A decoder architecture for advanced video coding standard*

Dianfu Li, Lu Yu, Jie Dong
Department of Information Science and Electronic Engineering,
Zhejiang University, China, 310027

ABSTRACT

In this paper, we describe a VLSI architecture of video decoder for AVS (Audio Video Coding Standard). The system architecture, as well as the design of major function-specific processing units (Variable Length Decoder, Deblockding Filter), is discussed. Analyzing the architecture of decoder system and the feature of each processing unit, we develop a system controller combined the centralized and decentralized control scheme, which provides high efficient communication between the processing units and minimizes the size of interconnected buffers. A bus-arbitration algorithm named Token Ring algorithm is designed to control the allocation of the SDRAM bus. This algorithm can avoid the conflicts on bus and reduce the internal buffer size, and its control logic is simple. Our simulation shows that this architecture can meet the requirement of AVS Jizhun Profile@4.0 level real time decoding, without a high cost in hardware and clock rate. Moreover, some design idea in the AVS decoder can be expanded to H.264 because of the similarity between the two video coding standards.
Keywords: AVS, H.264, Centralized Control scheme, Decentralized Control scheme, Token Ring algorithm

1. INTRODUCTION

H.264/AVC is the newest video coding standard of Joint Video Team (JVT) made by experts from ISO/IEC MPEG-4 Advanced Video Coding (AVC) and ITU-T H.264[1]. The new standard significantly outperforms previous ones in compression efficiency. AVS[2] is digital audio and video coding standard developed by China. The code efficiency of AVS video standard is 2-3 times of that of MPEG-2 (base on the different resolution of video picture), which has similar performance to MPEG-4 AVC/H.264 at high definition video, but with much less implementation complexity than it. The main differences between these two video coding standards are shown in Table.1 as follows.

Table.1. Main differences between H.264 and AVS

Video technology	**H.264**	**AVS**
Transform	Based on 4×4 block size	Based on 8×8 block size
Intra prediction	For luma block: 9 modes for 4×4 block, 4 modes for 16×16 block	For luma block: 5 modes for 8×8block
Inter prediction	Smallest motion compensation block: 4×4	Smallest motion compensation block: 8×8
Interpolation	6-tap filter for half samples 2-tap filter for quarter samples	4-tap filter for half and quarter samples
Entropy coding	CAVLC, CABAC	CAVLC

As shown in Tab.1, many differences between H.264 and AVS result from the size of smallest block, 4×4 for H.264 and 8×8 for AVS. But the basic technology framework is alike for the two coding standards. For examples, they both adopt the integer transform, the deblocking filter, the same idea of intra prediction and so on. Based on these similarities, their hardware implementation can be almost the same especially on the basic architecture of system and system control scheme.

In this paper, we develop a system architecture for AVS video decoder. In video decoder design, it is important to meet the requirement of real-time and cost-effective implementation. In order to achieve these aims, some efficient methods

*This research is supported by NSFC under contract No. 90207005.

Visual Communications and Image Processing 2005, edited by Shipeng Li,
Fernando Pereira, Heung-Yeung Shum, Andrew G. Tescher, Proc. of SPIE Vol. 5960
(SPIE, Bellingham, WA, 2005) · 0277-786X/05/$15 · doi: 10.1117/12.633201

are developed in this design. Working at low clock rate, Variable Length Decoder can realize the real-time decoding using parallel decode algorithm based on PLA mainly. For code words with large value, VLD adopts serial decode algorithm to reduce the hardware cost. An efficient architecture of Deblocking filter is designed to highly reduce the total cycles of filtering process and the hardware cost. A system controller with combined centralized and decentralized control scheme is developed to provide high efficient communication between the processing units. A bus-arbitration algorithm named Token Ring algorithm is adopted to avoid the conflicts on the bus. The rest of the paper is organized as follows. Section 2 begins with a brief introduction to the overall architecture of AVS decoder, followed by a detailed design of key processing units. The system control scheme with combined centralized and decentralized control scheme is described in section 3. In section 4, the bus-arbitration algorithm is introduced. And in section 5, results of system simulation are shown. We draw the conclusion in the last section.

2. DECODER ARCHITECTURE

2.1 Overview of the decoder architecture

The AVS decoder architecture is shown in Fig.1. The system consists of a System Controller and processing units. Processing units include VLD (Variable Length Decoder), Baseline unit, MC (Motion Compensation) unit, INTRA (intra prediction unit), DF (Deblocking Filter), WB (Write Back unit) and associated embedded buffers. Baseline unit consists of IZZ (Inverse Zigzag Scanner) and IQ/IICT (Inverse Quantizer and Inverse Discrete Cosine Transformer). And MC unit consists of AGU (Address Generate Unit), RRD (Read Reference Data) and ITP (Interpolation). The BSF (Bitsteam FIFO) receives the input bitstream data, which is then transferred to the off-chip Synchronous DRAM (SDRAM). The compressed data is fetched out again to VLD, where the bitstream is parsed into two parts. One part, the control information, is sent to System Controller. IZZ fetches the other part for inverse scanning, and sends the results to IQ/IICT for inverse quantizing and two dimension transforming. For those MBs (Macro Blocks) which are coded with motion compensation technique, MC unit is necessary. According to the coding type of the current MB, either INTRA or MC unit is performed to produce the intra or inter compensated data. The Adder is used to complete the intra/inter compensation. Filtered by the Deblocking Filter, the reconstructed MB is written back to SDRAM for display and as reference of future frames. System Controller enables the operation of each processing unit and provides efficient communication between the processing units.

In order to save the on-chip memory, all the data of bitstream, reference frame, decoded frame and MV (Motion Vector) is stored in the off-chip memory (SDRAM). SDRAM Arbiter/Controller receives and schedules all the requests from the processing units and controls the operation of SDRAM.

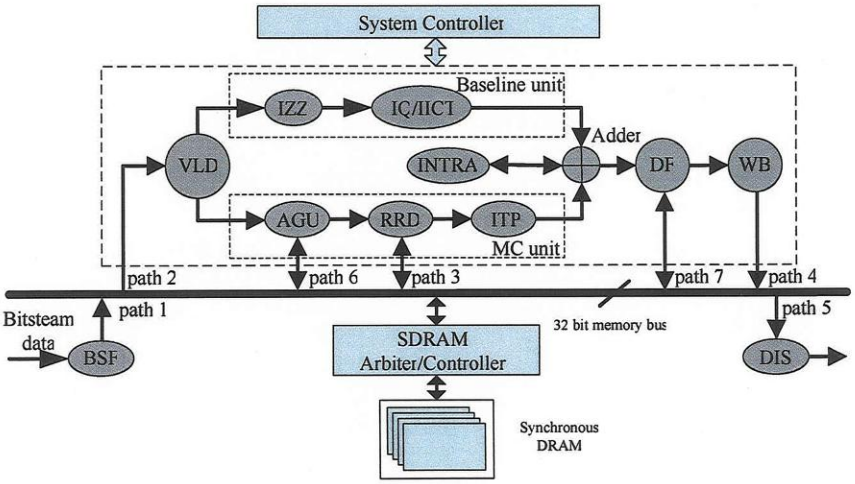

Fig.1. Architecture of AVS decoder

At the encoder side of AVS, the transform coefficients are encoded with scanning a block from the down-right to the up-left. At the decoder side, the position of transform coefficients by inverse scanning is unknown until the EOB (End of Block) flag is decoded out. A SRAM (64×9bit) is used to store the output data from IZZ. In baseline unit, IICT fetches two data each clock cycle and IQ outputs only one data at the same time. In this design, the clock frequency for IICT is two times of that of IQ, which can make the two processing units work synchronously. With this method, the buffer for storing the output data of IQ is unnecessary, so on-chip memory is saved. The MC unit is divided into three parts: AGU, RRD and ITP. The reference block address in SDRAM is generated by AGU, according which RRD gets the reference block data from SDRAM and sends them to ITP for interpolation. The division makes the motion compensation effective. The detail structure of key units, VLD and Deblocking Filter, is described as follows.

2.2 Variable Length Decoder

The main task of VLD is parsing the bitstream data to find the start code, insert-code and decode variable length coded or fixed length coded syntax elements. In AVS video standard, the code words for variable length coded syntax elements, such as MB type, MV (Motion Vector) and residual coefficients etc are structured based on Exponential-Golomb codes. All the syntax elements are mapping to CODENUM, and then mapping to a unique binary codeword in bitstream based on Exponential-Golomb codes. There are two kinds of implementation for Variable Length Decoder. One is basically a tree-searching algorithm and the other is a parallel structure implemented through the lookup table. The former is bit-serially operated, and its decoding speed is low but it can save the hardware cost greatly. The latter can provide a high decoding speed, but needs more hardware to realize the lookup table. Owning to the feature of the two algorithms, we adopt both of the decode algorithm. The parallel decode algorithm is designed based on PLA (Programmable Logic Array). PLA, a special structure of ROM, can store much information with few storage units, which is composed of some "And" arrays and "Or" arrays. Fig.2 shows the block diagram of the VLD for AVS video bitstream.

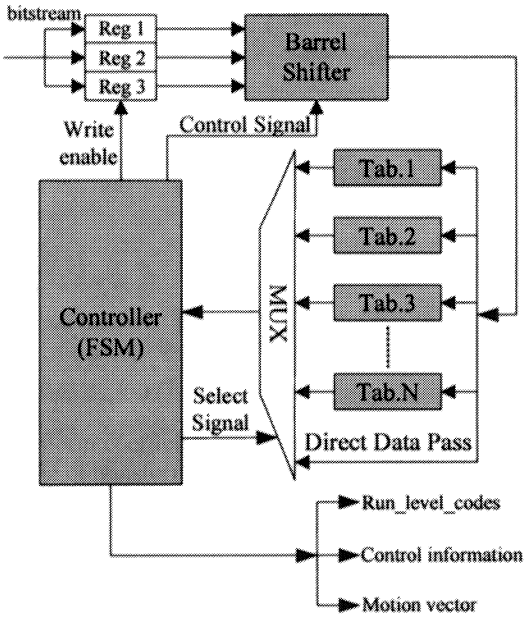

Fig.2. Architecture of Variable Length Decoder

The Tables in Fig.2, implemented by PLA, contain the CODENUM and code word length information. The Controller monitors the variable-length decoding process and also indicates the current processing position in a bitstream. The registers, which are 32bits wide each, contain the current bitstream to be processed. The Barrel Shifter operates like a sliding window on the contents of these registers. During the decoding process, the output of the Barrel Shifter is matched with all entries in the PLAs. If the data is suitable for the PLA decoding, a matched table is selected by the

Controller, otherwise, the Direct Data Pass is chosen. The corresponding CODENUM and the length of the code word are generated. After the code word length is fed back to the Barrel Shifter, the shifter is shifted to the beginning of the next code word according to the accumulated code length. The control information is sent to System Controller, while the motion vectors are sent to AGU for SDRAM address generation. The decoded run-level-codes are sent to the baseline unit for further processing. For the Barrel Shifter, the sliding window size (the length of output from the shifter) is 32 bits, the same length as the start code, which has the maximum code length among the code word in AVS bitstream. So the shifter maybe fetches the data from two registers at the same time. Three registers are used to store the bitstream, one is for written and the others for read. Our simulation shows that this method can guarantee correct output from the shifter, and avoid data operation conflicts in the registers.

The parallel decode algorithm based on PLA is not available for the code word with long length. Simulation result indicates that it is difficult for the synthesis tool to synthesize a PLA if the value of code word is larger than 2^8. In this case, the serial decode algorithm is adopted to solve the problem. According to the feature of Exponential-Golomb codes, when decoding a code word with serial algorithm, the VLD Controller fetches the bitstream data bit-by-bit through the Direct Data Pass until the current input bit is 1. Then the Controller parses out the code word by calculating the recorded count of 0 and feeds back the length of the code word to the Barrel Shifter. In AVS video standard, the value of some code words is larger than 2^8, such as mb_skip_run and MV. These code words are parsed with this algorithm.

Our simulation shows that the VLD synthesized gate count is only 13.6K (2K for PLA table) by the Synopsys Design Analyzer with critical path constrain set to 10ns (100MHz).

2.3 Deblocking Filter

Deblocking loop filter is adopted in both H.264 and AVS to eliminate blocking artifacts and to achieve much better both subjective views and objective quality. Contributing to a considerable amount of computation, the deblocking filter becomes the bottleneck in a video decoder. The first reason is conditional operations in the inner loop of the algorithm. In implementation view, conditional operations[8], such as clipping, threshold-decide, and jumping, will break or affect the pipeline operations, so the execution speed of deblocking filter function is very slow. The second reason is irregular data access caused by adaptive deblocking. Based on the above two reasons, an efficient deblocking filter architecture is designed. Data access is carefully organized and some other measures are adopted, all those facilitate pipelining in this architecture so as to increase the efficiency of getting data from the SRAM and highly reduce the total cycles of filtering process. Architecture of Deblocking Filter is shown in Fig.3.

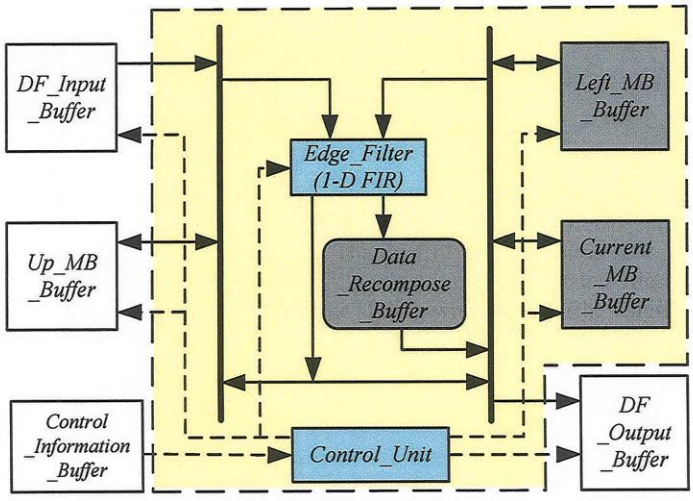

Fig.3. Architecture of Deblocking Filter2.3.1 Interface and Basic function unit

As shown in the Fig.3, there are four interfaces between Deblocking Filter and other processing units: the first one is *DF_Input_Buffer*, from which Deblocking Filter gets the current MB data; the second one is *Up_MB_Buffer*, which is used to store downmost four line pixels and control information of the up MB line; the third one is the interface of between Deblocking Filter and System Controller; and the last one is *DF_Output_Buffer* for storage of the results of Deblocking Filter. In the Deblocking Filter, there are 5 basic functional units. *Control_Unit* calculates filter parameters based on received control information and controls data input and output of *Edge_Filter*. *Edge_Filter* (1-D FIR), a parallel-in parallel-out filter, accomplishes different filtering processes based on different filter parameters. *Left_MB_Buffer* stores the rightmost four column samples of left MB. *Current_MB_Buffer* is used to store the intermediate data. The intermediate data is the result of horizontal filtering, which still needs vertical filtering. *Data_Recompose_Buffer* stores the data which needs to be recomposed.

2.3.2 Architecture analysis

After intra/inter compensation, each pixel is saturated to 0~255 and can be stored 8 bits. Due to the size of SDRAM bus is 32 bits, we group four pixels in one line of block into one 32 bits unit. This unit is called LOP (Line-of-Pixel). There two types of LOP: H_LOP (Horizontal LOP) which contains four pixels in the horizontal direction, and V_LOP (Vertical LOP) which contains four pixels in the vertical direction. During the filtering process, the input for *Edge_Filter* is two 32bit LOPs (8 pixels) and the output is the same. The H_LOP is used in the horizontal filtering and V_LOP used in the vertical filtering.

When filtering vertical edges, the input data of the *Edge_Filter* is two H_LOPs, one of which comes from the *DF_Input_Buffer*, and the other from the *Left_MB_Buffer*. After the horizontal filtering the current block, *Left_MB_Buffer* is updated by the data of current block. When filtering the horizontal edges, the input data of the *Edge_Filter* is two V_LOPs, one of which is from the *Up_MB_Buffer* and the other from the *Current_MB_Buffer*. After the filter process of current MB, the up MB data in the SDRAM will be changed by downmost four line pixels of current MB, which is used to filtering the MB in next MB line.

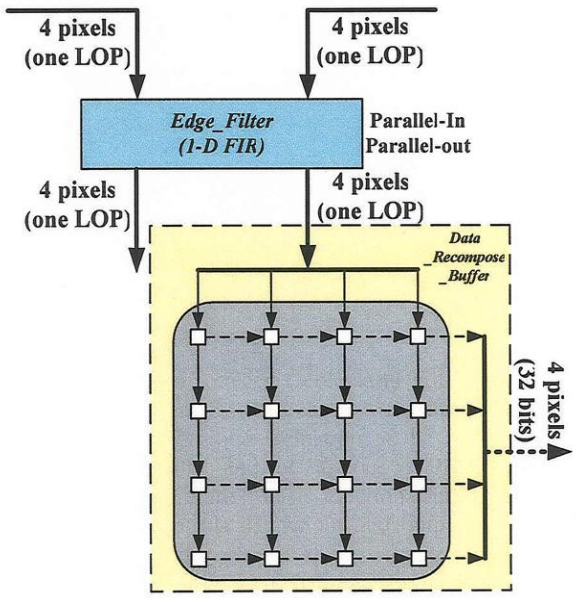

Fig.4. Diagram for Data Recompose

According to the deblocking loop filter of AVS video standard, vertical edges are filtered first, from left to right, and then horizontal edges are filtered from top to bottom for each MB, so some pixels after horizontal filtering is not final output data. During the horizontal filtering, the outputs of the *Edge_Filter* are H_LOPs. But for the vertical filtering, V_LOPs are needed. In order to facilitate to get data during the vertical filtering of horizontal edge, the output data

(H_LOPs) after horizontal filtering must be recomposed to V_LOPs and then stored in the *Current_MB_Buffer*. In this design, several registers organized carefully to accomplish the data reconstruction. As shown in Fig.4, every small square represents an 8-bit register, solid arrows denote input data path while dotted arrows denote output data path. Using this architecture the data is recomposed simply. And it facilitates to get data during vertical filtering, and support both horizontal filtering and vertical filtering on the same circuit. On the other hand, data after the vertical filtering come back to the horizontal storage format and then sent to *DF_Output_Buffer*.

Using proper storage organization and pipelining, the Deblocking Filter can increase the efficiency of getting data, and support both horizontal filtering and vertical filtering on the same circuit (a parallel-in parallel-out FIR filter), so the hardware cost is reduced. Our simulation shows that the synthesized gate count is only 12.2K by the Synopsys Design Analyzer with critical path constrain set to 10ns (100MHz).

3. CONTROL SCHEME

In video decoder architecture, processing units are connected together by data path and control path. The data path transfers the decoded data and the control path transfers the control information such as MB type, picture structure and so on, which are extracted by VLD. So System Controller in the decoder system is responsible for synchronization of the two paths and communication between different processing units. System Controller should sets up the right control information for each processing unit and enables the processing unit at the time when the control information in the control path is ready. Generally, there are two kinds of control schemes, centralized and decentralized control schemes. In this architecture, a controller combined the two control schemes is developed. Furthermore, different from some traditional control path[3, 4, 6], we use registers, instead of FIFO, to transfer the control information, which can make the hardware cost cut down.

3.1 Centralized Control scheme

Centralized Control scheme is a pipeline architecture on a block-by-block basis in this design. The controller enables the processing units simultaneously after making the control information available for them. Using this scheme, a dual-port RAM is needed between two interconnected processing units, where read and write operations can be active at the same time. In addition, the controller provides one set of registers storing the control information for each processing unit. After all the processing units finish decoding the current block, controller transfers the parameters from one register to the next one in pipeline as shown in Fig.5. In other words, the registers are updated by the control information in the previous registers. For example, if PU1 has finished decoding the current block, the controller can't enable it to decode next block until the PU2 and PU3 have finished their respective decoding task. Meanwhile, the control information in the registers can't be changed during the time. After all the three processing units finishing decoding their current block, the control information is shifted in the control path and then all the processing units are enabled again at the same time.

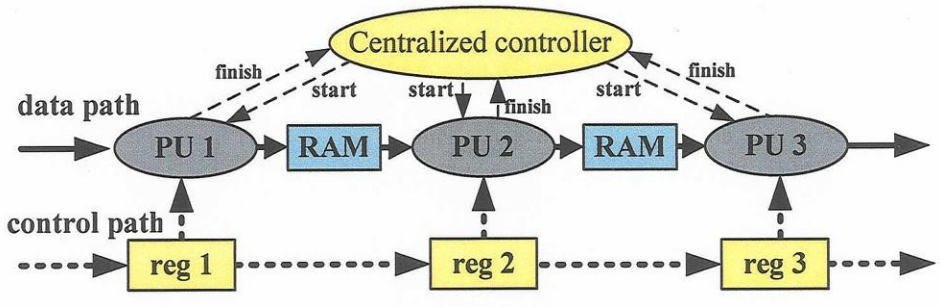

Fig.5. Centralized Control scheme

With this control scheme, the controlling logic is easy to be realized and only one set of registers is needed for each processing unit. Moreover the function of synchronizing the decoding data and control information could be realized in each processing unit simply. But a dual-port RAM is used between two interconnected PUs, which provides storage for two block data, so the Centralized Control scheme features a little high hardware cost for on-chip memory.

3.2 Decentralized Control scheme

Decentralized Control scheme is a data driven architecture. Processing unit operates only if the input data are presented and output buffer have enough space to store the result. In other words, each unit is "driven" by data. In order to work dynamically, the Decentralized Control scheme should have the following two features in common.

- Both input and output data buffers are organized as FIFO, which provides signal of buffers state to indicate whether it has enough data for processing or enough space for storing result.
- Each processing unit is controlled by a local FSM (Finite State Machine), which checks the states of both input and output FIFO before enabling an operation. If the input data FIFO has sufficient data and the output FIFO has enough space for storing the result, the operation is issued. Otherwise it is paused.

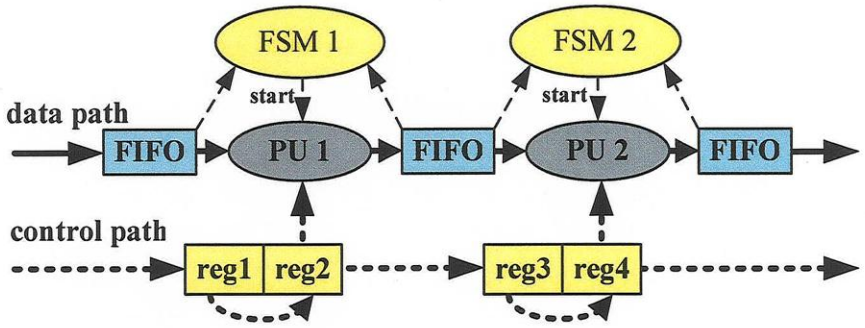

Fig.6. Decentralized Control scheme

In this case, different processing units may not be enabled at the same cycle. The controller provides two sets of registers to store the control information for each processing unit. One provides a storage space for storing the control information which is shifted from the registers of previous processing unit. The other is used to store the control information being used by the current processing unit. For example, as shown in Fig.6, PU1 and PU2 are under the control of their respective FSM. If PU1 has finished the current decoding task and PU2 has not, the controller shifts the control information from reg2 to reg3, instead of reg4 whose information is used by PU2 currently. When PU2 finishes the current decoding task, reg4 is updated by the information in reg3. And at the same time, the control information in reg4 is shifted to the next registers.

Contrast to Centralized Control scheme, Decentralized Control scheme has an efficient hardware cost. In Centralized Control scheme, each RAM should provide storage space of two blocks. In Decentralized Control scheme, storage space of one block or even less is enough. So the Decentralized Control scheme can greatly minimize size of the connecting buffers. Furthermore, because each processing unit is enabled only by the state of the input and output buffers, this scheme provides a more relax environment for the processing units and makes them operate more efficiently. But because processing in each processing unit relies to the signals of input and output buffers state, instead of other processing units, it is difficult to realize the synchronization of the decoding data and control information.

3.3 Combined centralized and decentralized control scheme

Analyzing the architecture of decoder system and the feature of each processing unit, we develop a system controller combined the centralized and decentralized control scheme. Centralized Control scheme is used to control RRD, ITP, INTRA, Deblocking Filter and WB, because a little control information needs to be transferred for them and executing cycles of each processing unit are almost the same. When all of them finish decoding their current block, the System Controller shifts the control information in the registers, and enables them to decode the next block. VLD, AGU, baseline unit and DIS use the Decentralized Control scheme. A local FSM is needed for each Decentralized Controller to checks the state of both input and output FIFO of corresponding processing unit. Only if both input and output buffer are available, the operation of processing unit can be issued.

Combining the two control schemes, the System Controller provides a high efficient communication among all the processing units. The size of the on-chip memory for storing the decoding data and control information is minimized greatly, without complex control logic.

4. SDRAM INTERFACE

SDRAM interface includes Arbiter and Controller. The Arbiter schedules all the requests from the processing units and the Controller controls the operation of the SDRAM. In some traditional designs, a strategy named non-preemptive or fixed priority bus scheduling[7] is employed to handle requests for SDRAM. With priority algorithm, the task with lower priority needs much buffer to store the data if the bus is occupied by those tasks with higher priority. Furthermore, if the time of occupying the bus by tasks with higher priority is long, and the buffer size for tasks with lower priority is not big enough, conflicts will happen. The conflicts will block the video decoding process. If all the tasks are blocked, there is "vacuum state" for the bus because of no request. A time division multiplexed task scheduling (TDM) is adopted in some design[5]. But this algorithm is only available for the case that both the start time to request for the bus and executing time on the bus for each task is fixed. In this paper, a bus-arbitration algorithm named Token Ring algorithm is proposed to avoid the conflicts on the SDRAM bus and improve the efficiency of the bus utilizing.

4.1 Seven tasks connected to SDRAM

In the architecture of the decoder, the data from seven different paths are routed to/from the SDRAM via the bus. As shown in Fig.1, Path 1 is from the BSF (Bitstream FIFO) to the SDRAM. Path 2 is from the SDRAM to VLD. Path 3 is from the SDRAM to RRD that gets the reference data. Path 4 is from WB, which writes back the reconstructed MB, to the SDRAM. Path 5 is from the SDRAM to DIS unit for display. Path 6 is a path to transfer MV (Motion Vector) for AGU, and Path 7 is a path to transfer samples of up MB line for Deblocking Filter. So the Arbiter receives requests from seven tasks: BSF VLD, RRD, WB, DIS, AGU and Deblocking Filter.

4.2 Token Ring algorithm

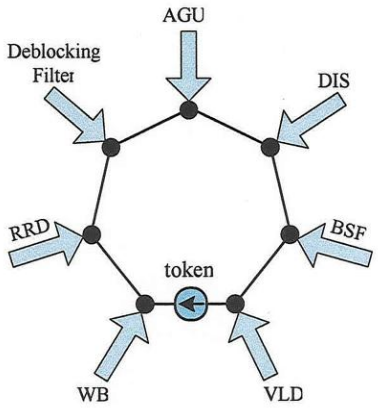

Fig.7. Token Ring algorithm in SDRAM Arbiter

Based on the control scheme in Sec.3, a Token Ring algorithm in SDRAM Arbiter is developed, which is shown in Fig.7. The seven requests make up a ring topology, in which all the requests act as nodes in a ring. The SDRAM Arbiter passes the signal, called a token, from one node to another in one direction around the ring. When the token arrives at a node, the node can occupy the bus if a request is occurred on the node. Otherwise, the token is transferred continually to the next node. When the node occupies the bus, the token can not be transferred backwards. In a word, the bus is allocated to the node on which the token is available and the request is occurred.

4.3 Algorithm analysis

The WB, RRD and Deblocking Filter processing units decode block-by-block under the Centralized Control scheme. The three tasks start to work at the same time, so they request for the SDRAM bus simultaneously. In order to reduce the

buffer size, requests from the three tasks are arranged closely in the token ring, as shown in Fig.7. This arrangement can guarantee the least time for the three tasks finishing the current block decoding.

As to AVS Jizhun Profile@4.0 level, the up bound of the decoding cycles for each block is 333 with the system working at 81MHz clock frequency. The time of the up bound cycles is called a block slot, which is obtained by averaging the frame time to each block. The executing cycles of occupying the bus for seven tasks are listed in Table.2. During a block slot, even if all the tasks request for the bus, the token can finish one revolution around the ring, because the total executing cycles of all the tasks are 230, which is lower than the up bound (333). That's to say, in a block slot, all the tasks which are need occupying the bus can finish one block decoding. The buffer size, shown in Table.2, is storage space for each task in a block slot.

Table.2. Executing cycles and buffer size for seven tasks

Tasks		WB	RRD	Deblocking Filter	AGU	DIS	BSO	BSI	Total
Cycles per block		17	102	25	3	63	10	10	230
Buffer Size	Token Ring	128byte	192byte	40byte	16byte	1096byte	256byte	256byte	1984byte
	Fixed Priority	128byte	192byte	40byte	32byte	2192byte	512 byte	512 byte	3608byte

Owning to the feature of Token Ring algorithm, the conflict problem of occupying the bus is simply resolved compared with the Fixed Priority schedule. If the schedule is a priority one, the task with the highest priority is executed first during bus contention. The data of those tasks with lower priority have to wait in the buffer, so the buffer size is very large. Considering to the WB, RRD and Deblocking Filter as the highest priority level, the buffer for other tasks has to store another block data. The size is described in Table.2. The total size in the Token Ring algorithm is 55% of that in the Fixed Priority schedule. Using the Token Ring algorithm for the bus-arbitration, computation and I/O operation are balanced to minimize the sizes of embedded buffers in the decoder. Furthermore, the implementation for the algorithm is simple.

5. SYSTEM SIMULATION RESULTS

The simulation is shown that, the decoder architecture described above takes 2.96E+6 cycles to decode a picture of AVS Jizhun Profile@4.0 level and meets the requirement of decoding in real time, when it operates with the clock cycles of 81MHz.

The circuit was synthesized using 0.18μm Artisan CMOS cell library by Synopsys Design Analyzer with critical path constrain set to 9.8ns (102MHz). The total gate count is about 461K, which includes 369K logical gate count and 92K SRAM (37K for FIFO and 55K for dual RAM) for buffering both data and control information among the processing units.

6. CONCLUSION

In this paper, a system architecture of AVS decoder is represented. Some design idea, such as the architecture for VLD and Deblocking Filter, the system control scheme and the basic frame of system architecture can be expanded to H.264 because of the similarity between the two video coding standards. Furthermore, the bus-arbitration can be adopted in H.264 decoder which also has many requests for the SDRAM bus. In this decoder architecture, a combined parallel decode algorithm based on PLA mainly and serial decode algorithm is adopted for VLD that can decode in real-time at low clock frequency with low hardware cost. Using proper storage organization and pipelining, the Deblocking Filter can increase the efficiency of getting data, and support both horizontal filtering and vertical filtering on the same circuit (a parallel-in parallel-out FIR filter). A bus-arbitration scheme using Token Ring algorithm is designed to avoid the conflicts on bus and minimize the buffer size. Designed based on the feature of all processing units, our control scheme can achieve a high efficient communication among all the processing units.

REFERENCES

1. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC), Mar. 2003.
2. Audio Video Coding Standard Workgroup of China (AVS), Video Coding Standard FCD1.0, Nov.2003.
3. Nam Ling, "An Efficient Controller scheme for MPEG-2 video decoder" IEEE Trans. on Consumer Electronics, vol. 44, No.2, May 1998.
4. Hae-Yong Kang, "MPEG4 AVC/H.264 decoder with scalable bus architecture and dual memory controller". 2004.ISCAS '04. Proceedings of the 2004 International Symposium on Circuits and Systems, Volume: 2, 23-26 May 2004.
5. Hui Wang, "SDRAM bus schedule of HDTV video decoder". Proc. SPIE Int. Soc. Opt. Eng. 4674, 150(2001).
6. Hui Wang, "A Novel HDTV Video Decoder and Decentralized Control Scheme". Consumer Electronics, IEEE Transactions on, Volume: 47, Issue: 4, Nov. 2001
7. Chia-Hsing Lin, Chein-Wei Jen , "On the Bus arbitration for MPEG2 Video Decoder", *Proceedings of Technical Papers*_International Symposium on VLSI Technology, Systems, and Applications, pp. 201 -205, 1995.
8. Yu-Wen Huang, To-Wei Chen, Bing-Yu Hsieh, Tu-Chih Wang, Te-Hao Chang, and Liang-Gee Chen, "Architecture Design For Deblocking Filter in H.264/JVT/AVC" Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on, vol. 1, pp. 693–696, 6-9 July 2003.