Smart Bus Arbiter for QoS control in H.264 decoders

Chanho Lee

Abstract—**H.264 decoders usually have pipeline architecture by a macroblock or a 4 × 4 sub-block. The period of the pipeline is usually fixed to guarantee the operation in the worst case which results in many idle cycles and higher data bandwidth. Adaptive pipeline architecture for H.264 decoders has been proposed for efficient decoding and lower the requirement of the bandwidth for the memory bus. However, it requires a controller for the adaptive priority control to utilize the advantage. We propose a smart bus arbiter that replaces the controller. It is introduced to adjust the priority adaptively the QoS (Quality of Service) control of the decoding process. The smart arbiter can be integrated the arbiter of bus systems and it works when certain conditions are met so that it does not affect the original functions of the arbiter. An H.264 decoder using the proposed architecture is designed and implemented to verify the operation using an FPGA.**

Index Terms—**H.264, QoS, implementation, AHB, arbiter, arbitration policy**

I. INTRODUCTION

H.264 is a video coding standard developed recently by JVT (Joint Video Team) of ISO/IEC and ITU-T. It is also called MPEG-4 part 10 AVC (Advanced Video Coding) [1]. Its compression rate is higher by about 25-40% than that of the MPEG-4 advanced simple profile [2]. It includes new features such as 1/4-pixel inter-prediction with variable block size, multiple reference frames, intra prediction, and context-based adaptive entropy coding: context-based adaptive variable length coding (CAVLC) and context-based adaptive binary arithmetic coding (CABAC) [1]. The new features require complicated compression algorithm and a lot of arithmetic calculation to offer the high compression rate and the image quality. The computational complexity of the H.264 decoder has been increased twice compared with the MPEG-4 decoder.

H.264 decoders usually have pipeline architecture by a macroblock (MB) or a 4 × 4 sub-block. The complexity and time for processing data at each pipeline stage are variable depending on the images because the encoding methods are different depending on the images. The period of a pipeline stage should be determined so that the most time-consuming stage can process the block of data in the worst case, which requires hardware with higher performance than that for normal case. The worst case would not occur so often that the processing units are often idle. Decoder hardware with lower performance can decode a movie by eliminating the idle cycles.

According to the results of the runtime analysis using H.264 decoding software, motion compensation (MC) uses up to 55% of total decoding time. That is, the efficient architecture of the MC is crucial for the design of the decoder [3, 4]. The performance of the MC is determined by two factors; one is the computation performance and the other is the time for reading reference pixel data from memory. H.264 decoders usually have three units that get access to the reference frame memory. The first one is the MC as mentioned above. The second one is a deblocking filter (DF), and the last one is a data_out unit that transfers image data to a display device or a host system. If a bus to a host system or a display unit is not separate, a variable length

Manuscript received Nov. 27, 2010; revised Feb. 7, 2011.
This paper was invited from ISOCC 2010
School of Electronic Engineering, Soongsil University, Sangdo-dong, Dongjak-gu, Seoul, 156-743, Korea
E-mail : chlee@ssu.ac.kr

decoder (VLD) which reads NAL (Network Layer Abstraction) stream also gets access to the memory bus. The three units compete in access to memory, and only one 'read' or 'write' transaction is possible at a time because an inexpensive single DRAM chip is usually used for the reference frame memory. The MC should have the highest priority in access to memory as it is the most time-consuming unit in a decoder as mentioned above. However, the priority needs to be adjusted because other units can also block the decoding process if either one cannot read from or write to the memory. For example, the DF must not write pixel data of the current frame to memory before all data of the earliest frame are read out by the data_out unit when the reference frame memory is full of pixel data. The above situation can happen for the adaptive pipeline architecture [5]. Therefore, the memory bus must provide smart arbitration policy for the three units to get access to the reference frame memory [6].

In this paper, we propose an improved smart arbiter and efficient H.264 decoder architecture using it. The proposed architecture is designed for the smart arbiter for both the QoS (Quality of Service) control in the memory bus and the normal network arbitration. An H.264 decoder is designed according to the proposed architecture. The decoder employs an adaptive pipeline in which periods of a pipeline stage varies depending on the encoded data so that the idle cycles are minimized. The stages in the pipeline communicate with each other by hand-shaking so that data are transferred without a controller for decoding processes. The controller for the memory access by MC, DF, VLD, and the data_out units is not necessary due to the smart arbiter. The smart arbiter and the adaptive pipeline architecture relieve the tight timing requirement of the internal computing units so that the replacement or modification of the units is easy. The decoder is implemented using an FPGA to verify the operation.

II. Proposed Architecture

1. Adaptive pipeline architecture

H.264 decoders usually consist of VLD, inverse transform and quantization (ITQ) unit, intra prediction (IP) unit, the MC unit, and the DF. The MC unit requires pixel data of multiple reference frames which may occupy tens of MBs. The amount of memory is too large for internal memory and so external memory is necessary. The DF writes pixel data of the current frame to the reference memory after filtering because the pixel data of the current frame become those of a reference frame when decoding the pixel data of the following frames. A data_out unit is necessary to read pixel data of the reference frame memory and to send them to a display device or a host system although it is not directly related to decoding. A single DRAM is usually employed as the reference memory to reduce the cost and the volume. As a result, three units compete to get access to the reference frame memory.

The complexity and the time for processing a macroblock are variable depending on the type of image. If the period of processing a macroblock is fixed, it should be large enough to accommodate the worst case [6]. As a result, the processing units will be idle frequently as shown in Fig. 1. The period is determined to accommodate the macroblock M0 which is assumed to be the worst case, and the decoder has idle cycles for M1 and M2. The pipeline is usually controlled by a central controller in the case.

The data transfers between computing units are not directed by a central controller in the adaptive pipeline architecture [5]. For example, as soon as parameter are decoded by Ex-Golomb(EG) decoder in the VLD, they are delivered to the MC or IP to obtain reference data. Fig. 2 shows the processing sequence of the adaptive

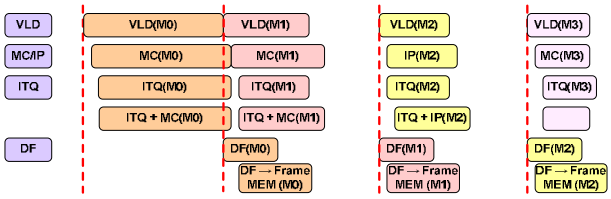


Fig. 1. Processing sequence of decoder pipeline with a fixed period [5].

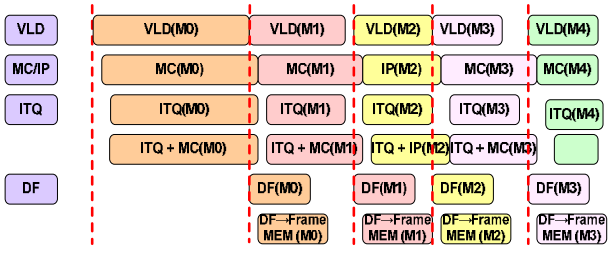


Fig. 2. Processing sequence of adaptive pipeline [5].

pipeline. The data are delivered using hand-shaking communication without a central controller as soon as the next stage is ready to receive the data. The periods of processing macroblocks are determined by the encoding types, and they are variable. At least one stage is working at any time because the data are delivered whenever it is possible. As a result, the idle cycles are minimized, and the overall decoding time is reduced.

2. Smart bus arbiter

The MC, DF and data_out unit need to get access to the reference frame memory, and the VLD and the data_out unit need to get access to the buffer memory of a host system as mentioned above. The adaptive pipeline gives out a problem of contention in access to the reference frame memory and the buffer memory since it is not guaranteed that the three units read or write pixel data during the variable period. In addition, the data_out unit writes pixel data to the frame memory of the display device and the VLD reads NAL stream data although it requires very small bandwidth. The absence of a centralized controller and the variable period may not give enough time for the three units to get access to the reference frame memory one by one in the period. As a result, the pipeline may be blocked or the pixel data of a frame may be overwritten without being read out to a display device. The frame rate for the display device should also be maintained so that the decoder does not play a movie too fast or slow.

A control unit called DMACA (Direct Memory Access Controller with Arbiter) was employed in the H.264 decoder with the adaptive pipeline in order to minimize the pipeline hazard [5]. The DMACA receives interrupt signals from the three units and determines whose request will be processed. Only the DMACA gets access to the reference memory and delivers data to the three units. It is inefficient for the DMACA to distribute data because twice the numbers of transfers are required for the delivery of pixel data.

In order to solve the problem, a smart arbitration scheme is introduced in the arbiter of the bus system as shown in Fig. 3 [6]. The arbitration scheme for the H.264 decoder was mixed with the AHB arbitration logic. The smart arbiter logic is now separated from the AHB arbiter and is configurable by a host system with an

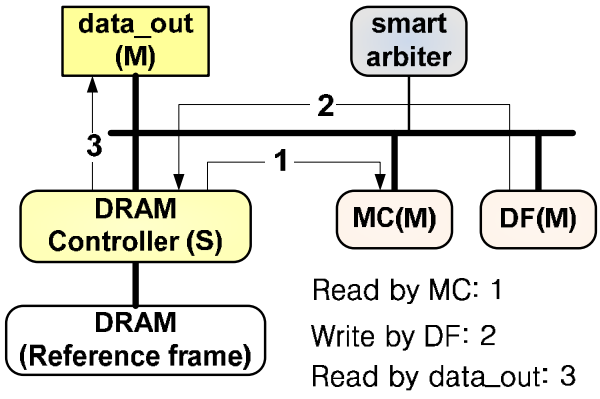


Fig. 3. Smart arbiter to solve the problem of adaptive pipeline.

improved arbitration scheme. The arbiter reads the addresses of the data transfer and monitors the status of the reference frame memory. The MC has the highest priority in the normal condition to increase the performance of the decoder. If the reference frame memory is full and even the pixel data of the earliest frame have not been read out, the arbiter gives higher priority to the data_out unit to prevent the hazard of missing frames due to the overwriting by the DF. There is one more condition that the data_out has higher priority than the MC unit. The data_out unit must read out pixel data to a display device or the host system at a given frame rate so that the decoded motion picture is displayed at the same rate as the original one. The smart arbiter monitors the rate that data_out unit transfers pixel data to a display device or a host system, and gives it the higher priority than others or does not give grant signals at all when the frame rate does not meet the required frame rate. Therefore, the smart arbiter provides the QoS control in the memory bus as well as the bus arbitration.

The three units have the master interface so that they can read or write data without a direct memory access controller (DMAC) [6]. It will reduce the number of data delivery cycles to a half of that with the DMACA. They also determine when they get access to the reference memory for them to increase the performance.

The AHB arbiter can have various arbitration policies such as fixed priority [7], Round-Robin [8], TDMA [9], and Lottery [10]. The TDMA and the Lottery scheme can allocate weighted chances of arbitration to certain masters. If the weight factor is selected appropriately it may work for the memory bus of H.264 decoders without the smart arbiter. I design an AHB arbiter which

is compliant to AMBA 2.0 specification with various arbitration policies, and apply it to a decoder. I simulate the decoder for the various arbitration policies of the arbiter for the same time period, and the image size is QVGA. The applied arbitration policies are the fixed priority (F), Round-Robin (RR), Round-Robin with fixed scheme as the second arbitration (RR+F), TDMA (T), TDMA with the fixed scheme (T+F), Lottery scheme (L), and Lottery with the fixed scheme (L+F). The DF is not blocked to write data to the reference frame memory when the memory is full and the frame rate is not controlled since the DMACA is eliminated. Regardless of the arbitration policies, the decoded pixel data have errors of missing pixel blocks although the locations of errors are different. The normal arbitration polices cannot control the QoS but controls only the allocation of the bandwidths of masters. Increasing the bandwidth of bus network by using high performance network such as AMBA AXI cannot solve the problem, either [11].

III. Design of H.264 Decoder

An H.264 decoder is designed using the proposed architecture. Fig. 4 shows the block diagram of the decoder. It consists of VLD, ITQ-MC/IP, DF and data_out units. Coefficients and parameters are delivered through a dedicated data path using hand-shaking communication. The pixel data of the reference frame and the decoded pixel data are transferred through an AHB network for the memory access. The reference frame data are stored in a PSRAM (Pseudo-SRAM). The ITQ-MC/IP, DF and data_out units have AHB master interfaces so that they can get access to the memory directly. It reads the pixel data of the reference frames from the reference frame memory directly. They also have AHB slave interfaces which provide internal parameters and data for debugging purposes. The slave interface of the data_out unit is also used to configure the decoder operation. The ITQ-MC/IP unit includes ITQ, MC and IP units which share a hand-shaking interface through which parameters and coefficients are delivered to the MC/IP and ITQ, respectively. It also has an interface with an internal memory for the IP unit. The DF has an interface with the internal memory to store pixel data of a macroblock. The data_out unit has an AHB master interface to read the pixel data from the reference frame memory and to send them out to a display device or a host system. The decoder is connected to a host system through an AHB-to-AHB bridge. SDRAM is used to store the NAL data and as a frame buffer of a TFTLCD controller to implement the decoder on my prototyping board.

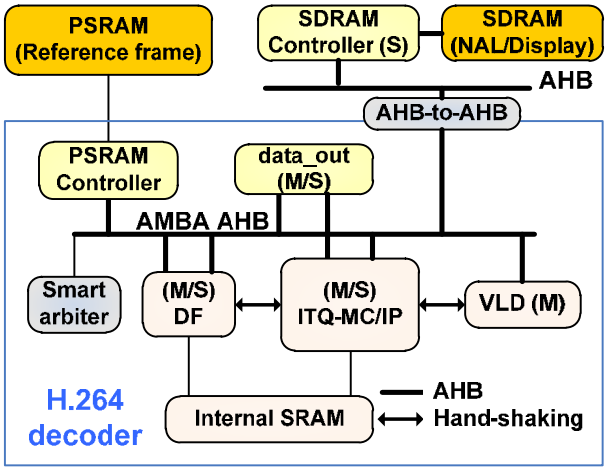


Fig. 4. Block diagram of the H.264 decoder based on the proposed architecture.

The smart arbiter of the AHB network has specialized priority logic to arbitrate the requests from the four masters. The VLD unit has the highest priority although the frequency of the request is very low. The MC unit has the second highest priority and the DF has the lower priority than the MC unit. However, if the buffer of the DF is full and the DF does not get a grant signal due to the MC, the DF blocks the pipeline and the MC also stops. As a result, the MC does not request a bus master, and the DF can write pixel data to the reference frame memory to empty the internal buffer so that the MC unit can work again. The MC and the DF can get access to the reference memory alternatively in this way.

If the bandwidth of the AHB network is not large enough or the period for a macroblock is not long enough, the data_out unit may not get grant signals from the arbiter. The smart arbiter monitors the status of the reference frame memory. If the reference frame memory is full and the pixel data of the earliest frame have not been read out, the smart arbiter gives the highest priority to the data_out unit so that the DF does not overwrite data to the location of the earliest frame without being read out. Overwriting data without being read out means that the corresponding frame is lost. The smart arbiter

also monitors the frame rate that the data_out delivers the pixel data. The frame rate was controlled by the data_out unit in the previous work [6]. It gives the higher priority to the data_out unit than others or stops the entire pipeline if the frame rate does not meet the requirement. The arbiter does not violate any AMBA AHB specification since the behavior for the smart arbitration mentioned above is implemented in the priority logic of the AHB arbiter and it can behave as a normal AHB arbiter by turning off the monitoring function for the reference frame memory. Fig. 5 shows the block diagram of the smart arbiter. It consists of an AHB arbiter and a flow control unit. The AHB slave interface is used to configure the flow control unit and is not used at runtime. Without the output signals from the flow control unit, the arbiter behaves like a normal AHB arbiter.

The decoder is designed using Verilog-HDL and the synthesis results are compared with others as shown in Table 1. The memory is a temporary storage in the decoder and does not include the reference frame memory. The operating frequency corresponds to the image size. The decoder is also implemented using an FPGA to verify the operation. Fig. 6 shows the verification result that a sample movie is decoded and displayed using a prototyping board.

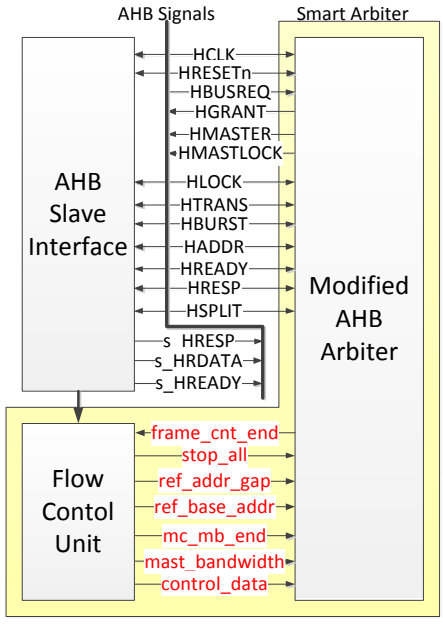


Fig. 5. Block diagram of the proposed smart arbiter.

Table 1. Synthesis results of decoders

	Proposed	[12]	[13]	[14]
Gate count	225,372	242,705	217,428	456,598
Memory [bits]	70,848	23,200	81,756	164,864
Operating frequency	25 MHz (0.18 μm)	54 MHz (0.25 μm)	120 MHz (0.18 μm)	100 MHz (0.18 μm)
Image size	480p30	CIF30	1024p30	1080HD

Fig. 6. Verification results of the H.264 decoder based on the proposed architecture using an FPGA prototyping board.

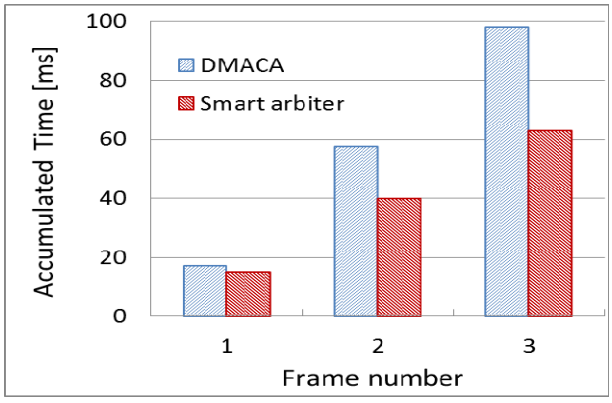


Fig. 7. Comparison results of the performance of the decoders with the smart arbiter and the DMACA.

Fig. 7 shows the comparison results of the performance of the decoders with the smart arbiter and the DMACA. The performance is measured using the accumulated decoding time until each frame of 'Foreman' is displayed. The operation frequency is selected so that the decoder with the DMACA can decode images at 30 frames per second. The decoding time for the decoder with the smart arbiter is obtained after the idle time to keep the frame rate of 30 is eliminated. The performance is improved about 30% which is smaller than the expected value of 50% because of the overhead to pass through the AHB-to-AHB bridge.

IV. CONCLUSIONS

The variable pipeline architecture for H.264 decoder is

proposed to reduce the decoding time by removing the idle cycles. The parameters and the coefficients are transferred through a dedicated data path using hand-shaking communication. The adaptive pipeline architecture does not need a centralized controller, and makes it easy to design a decoder. The problem of access to the reference frame memory in the adaptive pipeline architecture is solved using an improved smart bus arbiter with the function of the QoS control. The smart bus arbiter has priority logic that monitors the status of the reference frame memory and adaptively adjusts the priority of the masters so that the performance of the decoder is optimized. The QoS control cannot be provided by various arbitration policies. An H.264 decoder is designed based on the proposed architecture, and is also implemented using an FPGA to verify the operation. The decoder with the smart arbiter shows better performance by 30% than that with a central flow controller.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2010-0021225) and the EDA tools were supported by IDEC.

REFERENCES

[1] Joint Video Team, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May, 2003.

[2] C.-Y. Tsai, T.-C. Chen, T.-W. Chen, L.-G. Chen, "Bandwidth Optimized Motion Compensation Hardware Design for H.264/AVC HDTV Decoder," *Proceedings of 48th Midwest Symposium on Circuits and Systems*, Vol.2, pp.1199-1202, Aug., 7-10, 2005.

[3] M. Alle, J. Biswas, S. K. Nandyl, "High performance VLSI implementation for H.264 Inter/Intra prediction," *Proceedings of IEEE International Conference on Comsumer Electonics*, pp.1-2, Jan., 2007.

[4] W.-N. Lie, H.-C. Yeh, T. C.-I. Lin, C.-F. Chen, "Hardware-efficient computing architecture for motion compensation interpolation in H.264 video coding," *Proceedings of IEEE International Symposium on Circuits and Systems*, Vol.3, pp. 2136-2139, May 23-26, 2005.

[5] C. Lee, "Design of Low Power H.264 Decoder using adaptive pipeline," *Journal of IEEK*, Vol.47SD, No.9, pp.1-6 , Sep., 2010.

[6] C. Lee and S. Yang, "Design of an H.264 Decoder with Variable Pipeline and Smart Bus Arbiter," *Proceedings of 2010 International SoC Design Conference*, pp.432-435, Oct., 22-23, 2010.

[7] T. N. Mudge, J. P. Hayes, D. C. Winsor, "Multiple Bus Architectures," *IEEE Computer*, Vol.20, No.6, pp.42-48, Jun., 1987.

[8] A. B. Kovaleski, "High-speed bus arbiter for multiprocessors," *1EE PROC*, Vol. 130, Pt. E, No. 2, pp. 49-56, Mar., 1983.

[9] K. A. Kettler, J. P. Lehoczky, and J. K. Strosnider, "Modeling Bus Scheduling Policies for Real-time Systems," *Proceedings of 16th IEEE Real-Time Systems Symposium*, pp.242-253, Dec., 5-7, 1995.

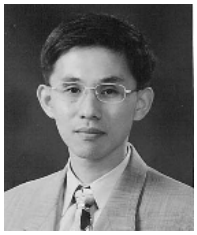
[10] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "LOTTERYBUS: A new high performance communication architecture for system-on-chip designs," *Design Automation Conference*, pp.15-20, Jun., 2002

[11] S. Na, S. Yang, and C.-M. Kyung, "Low-Power Bus Architecture Composition for AMBA AXI," *Journal of Semiconductor Technology and Science*, pp. 75-79, Vol.9, No.2, Jun., 2009

[12] S. M. Park, M. Lee, S. Kim, K.-S. Shin, I. Kim, H. Cho, H. Jung, D. Lee, "VLSI Implementation of H.264 Video Decoder for Mobile Multimedia Application," *ETRI Journal*, Vol.28, No.4, Aug., 2006.

[13] T.-C. Chen, C.-Jr Lian, L.-G. Chen, "Hardware Architecture Design of an H.264/AVC Video Codec," *Proceedings of ASPDAC 2006*, pp.750-757, Jan., 24-27, 2006.

[14] T.-M. Liu, T.-A. Lin, S.-Z. Wang, W.-P. Lee, K.-C. Hou, J.-Y. Yang C.-Y. Lee, "An 865-μW H.264/ AVC Video Decoder for Mobile Applications," *Proceedings of ASSCC 2005*, pp. 301-304, Nov., 1-3, 2005.



Chanho Lee received the BS and MS degrees in electronic engineering from Seoul National University, Seoul, Korea, in 1987 and 1989, and the PhD degree from the University of California, Los Angeles, in 1994. In 1994, he joined the semiconductor R&D center of Samsung Electronics, Giheung, Korea. Since 1995, he has been a faculty member of the School of Electronic Engineering, Soongsil University, Seoul, Korea, and he is currently Professor. His research interests are in SoC on-chip-network, SoC platform, memory controller, the design of H.264 codec, and 2D/3D graphic processor. He is a senior member of IEEE.