs to be over 460Gbyte/s[1]. A great amount of video data needs to be recorded in real time in aircraft electronic systems. With the increasing data processing resolution in airborne images, video compression recording module realized by software which were used before is hard to meet in terms of real-time requirement. Because of the high calculation amount in variable block size motion estimation (VBSME) algorithm, many hardware algorithm architectures[2-5] are proposed for accelerating VBSME calculation.

The features in airborne video images are hugely different with normal films and television images. A number of figures, lines and graphic characters with all kinds of sizes are added in airborne images. This information shows important flying status and parameters, needed to be recorded fully and clearly, which means details and distinctions of outlines should be considered of during the compressing process. Items' motion characteristics in airborne video are also hugely different from the ones in natural scenery video. Most of the video compressing algorithms used in common business are based on natural scenery images, which do not work effectively in aircraft electronic systems. This research proposes a new motion estimation algorithm based on H.264, which is suitable for airborne image real-time compression. Hardware implementation based on FPGA is also carried out.

1 Algorithm of line scan and block merge

1.1 Variable block size motion estimation based on H.264

H.264 adopts block matching motion estimation technique with variable blocks. Each macro block (MB) can be divided into 7 sub-blocks for the matching. Each 16×16 micro block has 4 ways for division: 16×16, 16×8, 8×16 and 8×8. For the 8×8

① Supported by the Aviation Science Fund of China(2009ZC15001).
② To whom correspondence should be addressed. E-mail: edua714@sina.com
Received on Feb. 21, 2013

blocks, a further sub-blocks division method, 8×8, 8×4, 4×8 and 4×4, can be carried out. Generally speaking, the smaller the sub-blocks are, the more accurate the matching is, and the more effective the coding is. However, the more the sub-block division is, the more computing complexity is, which is brought by searching processes in motion estimation[6]. Most of the fast search algorithms reduce the computing load by reducing reference position in ergodic searching windows. Most of these fast algorithms are based on some assumptions: the further distance between reference position and best matching position is, the bigger corresponding matching error is; the probability that motion vectors distribute on the horizontal direction is higher than the vertical direction; most of the motion is centralized in the centre area of the image. Experiments show that these assumptions are correct for natural scenery. Generally, these searching methods take vector (0, 0) as initial starting point, then search all around, first crudely, then finely. Typical methods include three-step search algorithm, four-step search algorithm, diamond search algorithm[7] and hexagon search algorithm[8], etc.

However, many airborne images do not have the characteristics in the assumptions. Airborne display images consist of a foreground and an overlaid background. There are many small objects such as figure and characters' severely shifting on a vertical direction. The object motion directions in the image follow a random distribution. There are always moving object targets shown at the edge area of the images. If traditional fast search algorithm is still being applied, either a very long searching path will be led, or local optimums will be fallen into. This research proposes a motion estimation algorithm based on the "line scan and block merge" method. One time scanning can calculate motion vector (MV) and sum of absolute difference (SAD) in one vertical line in the searching windows. Only moving scanning lines on a horizontal direction is needed during the searching, which can hugely reduce number of calculations, and is suitable for hardware implementation.

1.2 Line scan algorithm and architecture

H.264 adopts a rate distortion optimization (RDO) technology, in which both code rate and degree of distortion are considered during computation of the cost function for rate distortion (RD). An appropriate compromise is chosen. During the pattern selection process, every possible pattern's RD cost value, J_i are compared, and the pattern with the smallest cost value J is chosen for the best pattern. The algorithm in this research uses Eq. (1) as the cost function for pattern chosen:

$$J_i = SAD_i + \lambda_{\text{motion}} \times R(MVD_i) \tag{1}$$

In which, SAD is defined as follows:

$$SAD = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} | f_1(x,y) - f_2(x + mvx, y + mvy) | \tag{2}$$

λ in Eq. (1) is a Lagrange multiplier, its initial value is approximately calculated with Eq. (3), which is realized by table look-up in hardware. MVD is the difference between this Macro Block's ultimate MV and motion vector prediction (MVP). R is the number of bits in the binary system after MVD entropy coding. The value of R can be also gotten by table look-up in hardware realization.

$$\lambda = \sqrt{0.85 \times 2^{\frac{QP-12}{3}}} \tag{3}$$

N and M in Eq. (2) can be 4, 8 and 16 respectively, f_1 and f_2 are current pixel and reference pixel respectively, (x, y) is current pixel's coordinate position in the current frame, and mvx and myv are coordinates in x and y directions for motion vectors. The larger the SAD value is, the more severely moving this area is, of the image. On the contrary, this means a relatively motionless flat area. Most of the airborne images backgrounds are relatively motionless digital maps images, and most of the image details such as characters, lines and image units are overlaid as foreground. Because in practical application, foreground overlaid timings are known, the coefficient λ[9], which is used in computing cost function in motion estimation module, can be changed to self-adapt according to difference between the types of overlaid images. The smaller the coefficient is, the more attention is paid to the quality of the images. However, before overlaying, a bigger coefficient λ is adapted to get lower code rate.

In most of the old block matching algorithms, each computing is to search whether a corresponding macro block is matching according to some point's MV. However with the adoption of the computational array in Fig. 1, one line's related MVs are searched every time, namely the "line scan" searching method. MVX, MVY are used to describe motion vectors' components in X and Y directions, using pixels as the unit. Searching scope $MVX \in [-16,15]$, $MVY \in [-16,15]$.

Computational array is composed of 32 parallel processor element (PE) units, where each PE finishes RD cost value J calculations for 16 original points and 16 reference points. Because this architecture provides 48 reference points' input computational array in one clock cycle, it can carry out a list of data comparisons

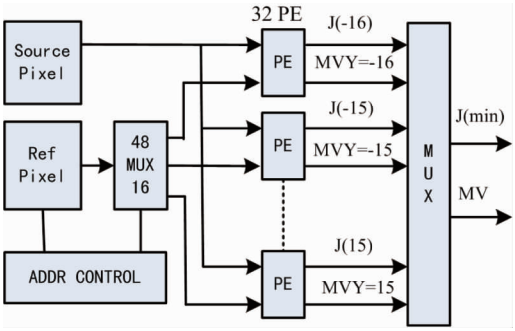


Fig. 1 Architecture of PE computational array

which equals the height of a whole searching window, and is able to accomplish 32 surrounding *MVY* searches for a given *MVX* for one time. A minimum value is screened out from these 32 output cost values *J*, and corresponding *MVY* is estimated according to the position of this minimum value. A group of related *MVY* computations can be completed for a given *MVX*, namely the computations for a scanning line. This architecture can flexibly adjust the PE unit array according to different resolution ratios, in order to satisfy requirements for different video band widths and speeds.

Line scan algorithm proposed in this manuscript is shown in Fig. 2. The over-searching window is of the size of 3 × 3 MBs. A rectangular coordinate system is adopted, and the original point is assumed at coordinate (0, 0). Towards the right is the forward direction for *X*, while downward is the forward direction for *Y*. The coordinate for the top left corner is (−16, −16). The scanning line is defined as coordinates (*MVX*, −16) ~ (*MVX*,15), corresponding to every vertical line in the figure. For example, scanning line at position *MVX* = −8 is corresponding to coordinates (−8, −16) ~ (−8, 15). To reduce searching times and save computing time, a line scan binary algorithm is designed in this research. For more accurate searching, scanning lines hop at half of original step length.

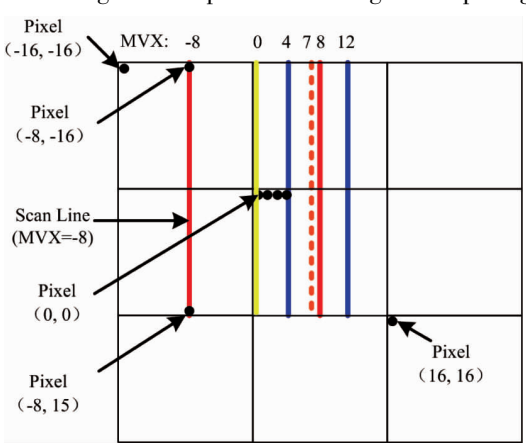


Fig. 2 Bisection algorithm of line scan in search window

In hardware implementation, hopping of the scanning line corresponds to changing of storage address, for the convenience of getting data for the computing unit from caches.

According to the circuit architecture described in Fig. 1, FPGA hardware parallel computation characteristics are fully used for the implementation of the algorithm in Fig. 2. In the first clock cycle Clk0, the computing unit receives 16 original pixel points, $D(mvx + 0, 0 \sim 15)$, 48 reference pixel points, $R(mvx + 0, -16 \sim 31)$. In Clk1, pixel data to be gotten horizontally moves one pixel position to the right, and original points are $D(mvx + 1, 0 \sim 15)$, while 48 reference points are $R(mvx + 1, -16 \sim 31)$. Same applies, in Clk15, and original points are $D(mvx + 15, 0 \sim 15)$, while 48 reference points are $R(mvx + 15, -16 \sim 31)$. 16 cost values *J*, corresponding *MV*, and the sum of 16 *J*, J_{sum}, are saved in line scan module output caches. J_{sum} is used to decide the moving direction for the scanning lines.

The line scan binary algorithm's steps are shown as follows: The first scanning line is at the position of $MVX = 0$. After several clk computations, 16 cost values, *J* and 16 corresponding values, $MV(x = 0, y \in [-16,15])$, which are corresponding to vertical line at position $MVX = 0$, are exported. Summation of these 16 *J*, the scanning line's J_{sum} is gotten, as the foundation of the scanning line's motion direction. The initial searching step length is set to half of the searching scope. The second searching line is at $MVX = 8$. The PE unit compares each of the 16 *J* saved in caches last time while computing, updates the smaller *J*, and updates corresponding MVs as well. Same applies, after several clk computations, 16 cost values *J* and 16 corresponding MVs are exported for the vertical line at $MVX = 8$. Comparing this line's $J_{sum}(MVX = 8)$ with previous one, if this J_{sum} is smaller, then the scanning line is moved to position $MVX = 8$. The third scanning line is at $MVX = -8$, with the same steps applied, this line's J_{sum} is calculated, and compared with previous J_{sum} to determine moving direction for the scanning line. Suppose J_{sum} for the scanning line at $MVX = 8$ is even smaller, then line $MVX = 8$ is taken as the centre, step length is reduced to half, and the searching lines are continued at $MVX = 4$ and $MVX = 12$. And likewise, suppose centre line moves to $MVX = 4$, then lines at $MVX = 2$ and $MVX = 6$ are searched. Suppose the centre line moves to $MVX = 6$, then lines at $MVX = 5$ and $MVX = 7$ are searched, till the step length is reduced to a single pixel, namely line $MVX = 7$ is chosen eventually. Totally there are 9 searches performed, and eventually the binary scanning process is finished.

Following, the block merge will compare MV caches according to the block merge policy. The best division method for this macro block will be exported eventually, as well as corresponding MVD. For the scanning process described above, according to airborne image empirical statistics, cost value threshold, T_{hreshold}, for this research is set as 1024, namely as long as $J_{\text{sum}} < t_{\text{hreshold}}$ is gotten for any line scanning in these 9 times, then this line scanning is considered the best, and the scanning process will stop before the schedule.

1.3 Block merge policy

During the H.264 coding process, every possible division for the macro block will get motion compensation, and then decision making will be based on the rate distortion optimization (RDO). RD cost values will be computed for each division, and comparisons will be carried out. Eventually, the division with the smallest cost value will be chosen as inter mode. Nevertheless, the more the block division mode configurations are, the more effectively encoding is performed. But this will increase computing complexity during the coding process, which will hugely reduce coding speed. This is not preferable for practical applications, especially for the occasions with high real time requirements.

When bigger size blocks are adopted for coding, residual errors after motion compensation are big normally and relatively. But the number of bits for coding motion vector (MV) and block types will be smaller, which is suitable for images with fewer details and stable areas. When smaller size blocks are adopted for codding, estimated residual errors are relatively smaller, and a better estimate accuracy is gotten. But number of bits for MV and block types will be increased. This is suitable for images with rich details/high resolution, and severe motion areas. Because airborne images contains a great amount of important figure information such as lines and characters, a high matching accuracy is required for the system. In order to satisfy these special requirements for airborne images, real time requirements in hardware implementation need to be satisfied as well, such as reusing computed SAD results for small blocks. This paper used bottom-up block merge policy[10,11]. We take small size 4×4 as initial mode configuration, and then check up rest 6 mode configurations from small to large according to their size. According to predefined block merge policy, it is determined whether small size blocks are needed to be merged into bigger size. If merging is needed, this division type will be saved after. If no merging is needed, small size mode configuration in last level will be kept as the best. Further searching will be terminated.

For four 4×4 small blocks in 8×8 block, namely block 1, block2, block 3 and block 4 in Fig. 3, MV for big size block after merging is related to MV for small blocks before merging. While the process for 8×8 block merging into 8×16, 16×8 and 16×16 are similar, unnecessary repetitions are avoided. Merge policy for 4×4 block into 4×8, 8×4 and 8×8 are shown as follows:

(a) If MV for blocks 1 and 3 in this figure are equal, and MV for blocks 2 and 4 are equal, then they will be merged into 4×8 type on the left side, namely block 5 and 6.

(b) If MV for blocks 1 and 2 in this figure are equal, and MV for blocks 3 and 4 are equal, then they will be merged into 8×4 type on the right side, namely block 7 and 8.

(c) Under the situation that neither (a) and (b) condition is satisfied, if MV for any two blocks in the four are equal, then they will be merged into 8×8 type, namely block 9.

(d) If neither of (a), (b) and (c) conditions are satisfied, the original 4×4 small block division method will be maintained.

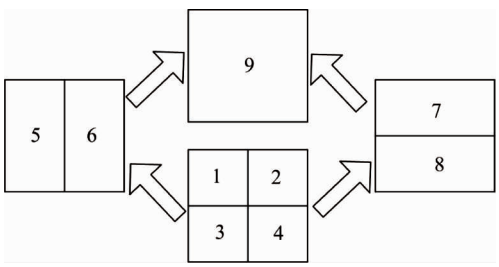


Fig. 3 Block merge of 4×4 to 4×8, 8×4 or 8×8

2 Experimental results and analysis of algorithm

For search window with search zone as W, a traditional searching algorithm will need $(2W + 1)^2$ passes. The line scan binary algorithm proposed in this paper reduces searching times hugely. Only 9 line moves can reach a search zone as $[-16, +15]$, and computational points are reduced by 73% compared with full search. Simulation is carried out with a personal computer with Intel Core2 E6550 2.33GHz, 2GB storage. Standard C based on identifying code is used with Visual C++6.0. Natural scenery image "Forman" with QCIF(176×144) resolution ratio and airborne image array "Map" are adopted for testing arrays. Length is 150frames, encoding frame rate is 30 frame per second, IPPP mode is adopted, motion estimate searching scope is $-16 \sim +16$, and a single ref-

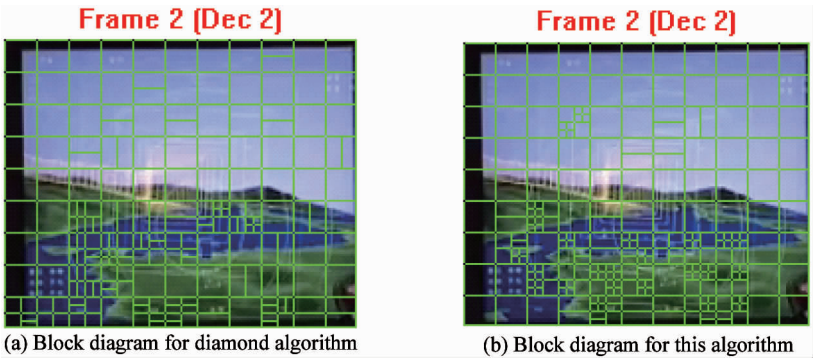
erence frame mode is adopted.

It can be seen from Table 1 that for "Foreman" with a great amount of random motion arrays, when QP = 28, the algorithm in this paper reduces PSNR by only 0.02 compared with full searching, while the code rate increases by 3.48%. For airborne image array "Map", which has lots of overlaid images such as lines and characters, the algorithm in this paper reduces PSNR by only 0.04 compared with full searching, while code rate increases by 9.2%. This algorithm has an even higher PSNR than the Diamond algorithm, and only brings a few code rate increase.

Both the diamond search algorithm and the "line scan and block merge" algorithm are applied to airborne image array "Map". Compressed code stream-oriented files are saved after processing. By decoding broadcasting of these code streams, block diagrams are gotten as shown in Fig. 4. It can be seen that the compression coding function in this manuscript is correct, a larger number of 4 × 4 small blocks are used by lines and figures in the image. It is more apt for hardware implementation with the premise of maintaining PSN.

Table 1 Experimental result comparison among full search, diamond search and proposed algorithm

Foreman QCIF	QP = 28		QP = 32	
	PSNR	Bit Rate(kbps)	PSNR	Bit Rate(kbps)
FS	35.80	148.38	32.97	85.36
DS	35.75	147.75	32.88	85.52
Proposed	35.78	153.54	32.95	97.22
Map QCIF	QP = 28		QP = 32	
	PSNR	Bit Rate(kbps)	PSNR	Bit Rate (kbps)
FS	36.26	272.49	33.37	152.12
DS	36.19	274.22	33.24	154.90
Proposed	36.22	298.44	33.31	174.96



(a) Block diagram for diamond algorithm (b) Block diagram for this algorithm

Fig. 4 Simulation results for inter mode selection

3 Hardware implementation of motion estimation algorithm

3.1 Hardware module division

Some newly modified algorithms appear on the basis of basic fast search algorithms, such as Unsymmetrical -Cross Multi-Hexagon-grid Search (UMHexagonS)[12] and Enhanced Predictive Zonal Search (EPZS)[13]. Lu, et al. analyses the detailed extent and structural directions for the blocks beforehand, which reduces the range of mode choices according to the analysis, thereby increasing the mode search speed[14]. Zou, et al. reduces Macro Block match time by fast-extracting human body areas in the foreground and background, as well as further removing MBs which affect overall motion estimation, according to reference MB motion directions[15]. Even though the algorithms above have a high coding efficiency, search paths are complicated, and search mode configurations varied. Unified parallel processing structures are hard to design for the corresponding computing processes. This is not suitable for the requirements of hardware design. Moreover, computing processes for threshold are over-complicated, which are not suitable for airborne real time process systems. Pure software algorithms haven't taken the FPGA hardware implementation's architecture characteristics into consideration. For example, during the search process, two continuous searching points are not adjacent in space, so while every search point is computed, all needed data have to be re-read from caches, which is a disadvantage of data reusing. Search computation is carried out respectively on different division mode configurations, so parallel advantages in hardware design cannot be used.

In the past most of the hardware implementations adopted a full search algorithm. Because of the limitation of Macro Blocks' computing speed, high definition video processing requirements are hard to be satisfied. Therefore, in order to satisfy an airborne image's real time compression and expansibility requirements, reference frames are reduced to one frame in this re-

search. The position corresponding to current MB is taken as the centre, and the search zone is fixed as the size of 3 × 3 MB, namely a 48 × 48 pixel search window. To reduce external memory bandwidth requirements, data reusing techniques are used as fully as possible. The motion estimation hardware circuit architecture designed in this research is shown in Fig. 5:

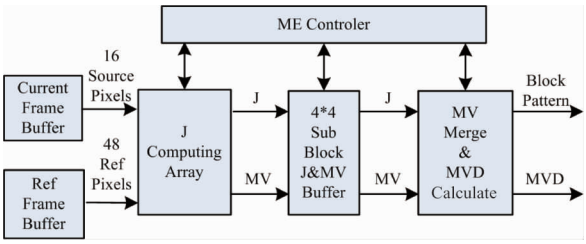


Fig. 5 Hardware architecture of motion estimation

Inside FPGA, rich RAM resources are used to constitute current frame caches and reference frames caches on chip, therefore the pressure of bandwidth for external frame memory chip is reduced. In every clock cycle, current frame caches can provide 16 original pixel data points, and reference frame caches can provide 48 original pixel data points. In the cost value computation arrays, there are 32 process elements (PE), while each PE unit accepts inputs of 16 original points and 16 reference points at clock rate. The best cost value J is gotten by comparing accounts. Under the coordinates of the motion estimation controller, "4 × 4 sub-blocks J and MV caches" are saved successively. When one MB computation is done, 16 best J are saved in caches along with the corresponding MVs. MV merge modules make different decisions on different divisions based on the block merge policy. The ultimate division mode configuration and corresponding motion vector difference (MVD) are exported. Next, MB's motion estimation computations will be carried on by repeating the above mentioned process, till all MB computations are done. The degree of distortion for bigger blocks can be obtained by the degree of distortion for merging smaller blocks. That is to say, 16 × 16 MBs only need to be divided into 16 4 × 4 sub-blocks, then calculate degree of distortion for each sub-blocks, and degree of distortion for any MB under 7 mode configurations can be obtained by overlaying results of sub-blocks. The designs in this manuscript adopt a bottom-up block merge policy, where results of small blocks SADs are fully reused. Repeated searching computations are left out in various division mode configurations, therefore computing speed is hugely improved.

3.2 FPGA simulation result

After the designs of RTL level for circuit architecture mentioned above are done, Active-HDL6.2 is used for simulation. Simulation waveforms are obtained as shown in Fig. 6, where Signal P _ data is for the current pixel point's data, and signal I _ data is for the reference pixel point's data. When zero clearing signal clr _ acc is effective, the position of 16 reference points changes successively. Each PE unit receives data for 16 original points and 16 reference points, and stream-oriented operations and computations are carried out. After 12 clks, signal din _ sad _ 0t turns into 26, namely cost value for the first 4 × 4 small block in the first line written in by this Macro Block. Same applies to signal din _ sad _ 1t, din _ sad _ 2t and din _ sad _ 3t, which are the cost values for the first 4 × 4 small block in the second, third, and third line respectively written in by this Macro Block. It can be seen that along with signal valid _ qj _ 16x16 output being in effect, signal qj _ 16x16 outputs 1184, namely the sum of 16 4 × 4 small blocks' cost value, 1184 = 4 × (26 + 58 + 90 + 122). Thus it can be seen that time sequence function for the motion estimation modules is correct, and meets design requirement. The motion estimation circuit architecture designed in this manuscript can be changed quickly according to different design resolution requirements. Consequently, real time compression and encoding for high definition images can be accomplished. The search scope is 32 × 32, and all division mode configurations are supported.

4 Conclusion

Based on the analysis on advantages and disadvantages of available algorithms, this manuscript proposes a fast motion estimation algorithm which is suitable for airborne video compressing techniques. Hardware architectures are given out, and software simulation and testing are carried out, as well as FPGA hardware implementations. Aiming at features for variable block size motion estimation technique adopted by H.264, the line scan fast search algorithm is designed using relativity of the search window data. A bottom-up block merge policy is introduced, and FPGA hardware parallel processing feature is fully used. The computing speed is hugely increased, and video quality and code rate are basically maintained unchanged. This is practically significant for system implementation of new types airborne real time image compressing modules.

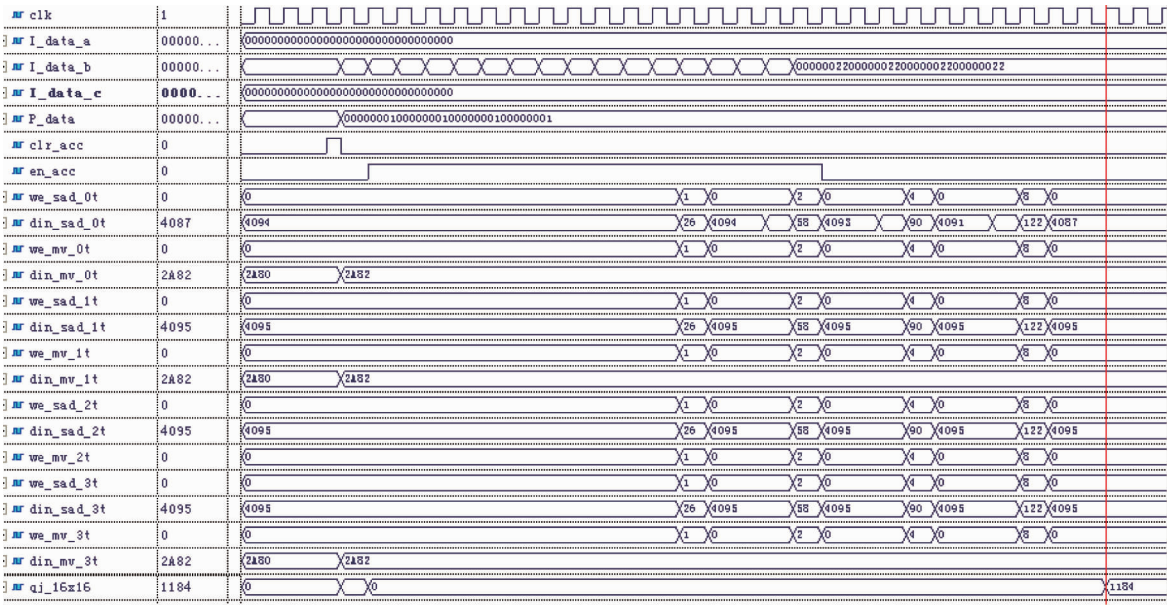


Fig. 6 Simulation waveform of PE computational array

References

[1] Chien S Y, Huang Y W, Chen C Y. Hardware architecture design of video compression for multimedia communication systems. *IEEE Communications Magazine*, 2005, 43 (8): 122-131

[2] Huang Y W, Wang T C, Hsieh B Y. Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H. 264. In: Proceedings of the IEEE International Symposium on Circuits and Systems, Bangkok, Thailand, 2003. 796-799

[3] Lu L, Mccanny J V, Sezer S. Reconfigurable system on a chip motion estimation architecture for multi-standard video coding. *IET Computers and Digital Techniques*, 2010, 4(5): 349-364

[4] Kao C Y, Wu C L, Lin Y L. A high-performance three-engine architecture for H. 264/AVC fractional motion estimation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2010, 18(4): 662-666

[5] Ndili O, Ogunfunmi T. Algorithm and architecture co-design of hardware-oriented, modified diamond search for fast motion estimation in H. 264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 2011, 21 (9): 1214-1227

[6] Chen T C, Chien S Y, Huang Y W. Analysis and architecture design of an HDTV720p 30 frames/s H. 264/AVC encoder. *IEEE Transactions on Circuits and Systems for Video Technology*, 2006, 16(6): 673-688

[7] Tham J Y, Ranganath S, Ranganath M. A novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 1998, 8(4): 369-377

[8] Ce Z, Xiao L, Chau L P. Hexagon-based search pattern for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2002, 12(5): 349-355

[9] Lin Z K, Lin H H, Chen Y H. On the performance improvement of H. 264 through foreground and background analyses. In: Proceedings of the IEEE International Conference on Multimedia and Expo, Amsterdam, Holland, 2005. 848-851

[10] Wu D, Wu S, Lim K P. Block inter mode decision for fast encoding of H. 264. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, Canada, 2004. 181-184

[11] Zhou Z, Sun M T, Hsu T F. Fast variable block-size motion estimation algorithm based on merge and split procedures for H. 264/MPEG-4 AVC. In: Proceedings of the International Symposium on Circuits and Systems, Vancouver, Canada, 2004, Vol3. 725-728

[12] Chen Z B, Xu J F, He Y. Fast integer-pel and fractional-pel motion estimation for H. 264/AVC. *Journal of Visual Communication and Image Representation*, 2006, 17(2): 264-290

[13] Alexis M, Tourapis. Enhanced predictive zonal search for single and multiple frame motion estimation. In: Proceedings of Visual Communications and Image Processing, 2002, Vol. 4671. 1069-1079

[14] Lu L, Zhou W. Efficient inter mode selection algorithm for H. 264. *Journal on Communications*, 2006, 27(7): 117-121

[15] Zou B J, Han L Q, Peng X N. A global motion estimation method for diving video sequences. *Acta Electronica Sinica*, 2008, 36 (12): 2412-2417

Wang Ke, born in 1979. He received his Ph. D degree from Tongji University in 2011. He is a senior engineer in China National Aeronautical Electronic Radio Research Institute now. His research interests include avionics, SoC technology and image processing.