A hardware/software co-design approach for multiple-standard video bitstream parsing

Sha Shen, Huibo Zhong, Yibo Fan, Xiaoyang Zeng

The State key lab of ASIC and system, Fudan University, Shanghai 200433, China
Email: 10110720050@fudan.edu.cn

Abstract — In this paper, a hardware/software co-design approach is proposed to parse the video bitstream which conforms to various video compression standards. The layered structure of the syntax elements in video bitstreams is analyzed. Then a hardware/software partition is proposed accordingly. Due to the high data rate, syntax elements in slice data and lower layers are commonly parsed by hardware. As for syntax elements in slice header and upper layers, we proposed a hw/sw co-design approach in order to combine the advantage of hardware acceleration and software flexibility. specific hardware accelerators are designed to parse these codes. But the parsing process of these codes in slice header and upper layer is controlled by software instead of hardware Finite state machine (FSM). This approach can speed up the process of Variable-Length Decoding (VLD) while it still has the flexibility to support multiple video coding standards.

Index Terms —bitstream parsing, fixed-length code, Exponential Golomb code, VLD

1. Introduction

As the multimedia technology advances in the recent years, a lot of new video coding standards (such as H.264/AVC[1], VC-1[2], AVS[3], etc.) have emerged and come into application along with the traditional standards like MPEG-2[4] or MPEG-4[5]. New video coding standard like HEVC/H.265[6] aiming at the ultra-high application is also in progress. The video decoder, especially the one used in consumer electronics like digital TV, smart phone, STB, is desired to support multiple coding standards. It's a great challenge to design a multiple standard video decoder due to the various specification of each standard. A flexible architecture is needed to achieve this requirement.

The first step of video decoding is to parse the video bitstream and decode each syntax element. Video bitstream in all video standards are organized in a similar layered structure which is shown in Fig.1. A video sequence is composed of several pictures. Each picture is divided into several slices. Each slice can be further divided into several blocks. In HEVC/H.265, the block size is 64x64 pixels square while in previous standards the block size is generally set as 16x16.

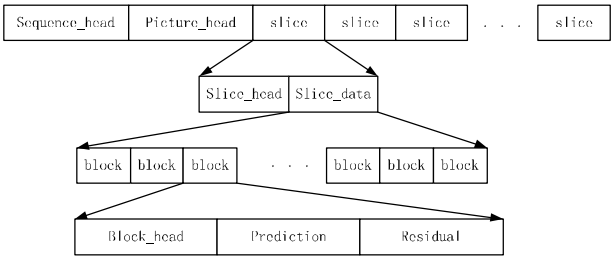

Fig. 1. The layered structure of video bitstream

Various entropy coding method are used in the video compression standards. In video standards like MPEG-2, MPEG-4, the syntax elements in sequence layer and picture layer are coded in fix-length code. In order to improve the coding efficiency, AVS, H.264 and HEVC adopt the exponential Golomb code. As for the syntax elements of intra/inter prediction and residual in block layer, specific VLC codes are used for each different video standard. In H.264 and HEVC, the context adaptive arithmetic code (CABAC) is also introduced as an alternative to the VLC code. The arithmetic code can improve the coding efficiency further at expense of more computational complexity.

In this paper, we proposed a HW/SW co-design method to parse the video bitstream. Syntax elements in slice data and lower layer are parsed by dedicated hardware while the syntax elements in slice header and upper layer (including picture header and sequence header) are parsed in a HW/SW co-design approach.

The rest of this paper is organized as following. In Section 2, the layered bitstream structure for various video standards is analyzed. In section 3, Several previous architectures for video stream parsing are surveyed. Their advantage and disadvantage are also discussed. Then an improved HW/SW co-design architecture is proposed. The experiment result of our architecture is given in section 4. Finally, a conclusion is given in section 5.

2. Analysis of bitstream structure

The video compression technology has evolved a lot in the last twenty years. Various standards (such as MPEG-1, MEPG-2, MPEG-4, H.264, AVS, etc.) have been proposed by various groups and organizations. Very

Table 1. The layered structure of video bitstream

Layer	Syntax elements	Entropy coding method	Implementation choice			
			[8]	[10]	[11]	This work
1	Start code	Fixed length	HW	SW	SW	HW-SW
	Emulation prevent byte	Fixed length				
2	Sequence header	Fixed length, Exp-Golom	HW	SW	SW	HW-SW
	Picture header	Fixed length, Exp-Golom				
	Slice header	Fixed length, Exp-Golom, VLC lookup table,				
	User defined data	Fixed length				
3	Intra/Inter prediction	Fixed length, Exp-Golom, VLC lookup table, CABAC	HW	SW	HW	HW
	Residual for DCT coefficient	Fixed length, VLC lookup table, CABAC				

Note: HW means hardware only, SW means software only, HW-SW means hardware-software co-design.

popular acceptance of these standards has been got in a wide range of applications. As more complex coding tools like CABAC, deblocking filter, multiple reference picture, adaptive loop filter and tree-block partition have been introduced into the latest video standards[1],[6], the coding efficiency has been doubled[7].

But the basic structure of the video bitstream remains almost the same. The video bitstream can be divided into three layers: (1) start code. A start code indicates the beginning of a video sequence. This code is presented in fixed length code among all video standards. The parsing process of all other syntax elements can start only after the start code is found. In some video standard like AVS or H.264, the same bit pattern as start code may occur in the middle of bitstream due to the use of exponential Golomb code. A specific byte called emulation prevent byte will be inserted into this pattern in order to make it different with the start code. (2) Header information and user define data. Video sequence header contains the parameters such as video profile and level, picture width and height. Picture header contains the parameters such as slice group number, QP offset. Slice header contains parameters such as slice type, frame number. Syntax elements for header are usually coded by fixed length code or exponential Golomb code. User defined data can be used to specific needs which is not defined in the standard. For example, the description of the video contents or the provider information of the video can be inserted into the video bitstream. User defined data are usually presented as fixed length code. (3) Prediction information and residuals. Inter/intra prediction is essential to remove the spatial and temporal redundancy of a video sequence. The error between prediction and actual picture is transformed by DCT and quantization. The result is called "residuals". Prediction information and residuals constitute the most part of a coded video bitstream. In order to achieve better coding efficiency than exponential Golomb code or fixed length code, some complex entropy coding algorithms like context adaptive VLC or context adaptive arithmetic coding are used for prediction information and residuals.

3. Proposed architecture

Table 1. shows how the syntax elements are organized in the layered structure of video bitstream. The possible entropy coding method of syntax elements in the three layers are also listed in this table.

Several architectures have been proposed to parse the video bitstream. The implementation choice to parse each syntax element is also shown in Table 1. These architectures can be categorized into three types: (1) Hardware parser. The video bitstream is parsed by pure hardware. Li[8] proposed a variable-length decoder for MPEG-2 video decoder. Xu[9] proposed a circular buffer based H.264/AVC bitstream parser. Hardware implementation is efficient for one single standard but not flexible to support multiple standards. (2) Dedicated bitstream processor. Chang[10] proposed a bitstream parsing processor with 4 pipeline stages for MPEG-4 video decoder. Special instruction set is also designed to facilitate the parsing process of prediction information and residuals. This architecture is feasible to support multiple standards. The disadvantage of this approach is that quite a lot of effort must be taken to design a dedicated processor and develop the associated software tool-chain. (3) HW/SW co-design approach. Liu[11] proposed a HW/SW co-design approach to parse the video bitstream. Syntax elements in layer 1 and layer 2 are parsed by software while syntax elements in layer 3

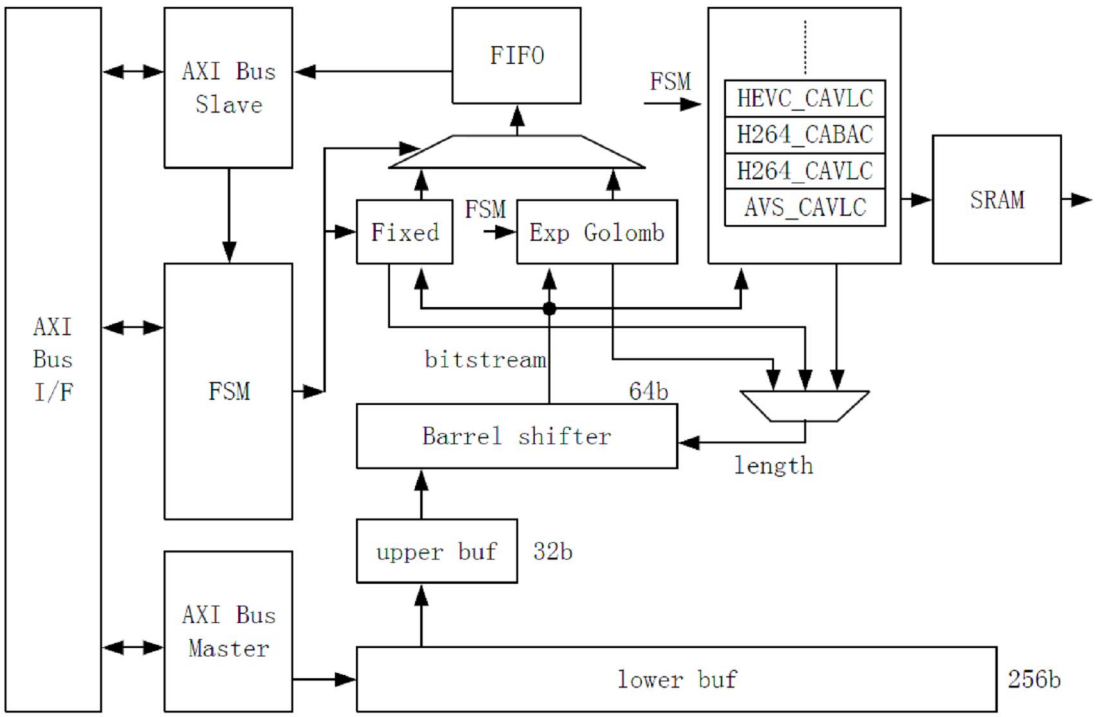

Fig. 2 Architecture diagram of HW/SW co-design approach

are parsed by hardware. This architecture is designed to support both AVS and H.264. It's easy to be extended to support more video standards. The disadvantage of this architecture is that syntax elements in layer 1 and 2 are parsed by software only and no hardware acceleration is used. It will pose a heavy burden on CPU.

In this paper, we proposed a new hardware/software co-design approach to parse the video bitstream. Compared with the work of Liu[11], the major improvement is that specific hardware acceleration for syntax elements in layer 1 and layer 2 are provided, which can greatly reduce the CPU loading for video bitstream parsing. Syntax elements in layer 3 are processed by hardware.

Due to its superior performance, AMBA AXI bus protocol is adopted in this work as the bus interface. The key feature "out-of-order transaction completion" and "multiple outstanding transaction" of the AXI protocol maximize the bus throughput.

Both AXI bus master and bus slave are presented in this design. AXI bus master is in charge of prefetching the video bitstream from external video bitstream buffer into on-chip buffer. Two on-chip buffer named "lower buffer" and "upper buffer" are provided. In previous work [8], these two buffers are designed to have the same size. In order to reduce the memory transaction and on-chip buffer size, we increase the size of lower buffer burst transaction. In our design, the data bus width is

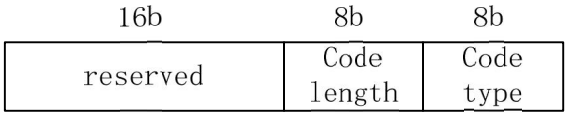

Fig 3. Bit definition of kick-off register

64-bit and burst size is 4. So the size of lower buffer is 256-bit. The upper buffer size is designed to be greater than the maximum length of a code word. The size of upper buffer is set as 32-bit.

After each code word is decoded and its code length is figured out, the barrel shifter will shift the bitstream in order to make sure that the bitstream in the buffer starts from the beginning of next code word. The size of barrel shifter is designed to be double than that of upper buffer. So the size of barrel shifter is 64-bit in this design.

AXI bus slave is in charge of accessing the configuration registers inside the video decoder. A special 32-bit register called "kick-off" register is provided. The definition of each bit in this register is depicted in Fig3. The last 8 bits is used to indicate the type of code word: fixed length code, exponential Golomb code or other type code word. Next 8 bits before last 8 bits are used as the word length of a fixed length code. Each time when a syntax element in layer 1 or layer 2 is to be parsed, this kick-off register will be written. A write transaction to this register will trigger the hardware to decode one code word. The parsing

result is stored in a FIFO in order to speed up the data throughput.

The syntax elements for prediction and residual information are parsed by hardware due to their high data rate. Each video standard has defined its unique coding method for these syntax elements. So hardware sharing is not feasible here. The result is stored in on-chip SRAM for future use.

4. Experimental results and comparison

This design has been implemented in fully synthesizable Verilog HDL. We use SMIC 0.13um standard library cell and Synopsys Design Compiler to get the timing and area information of this design. The target working frequency is set as 100MHz and the gate count is 11K. Hardware accelerators used to parse syntax elements in layer 3 are not included here due to the various specification of each video standard.

Compared with the hardware only implementation, this design shows greater flexibility. In previous work of Li[8] and Xu[9], hardware FSM is used to determine the parsing process of syntax elements in layer 1 and 2. Each syntax element needs a corresponding FSM state. For example, 107 FSM states are used in Xu[9] to support one single H.264 standard. If multiple video standards are to be supported, the number of FSM states will become extremely large. In our design, no FSM state is needed for the syntax elements in layer 1and 2, which reduce the most part of FSM state. Only 32 FSM states (5 global states and 27 states for H.264 syntax elements in layer 3) are needed for H.264 baseline profile.

Compared with the previous work of Liu[11], this design can reduce the CPU loading by hardware acceleration and speed up the parsing process of syntax elements in layer 1 or 2. This design only takes one cycle to decode an exponential Golomb code whose code length is no greater than 31-bit. Software solution will takes more than 6 cycles to decode a 31-bit exponential Golomb code.

5. Conclusion

We have proposed a hardware/software co-design approach to parse the video bitstream. Each syntax element in layer 1 or layer 2 of the video bitstream can be parsed by hardware accelerator while the overall parsing process of such syntax element can be controlled by the collaboration of software routine and the "kick-off" register. Due to the high bit rate, syntax elements in layer 3 are parsed by hardware only. This architecture has the flexibility to support multiple video standards while the high data throughput can also be guaranteed.

References

[1] Joint Video Team, "Advanced video coding for generic audiovisual services", ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC, Sept. 2005

[2] Soc. Motion Picture and Television Engs. (SMPTE), "Proposed SMPTE Standard for Television: VC-1 Compressed Video Bitstream Format and Decoding Process," SMPTE 421M, SMPTE, Aug. 2005

[3] Audio Video Coding Standard Workgroup of China (AVS), Advanced Coding of Audio and Video - Part 2: Video, December 2004.

[4] Moving Picture Experts Group, "Information technology-generic coding of moving pictures and associated audio information: Video," ISO/IEC 14496-2,1998.

[5] Moving Picture Experts Group, "Information technology-generic coding of moving pictures and associated audio information: Video," ITU-T H.262, ISO/IEC 13818-2,1994.

[6] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, "JCTVC-E603: WD3: Working Draft 3 of High-Efficiency Video Coding", Mar. 2011

[7] Wiegand T., Sullivan G.J., Bjontegaard G., Luthra, A., "Overview of the H.264/AVC video coding standard" ,IEEE Transactions on Circuits and Systems for Video Technology, p560-576,Jul. 2003

[8] Jui-Hua Li, Nam Ling, "Architecture and bus-arbitration schemes for MPEG-2 video decoder", IEEE Transactions on Circuits and Systems for Video Technology, p.727-736, Aug. 1999.

[9] Ke Xu, Chiu-Sing Choy, Cheong-Fat Chan, Kong-Pong Pun, "Power-efficient VLSI Implementation of Bitstream Parsing in H.264/AVC Decoder", IEEE International Symposium on Circuits and Systems, 2006

[10] Yung-Chi Chang, Chao-Chih Huang, Wei-Min Chao, and Liang-Gee Chen, "An efficient embedded bitstream parsing processor for MPEG-4 video decoding system", International Symposium on VLSI Technology, Systems, and Applications, 2003

[11] Liu Wei, Chen Yong-en, Wang Peng, "VLD Design for AVS and H.264 Dual Standards Video Decoder Video Decoder", 5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009