May 2006, Vol.21, No.3, pp.370–377 J. Comput. Sci. & Technol.

An Efficient VLSI Architecture for Motion Compensation of AVS HDTV Decoder

Jun-Hao Zheng[1,3] (郑俊浩), Lei Deng[2] (邓 磊), Peng Zhang[1,3] (张 鹏), and Don Xie[4] (解晓东)

[1] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P.R. China*

[2] *Department of Computer Science, Harbin Institute of Technology, Harbin 150001, P.R. China*

[3] *Graduate University of Chinese Academy of Sciences, Beijing 100039, P.R. China*

[4] *Grandview Semiconductor (Beijing) Corporation, Beijing 100039, P.R. China*

E-mail: {jhzheng, ldeng, zhangpeng}@jdl.ac.cn; don.xie@grandviewsemi.com

Received October 31, 2005; revised March 13, 2006.

Abstract In the part 2 of advanced Audio Video coding Standard (AVS-P2), many efficient coding tools are adopted in motion compensation, such as new motion vector prediction, symmetric matching, quarter precision interpolation, etc. However, these new features enormously increase the computational complexity and the memory bandwidth requirement, which make motion compensation a difficult component in the implementation of the AVS HDTV decoder. This paper proposes an efficient motion compensation architecture for AVS-P2 video standard up to the Level 6.2 of the Jizhun Profile. It has a macroblock-level pipelined structure which consists of MV predictor unit, reference fetch unit and pixel interpolation unit. The proposed architecture exploits the parallelism in the AVS motion compensation algorithm to accelerate the speed of operations and uses the dedicated design to optimize the memory access. And it has been integrated in a prototype chip which is fabricated with TSMC 0.18-μm CMOS technology, and the experimental results show that this architecture can achieve the real time AVS-P2 decoding for the HDTV 1080i (1920×1088 4 : 2 : 0 60field/s) video. The efficient design can work at the frequency of 148.5MHz and the total gate count is about 225K.

Keywords motion compensation, AVS, VLSI architecture

1 Introduction

Chinese Audio Video Coding Standard [1] is a new national standard for the coding of video and audio which is known as AVS. The first version of AVS video standard[2] (AVS part 2, AVS-P2 in short) has been finished in Dec. 2003.

AVS-P2 defines a hybrid block-based video codec, similar to prior standards such as MPEG-2[3], MPEG-4[4], H.263[5] and H.264[6]. The hybrid coding framework combines inter-picture prediction with transform-based coding of the prediction errors. However, AVS-P2[7] is an application driven coding standard with well-optimized techniques. By adopting many new coding features and functionality, AVS-P2[8] achieves more than 50% coding gains over MPEG-2 and similar performance with lower cost compared with H.264.

The traditional block-based motion compensation (MC) which is used to exploit the temporal redundancy is improved in the AVS-P2 standard in order to achieve higher coding efficiency. These new features in the MC of AVS-P2 contains variable block sizes (16×16 down to 8×8), new motion vector (MV) prediction based on the vector triangle, multiple reference pictures (up to 2 frames or 4 fields), direct and symmetric prediction modes, unrestricted MV and quarter precision interpolation. All new features require higher calculation capacity and more memory bandwidth which directly affect the cost effectiveness of a commercial AVS-based video decoder solution. The amount of memory access[9] for MC is about 50% and the time consumed by pixel interpolation processing is about 25% in the AVS-P2 decoder. So MC becomes one of the most data intensive parts of the AVS-P2 decoder, and a bottleneck of implementation.

The application target of AVS-P2 is high definition digital broadcasting and high-density storage media. For HDTV 1080i (1920×1088 4 : 2 : 0 60field/s) video, the macroblock (MB) rate arrives to 244,800MB/s. If using the 150MHz system clock, the time budget is only 610 clock cycles assigned for the processing of one MB which contains four luminance blocks and two chrominance blocks. It is so tight that pure software implementation cannot provide real-time decoding if just depending on a simple or low-end CPU. So for the high-end application such as Set Top Box etc., it is necessary for the dedicated hardware accelerators. In [10–12] some kinds of dedicated MC architectures had been proposed which were based on prior specific video standards. However AVS-P2 is a new standard, its own features associated with the new requirements make the old designs unsuitable. For example, in order to find out more accurate predicted MV, AVS-P2 specifies the complicated algorithm using the spatial or temporal relative information, as further described in Subsection 2.1. The involved multiplier and division operations increase significantly the calculation complexity. While in other video standards such as MPEG-4 and H.264, a simple scheme is applied. The predicted MV is equal to the median value selected from three decoded MVs of the spatial neighborhood.

In this paper, we propose an efficient VLSI architec-

Regular Paper
Supported by the National High Technology Development 863 Program of China under Grant No.2003AA1Z1290.

ture for MC which contains a three-stage pipeline. The hardware architecture is composed of MV predictor unit, reference fetch unit and pixel interpolation unit. MV predictor unit employs the pipelined structure to exploit the parallelism for the AVS's special median prediction algorithm and uses the specific FIFO to smooth the memory accessing. An effective memory mapping scheme is applied to helping reference fetch unit achieve the high efficient memory access. Pixel interpolation unit adopts the flexible transfer array and the dedicated ALU to perform the heavy calculation task.

The remainder of the paper is organized as follows. The MC algorithm applied by AVS-P2 is described in Section 2. Section 3 describes the details of the implemented architecture of the MC subsystem. Simulation results and VLSI implementation will be shown in Section 4. Finally, we draw a conclusion in Section 5.

2 Motion Compensation Algorithm

The aim of MC is to exploit temporal redundancy to obtain the higher coding performance. AVS-P2 adopts the complicated but effective algorithms to improve the efficiency of MC. In this section all functional blocks of the MC algorithm will be explained.

2.1 Motion Vector Prediction

The prediction for an inter-coded MB is determined by the set of MVs that are associated with the MB. The MVs are generated through the vector difference (MVD) plus the predicted vector (MVP). Since significant gains in efficiency can be made by choosing a good prediction, the process of MV prediction becomes quite complicated in the AVS-P2 standard to find out more accurate MVP.

AVS-P2 uses either median or directional prediction to build the MVP, depending on the block partition mode and the availability of neighboring vectors. As described above, AVS-P2 employs a novel median selector. The edge with median length is selected from the vector triangle[2]. The scaled MVs make up of the triangle. For the scaling techniques, however, the complicated arithmetic calculations are involved which increase significantly the calculation complexity of the MV decoding process.

Fig.1 describes the specific triangle which consists of three edges *VAB*, *VBC* and *VAC*. The calculation process for MVP value of current block *E* is listed as follows.

Step 1. Calculate the scaled *MVA*, *MVB* and *MVC* according to the distance relationship between the neighboring and current block using (1):

$$MVX = \frac{\frac{512}{BlkDistX} \times mvX \times BlkDistE + 256}{2^9}, \qquad (1)$$

where X denotes block A or B or C and mv denotes the origin MVs for the neighboring block. Because multiple reference pictures can be supported by the AVS-P2 standard, different block has different reference picture. *BlkDist* is used to denote the distance differences between the two reference pictures of the neighboring blocks. The vector with double arrows is the scaled MV in Fig.1.

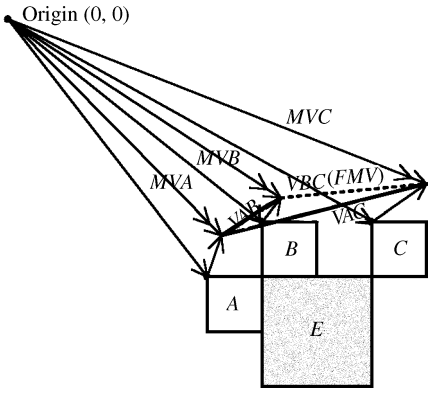


Fig.1. MV spatial prediction.

Step 2. Calculate the spatial distances between two scaled MVs.

$$VAB = \mathrm{Abs}(MVA\_x - MVB\_x) + \mathrm{Abs}(MVA\_y - MVB\_y),$$
$$VBC = \mathrm{Abs}(MVB\_x - MVC\_x) + \mathrm{Abs}(MVB\_y - MVC\_y),$$
$$VAC = \mathrm{Abs}(MVC\_x - MVA\_x) + \mathrm{Abs}(MVC\_y - MVA\_y).$$

Step 3. The temporary parameter *FMV* is given by the median of the corresponding spatial distances. The dashed line denotes the *FMV* in Fig.1.

$$FMV = \mathrm{Median}(VAB, VBC, VAC).$$

Step 4. Select the edge with median length and obtain the MVP using the scaled value from the corresponding vertex.

$$\begin{array}{lll} \text{if} & (FMV == VAB) & \mathrm{MVP} = MVC; \\ \text{else if} & (FMV == VBC) & \mathrm{MVP} = MVA; \\ \text{else} & & \mathrm{MVP} = MVB. \end{array}$$

Three vertexes (see Fig.1) need to be calculated so as to get only one MVP value which totally needs 3 divisions, 12 multiplications and 15 additions. Furthermore, AVS-P2 can support the 8×8 partition thus the maximum number of MVP value in one MB is 5 (three 8×8 blocks with unidirectional prediction and one 8×8 block with spatial direct prediction). The special method needs to be applied to accelerate the process which is described further in the Subsection 3.2.1.

2.2 MC Prediction Mode

AVS-P2 can support rich MB coding schemes with more than 30 kinds of MB types (mb_type) and tree structure MB partition. The predictive modes include intra, skip, forward, backward, spatial direct, temporal direct and symmetric. AVS-P2 adopts its own particular way to specify the symmetric and direct mode.

For the symmetric prediction, only forward MV is transmitted for each partition. The backward MV is conducted from the forward one by a symmetric rule as shown in Fig.2.

For traditional bi-prediction picture, the coding performance can be improved by joint estimation. But it is

very expensive to perform the joint estimation of forward and backward MVs in reality. However, for symmetric mode, joint estimation algorithm can be easily implemented with the same solution as conventional forward or backward searching. Besides, the symmetric mode can efficiently save bits of coding the backward MV.

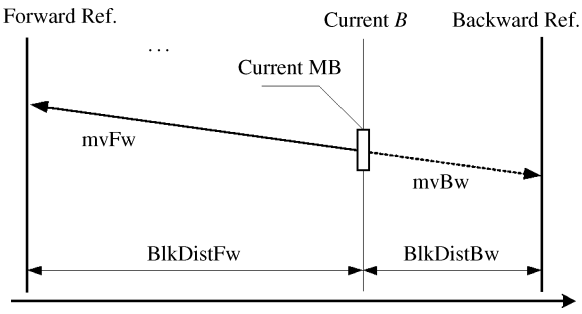


Fig.2. Symmetric mode.

For the direct prediction, both forward and backward MVs are derived from the MV of the collocated inter-coded block in the backward reference picture. Fig.3 illustrates the process of the temporal direct mode.

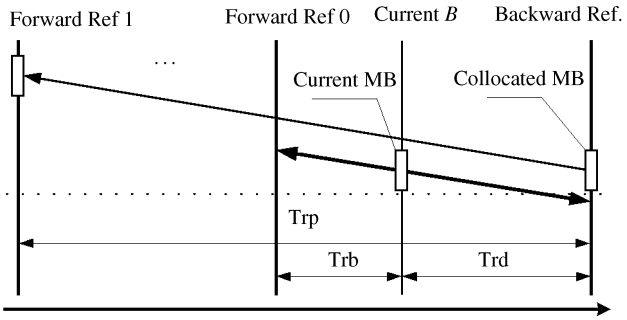


Fig.3. Temporal direct mode.

If the collocated block is intra-coded, AVS-P2 uses the spatial prediction algorithm to generate the forward and backward MVs respectively described in Subsection 2.1.

To support temporal direct MV prediction, all MVs in the latest P-picture need to be stored in memory as collocated MV buffer. However, for AVS-P2 1080i video the total bits of all motion data in one picture is about 118KB. It is so huge that all data must be stored to the external memory rather than the on-chip memory.

2.3 Pixel Interpolation

In the AVS-P2, MVs from the luminance component are specified with quarter-sample accuracy. Interpolation of the reference video pictures is necessary to generate the predicted MB using sub-sample precision MVs. The position of the pixel is illustrated in Fig.4, where the grey pixels are the integer pixels, light grey ones are 1/2-pixel and white ones for 1/4-pixel samples. 1/2-pixel interpolation filter is a 4-tap filter $(-1/8, 5/8, 5/8, -1/8)$. For ordinary 1/4-pixel samples, *a, c, d, f, i, k, n* and *q* in Fig.4, a 4-tap filter $(1/16, 7/16, 7/16, 1/16)$ is applied, and four special 1/4-pixel samples, *e, g, p* and *r*, are filtered by 2-tap bilinear filter (1/2, 1/2). A bilinear filter is used to interpolate the chrominance pictures.

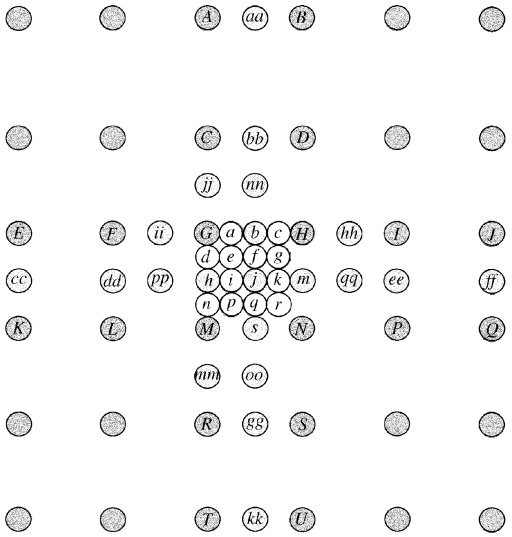


Fig.4. Position of integer pixels, 1/2 pixels and 1/4 pixels.

The complexity of the required interpolation filter varies depending on the phase specified by the MV. The calculation for quarter pixel sample requires the value of half pixel sample. The computation of $N \times N$ block needs a $(N+5) \times (N+5)$ reference block as illustrated in Fig.4.

For a detailed description of the MC algorithm, see the AVS-P2 standard[2].

3 Implemented Architecture

Fig.5 shows the system-level architecture of AVS decoder. The whole decoding system runs in macroblock level pipeline. The grey parts are responsible for motion compensation which consists of three stages.

MV predictor unit receives the motion data decoded by VLD and generates the real MVs. Reference fetch unit uses these MVs to fetch the integer pixel data from the decoded picture buffer. The fractional pixel data are calculated in the pixel interpolation unit and are trans-

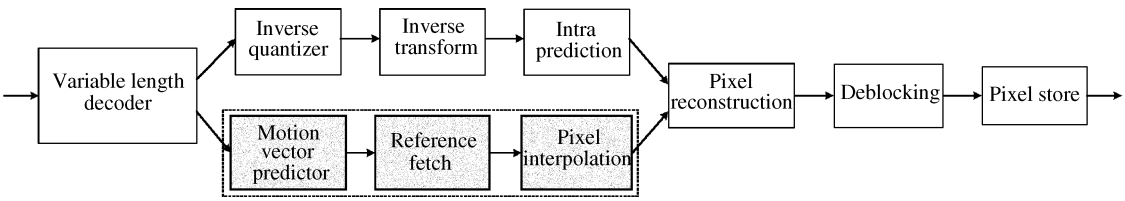


Fig.5. System-level architecture of AVS decoder.

ferred as the result of the inter prediction to the pixel recon struction unit. In this section, the MC architecture with efficient MB pipelining is proposed for HDTV specification (1920×1088 4 : 2 : 0 60field/s, the Jizhun profile at the Level 6.2). The proposed architecture can support all features for MC in the AVS-P2 standard. The detailed analysis and system/module designs are described in the following subsections.

3.1 MC Subsystem

From the algorithm analysis above, it is observed that the MV predictor, the reference fetch and the pixel interpolation are separable because there is no feedback loop among them. Therefore, a task-level interleaving scheme, i.e., macroblock pipelining, is incorporated into our design to accelerate the processing speed. So a three-stage MB pipelining is proposed as shown in Fig.6. All submodules have one or two input FIFOs to buffer MB data from the previous stage. Note that all logic blocks will be described in the later subsection.

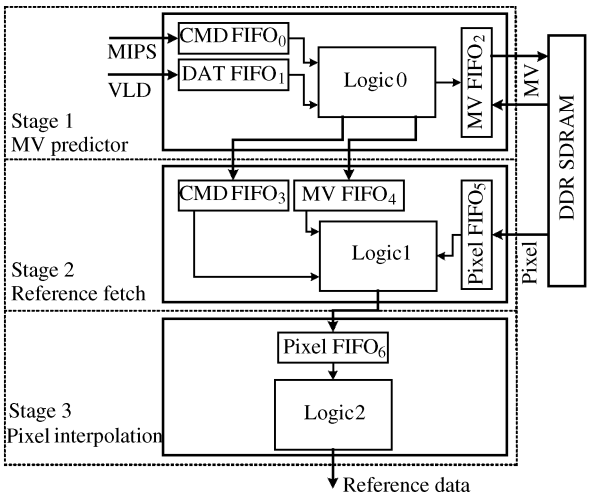


Fig.6. MC top level block diagram.

MV Predictor unit reads the side information from Command $FIFO_0$ which is written by MIPS and the MVD data from Data $FIFO_1$ written by VLD. The final MV and relative MB info are outputted to the MV $FIFO_4$ and Command $FIFO_3$ respectively. Reference Fetch unit reads the two FIFOs to generate the data address of DDR SDRAM. The integer pixel block samples from the DDR SDRAM are stored into the pixel $FIFO_5$ firstly. After the post process for combining and padding, the aligned data are sent into the pixel $FIFO_6$ in the pixel interpolation unit. Finally the interpolated block is outputted as the result of inter-prediction. The special MV $FIFO_2$ will be explained in Subsection 3.2.2.

There are two DDR interfaces: one for the reference motion information used by the direct mode in B-picture; the other for the integer pixel samples. In the architecture, 64-bit DDR SDRAM is adopted during the process of the design and implementation in order to guarantee the requirement of memory bandwidth.

3.2 MV Predictor Unit

MV predictor unit is the first stage for the whole MC subsystem which generates all motion data including MVs and reference picture indices. Fig.7 shows the implemented architecture of the MV predictor. The real lines indicate the data flow, and the dash lines for control messages.

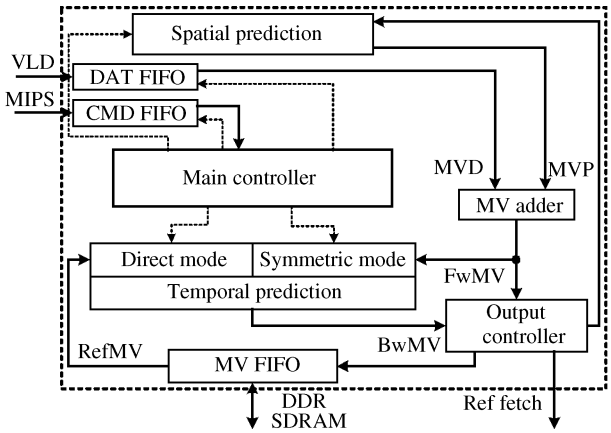


Fig.7. MV Predictor block diagram.

Main controller unit firstly parses the commands sent by MIPS which contain the MB information such as mb_type, the available flag for neighboring MBs etc. Then the controller invokes the corresponding submodule working according to the current MB mode. For example, if the controller finds the mb_type is equal to the symmetric mode, the symmetric prediction module will be activated through the handshake protocol.

Spatial and Temporal Predictions perform the spatial and temporal MV predictive operations respectively. The motion data from or to the external memory are stored to the MV FIFO firstly to avoid trivial memory accessing requests. The MV data read from the MV FIFO are used by the Direct Prediction as the reference motion data. Output controller manages the final motion data to output to the reference fetch module and maintains the neighboring information buffer used by spatial prediction. Besides, in order to support the variable prediction blocksize, the MV predictor unit transfers all block modes to the uniform 8×8 block which is the minimum blocksize to simplify the operations in the downstream stages.

Due to the limit space, only the spatial prediction and MV FIFO module are described in detail.

3.2.1 *Pipelined Spatial Prediction*

The algorithm is described in Subsection 2.1. The pipelined architecture for the spatial MV prediction is shown in Fig.8. It contains 5 stages for FMV calculation.

The 10b/9b division costs 2 cycles in our design. So it takes only 15 cycles to finish all operations for the cal-

culation of one MVP including preparing the input data. So for the worse case the total cycle is $15 \times 5 = 75$ cycles.

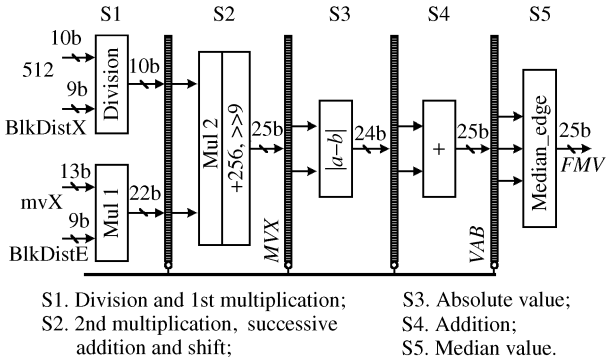


Fig.8. Pipelined spatial prediction.

Because the scaling technique is also applied to the direct and symmetric prediction in the AVS-P2 standard, the similar pipelined structures are implemented in the temporal prediction unit (see Fig.7).

3.2.2 MV FIFO

Direct mode needs to use the reference MVs from the backward reference picture. It is known that the motion data in a reference picture must be stored into the DDR SDRAM according to the previous analysis. The straightforward way is to access the memory once the decoder finds the direct prediction mode in the process of current MB decoding. However, it is awful to request the DDR controller frequently and irregularly. Because the DDR controller has to serve multiple clients and guarantee the schedulability of all critical tasks, i.e., the display feeder, it is probable that the request for the motion data in the current MB decoding period could not be acknowledged in time and the irregular request will also impact the services for other clients. So a dedicated FIFO is built to improve the efficiency of the memory access. In the P-picture decoding, the MV FIFO works as a cache. Motion data are written into MV FIFO after each MB is decoded. When MV FIFO is half full, the writing request is sent out to inform DDR controller and then the data are read from MV FIFO successively, at the same time, sending them to DDR SDRAM through the DDR interface. In the B-picture decoding, MV FIFO pre-fetches motion data from the DDR SDRAM. The data flows are shown in Fig.9.

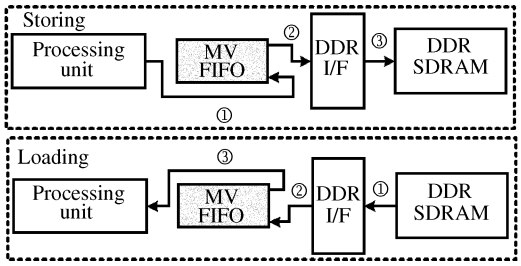


Fig.9. Data flows for MV FIFO.

3.3 Reference Fetch Unit

Reference fetch unit is the middle stage of the MC pipeline. It receives the MVs and control signals from the MV predictor, and generates and sends the address to DDR controller for fetching the reference pixels. The architecture is designed and illustrated in Fig.10, which consists of the main controller, data requesting, data padding, data combination and some necessary FIFOs.

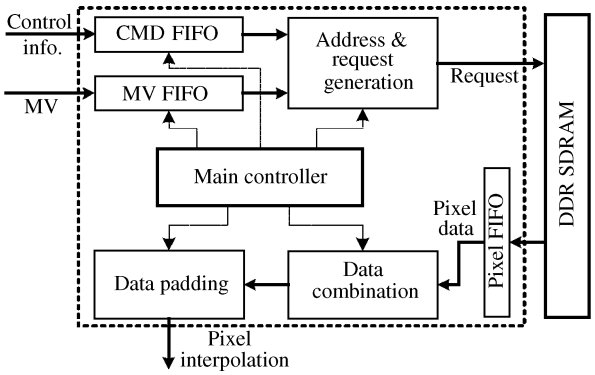


Fig.10. Reference fetch block diagram.

In fact, the memory mapping of reconstructed picture is the key issue which determines whether the decoder can operate in real-time or not. In order to meet the memory bandwidth requirement, an efficient mapping method is applied as described below. Besides, in the AVS-P2 standard, MVs can point to any position of the picture and even to outside of the picture. Thus two additional functions, named data combination and data padding units (see Fig.10) are employed to deal with these exceptional cases.

3.3.1 Memory Mapping

The decoder system adopts the 64-bit DDR SDRAM which can be considered as the logic memory with 128-bit width, that is, it can read and write 16 pixels at one cycle.

MC consumes the most part of the memory bandwidth. The rate is more than 50% and even more in the worse case[9]. The memory addresses of the reference blocks are unpredictable. A typical reference line with 21 pixels ($N = 16$) may require three 128-bit words. It is probable that the three words are located in the different banks, thus more cycles will be cost to activate the additional bank.

So our main idea is to optimize the MC reference fetch accessing and reduce the SDRAM operational overhead, i.e., bank conflict. In our design, each 8 consecutive MBs are put into one row of one bank to minimize the number of bank conflict. Pixels in a line of the picture are located at the successive address in the DDR SDRAM. Picture data are organized and aligned well in the DDR SDRAM by memory pages. The scheme reduces the probability of changing the row address which can save many cycles so as to fulfill the real-time MC requests.

3.3.2 Data Combination

The data combination unit is responsible for aligning fetched pixel data and combining them to form the required reference line. Because the required reference block from the DDR SDRAM is usually not aligned in the boundary of the pixel array. An example is shown in Fig.11, in which a 21-pixel reference line is required. Firstly three successive memory words are fetched from the DDR SDRAM, and then they are combined to form the required 21-pixel reference line.

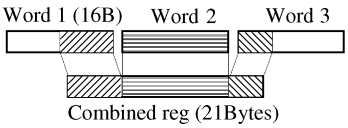


Fig.11. Data combination.

3.3.3 Data Padding

The data padding unit is used to obtain the pixels which are outside of the picture in order to support the unrestricted MV. According to the MV value, there exist five cases in the horizontal locations for the reference block as shown in Fig.12.

Cases 1 and 2 need leftward boundary padding, while cases 4 and 5 need rightward boundary padding. Case 3 does not need any padding. The vertical padding also has five cases, and it is merged into the downstream module, i.e., pixel interpolation. Since the reference block may be outside of the reference picture, the combined reference line in Subsection 3.3.2 may be not complete. Its outside part is derived by horizontally padding the pixels on the boundary of the picture.

Case 2 is illustrated as an example in Fig.12 where the left part is out of the picture. Thus the left first pixel of the combined reference line is duplicated to the outside part of the required reference line.

All combined and padded reference lines are finally written into the pixel FIFO of the pixel interpolation module.

3.4 Pixel Interpolation Unit

Pixel interpolation unit takes up the major computational task of fractional sample interpolation. The interpolated results are the final results of the MC process. The architecture is given in Fig.13. In order to meet the computational requirement of the AVS-P2 HDTV decoder, this architecture is constructed by two interpolation pipelines. Each pipeline has its own pixel register array for data transferring and ALU for calculating the fraction samples.

3.4.1 Data Feeder

The data feeder is responsible for feeding data into the interpolation pipeline. The interpolation pipeline is based on the minimal block size, i.e., 8×8 block size. Thus the data feeder also needs to split other block into 8×8 blocks before transferring them into the pipeline. An example for 8×16 block interpolation is illustrated in Fig.14. The data feeder will send the first 13 rows from the 0th to the 12th position into the data transfer. Then the read pointer should be rolled backward to the 8th position. The second reference block from the 8th to the 20th position is read successively for the lower 8×8 block.

By adjusting the read pointer of the input FIFO, the data feeder unit also implements the vertical padding operation, which just repeats reading the data from the same position in the input FIFO.

3.4.2 Data Transfer Array

The pixel data transfer array has a register array which has 13-row and 6-column registers used to reserve the data for the current or later processing. The active pixel register serves the current processing. And the passive pixel register stores the data for the later processing. Active rectangle signed by the shadow area in Fig.15 has 6×6 active pixel registers used to hold the MB samples for 1/2 or 1/4 interpolation calculation. The data transfer in three ways: upwards, downwards and to the left to obtain the high transfer efficiency.

In addition, a dedicated ALU are adopted to achieve the high computational capability. The details of data transfer array and ALU can be found in our previous work[9].

4 Implementation Results

We have described the design in Verilog HDL at the RTL level. According to the AVS-P2 verification model[13], a C-code model of the MC subsystem is also developed to generate simulation vectors. By testing with 12 HD (including 720p and 1080i) bitstreams, Synopsys VCS simulation results show that our Verilog code is functionally identical with the MC functional model specified by the AVS-P2 standard.

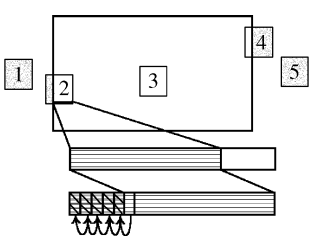


Fig.12. Data padding.

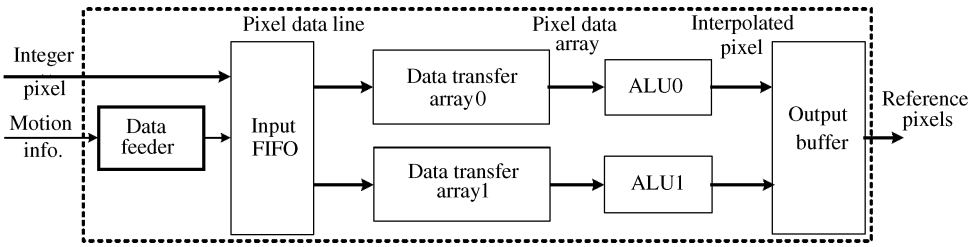


Fig.13. Pixel interpolation block diagram.

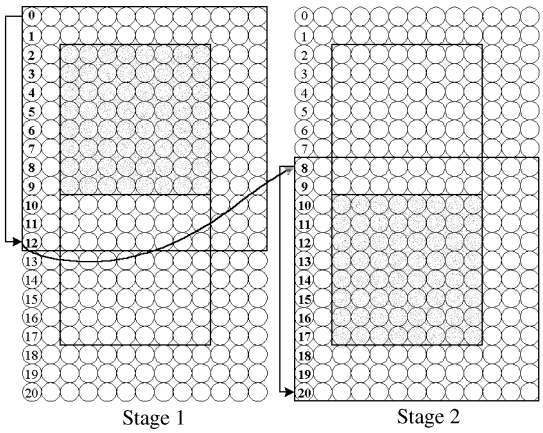


Fig.14. Pointer rollback for data feeder.

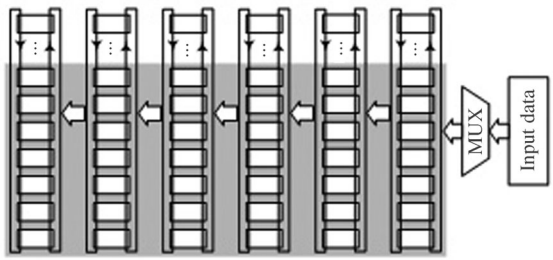


Fig.15. Data transfer array.

The validated Verilog code is synthesized using TSMC 0.18-μm complementary metal oxide semiconductor (CMOS) cells library by the Synopsys Design Compiler. The circuit totally costs about 225K logic gates exclusive the SRAM when the working frequency is set to 148.5MHz. The implemented architecture costs at most 580 cycles to perform the MC operations for each MB, which is sufficient to realize the real-time MC process for HDTV AVS-P2 bitstreams.

The gate count of each functional block is listed in Table 1. Several SRAM modules worked as data FIFOs are used in the MC subsystem as shown in Fig.6. The total area for all SRAM is about 0.7mm^2.

Table 1. Gate Count Profile

Functional block	Gate count
MV predictor	71K
Reference fetch	54K
Pixel interpolation	100K
Total	225K

The proposed VLSI architecture for MC has been integrated into our AVS-P2 decoder SoC. A prototype chip named AVS101 was fabricated in Mar. 2005. The AVS101 is an AVS-P2 decoder chip capable of full HD real-time decoding. The core size is about 6.9×6.9mm^2. The chip photo is shown in Fig.16. AVS101 can fully support for AVS-P2 stream up to the Jizhun profile at the level 6.2 which is the highest level in the AVS-P2 standard.

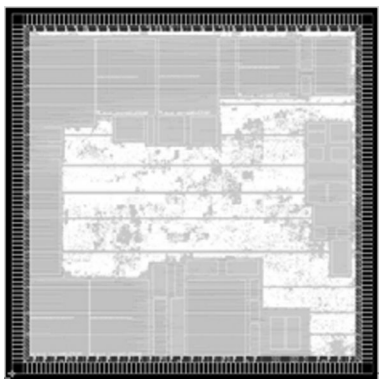


Fig.16. AVS101 chip.

5 Conclusion

In this paper, we contribute an efficient VLSI architecture for MC of the AVS-P2 standard. Firstly we analyze the algorithm of the AVS-P2 MC to obtain the proper parallelism information based on the new features.

Secondly, a macroblock-level pipeline is proposed. Table 2 provides a list to describe our proposed solutions which can meet the new requirements of the AVS-P2 standard. Our main idea is to use the three-stage operations to simplify the hardware design and the pipelined structure to improve the processing performance. The proposed VLSI architecture for MC contains a three-stage pipeline which consists of MV predictor unit, reference fetch unit, and pixel interpolation unit. MV predictor unit employs the pipelined structure to accelerate the process for the new median prediction algorithm and uses the dedicated MV FIFO to smooth the memory accessing. Reference fetch unit implements the parallel between the memory request and the data post processing to achieve the high efficient memory access through the effective memory mapping scheme. Pixel interpolation unit adopts the flexible data feeder unit to support variable block partition. The efficient transfer array and the dedicated ALU unit are integrated into the pixel interpolation unit to take up the heavy calculation task. In addition, because the major framework of motion compensation between H.264 and AVS-P2 are similar, the efficient schemes provided by the proposed architectures can also work for H.264 with the minor modifications to deal with the differences between the two algorithms in details.

Table 2. New Features vs. Proposed Solutions

New features	Proposed solutions	Owner unit
Variable block sizes	Transfer to the uniform block size	MV predictor
New complicated MV prediction	Pipelined structure	MV predictor
Multiple reference pictures	Specific address generator	Reference fetch
Direct and symmetric prediction mode	Specific accelerator for MV calculation	MV predictor
Unrestricted MV	Specific schemes for all situations	Reference fetch
Quarter precision interpolation	Efficient data transfer array	Pixel interpolation

Finally, we give our implementation results. A prototype chip integrated the proposed MC architecture was fabricated with TSMC 0.18-μm CMOS process and is capable of decoding 1920×1088 AVS-P2 interlace video ($4:2:0$ 60 field/s) in real time at the working frequency of 148.5MHz. The gate count for MC subsystem is 225K.

References

[1] AVS working group official website. http://www.avs.org.cn.

[2] Information technology—Advanced coding of audio and video—Part 2: Video. AVS-P2 Standard draft, Mar. 2005.

[3] Information technology—General coding of moving picture and associated audio information: Video. ITU Recommendation H.262 | ISO/IEC 13818-2 (MPEG-2) Standard draft, Mar. 1994.

[4] Information technology—Coding of audio-visual objects—Part 2: Visual. ISO/IEC 14496-2 (MPEG-4) Standard, Jul. 2001.

[5] Video coding for low bitrate communication. ITU-T Recommendation H.263 Standard, Nov. 1995.

[6] Advanced video coding for generic audiovisual services. ITU-T Recommendation H.264 | ISO/IEC 14496-10 AVC Standard draft, Mar. 2005.

[7] Liang Fan, Siwei Ma, Feng Wu. Overview of AVS video standard. In *Proc. IEEE Int. Conf. Multimedia and Expo (ICME2004),* Taipei, Jun. 2004, pp.423–426.

[8] Lu Yu, Feng Yi, Jie Dong, Cixun Zhang. Overview of AVS-Video: Tools, performance and complexity. In *Proc. SPIE, Visual Communications and Image Processing,* Beijing, China, Jul. 2005, pp.679–690.

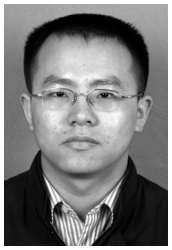
[9] Lei Deng, Wen Gao, Ming-Zeng Hu, Zhen-Zhou Ji. An efficient VLSI implementation for MC interpolation of AVS standard. In *Advances in Multimedia Information Processing—PCM 2004: 5th Pacific Rim Conference on Multimedia,* Tokyo, Japan, Dec. 2004, pp.200–206.

[10] He Wei-Feng, Mao Zhi-Gang, Wang Jin-Xiang, Wang Dao-Fu. Design and implementation of motion compensation for MPEG-4 AS profile streaming video decoding. In *Proc. 5th International Conference on ASIC,* Beijing, China, Oct. 2003, pp.942–945.

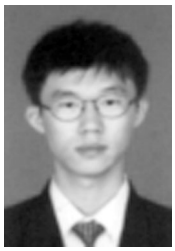
[11] Lee J, Vijaykrishnan N, Irwin M J. High performance array processor for video decoding. In *Proc. IEEE Computer Society Annual Symposium on VLSI,* Florida, USA, May 2005, pp.28–33.

[12] Chih-Da Chien, Ho-Chun Chen *et al.* A low-power motion compensation IP core design for MPEG-1/2/4 video decoding. In *IEEE Int. Symp. Circuits and Systems (ISCAS2005),* Kobe, Japan, May 2005, pp.4542–4545.

[13] AVS1.0 part 2 reference software model. RM52r1, Dec. 2004.



Jun-Hao Zheng received the B.S. degree (June 2000) and M.S. degree (June 2003) both from Huazhong University of Science and Technology, Wuhan, China. Now he is working toward the Ph.D. degree in the Dept. Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences. His major research interests include video coding technology and associated VLSI architectures.



Lei Deng received his B.Sc. degree in computer science, in 1998 from Jilin University and M.Sc. degree in computer science and engineering, in 2000 from Harbin Institute of technology. He is pursuing his Ph.D. degree in the Harbin Institute of Technology for computer architecture and video signal processing. His research interests lie in the areas of computer architecture, digital signal processing and video compression.

Peng Zhang received the B.S. degree in electronic engineering and information science from University of Science and Technology of China, in 2002, and the M.S. degree in computer science from Institute of Computing Technology, Chinese Academy of Sciences, in 2004. At present, he is a Ph.D. candidate in Institute of Computing Technology, Chinese Academy of Sciences. His research interests include video coding, computer architecture, effective VLSI implementation and SoC design.

Don Xie received the M.S. and the Ph.D. degrees in electrical engineering from University of Rochester, New York, USA in 1992 and 1994, respectively. Now he is the engineering director of the Grandview Semiconductor, Beijing, China. His research interests include the SoC design and embedded system for consumer electronics.