ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

MPEG-4 Video Bitstream Structure Analysis and Its Parsing Architecture Design

*Hao-Chieh Chang, Yung-Chi Chang, *Yuan-Bin Tsai, *Chih-Peng Fan and Liang-Gee Chen*

DSP/IC Design Lab, Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.
*N100, Computer and Communications Research Laboratories (CCL)
Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan, R.O.C.

ABSTRACT

In this paper, the hardware-oriented structure analysis and an efficient and flexible bitstream parser for MPEG-4 video are presented. The analysis of bitstream structure explores processing requirement and design constraint for bitstream-level processing. The proposed architecture is basically RAM-based that can be reconfigured for various applications. For high bitrate as about 40 Mbit/s, it needs only about 19 MIPS to parse the bitstream. The impact of the proposed architecture on MPEG-4 video is to enhance and extend the processing for bit domain translation and related real time applications.

1. INTRODUCTION

Compression of video data is essential for cost-effective transmission under limited communication channels and storage media. Hybrid video coding is a promising technique to minimize the required bitrate and has been widely adopted in many international standards, such as H.261 [2], H.263 [3], MPEG-1 [4], MPEG-2 [5], and MPEG-4. In real case, in addition to these coded data, several header information have to be inserted to transmit a complete bitstream. Thus, to correctly extract header information and coded data from various bitstream structures, a parsing processor is required for most prevailing multimedia terminals. Generally speaking, the implementations of bitstream parser can be divided into two classes. The most adopted implementation is the dedicated header decoder based on FSM [6][7]. The other one is RISC-core based architecture [8]. Dedicated architectures can achieve higher performance and more cost-effective with the penalty of lacking flexibility. On the other hand, RISC-core based architectures can provide better flexibility.

In previous designs for MPEG-2 or other video coding systems, a dedicated header decoder is usually adopted as front-end decoder. In MPEG-4, however, more complicated and flexible bitstream structure is required so as to provide more functionality. Such requirements lead the front-end decoder to be capable of flexible decoding. Although a RISC processor could provide such capability, it will lead to the performance degradation due to its inefficiency for bit-level processing. This motivates the research for flexible and efficient parser architecture design.

2. BITSTREAM STRUCTURE ANALYSIS ON MPEG-4 VIDEO

A bitstream is composed of several codewords, which represent some information or symbols while delivering multimedia data. Some codewords are fixed-length, while others are generated by entropy coding that will generate variable-length codes. Therefore, it is necessary to perform VLC table-lookup while encoding or decoding a bitstream. Additionally, to access and handle a bitstream, several functions are required to perform bit-level processing as listed in the following:

1. FillBuffer: To read a piece of the bitstream from the bitstream file to the bitstream buffer.

2. ShowBits: To see the next several bits after the current position of the bitstream pointer without advancing the pointer. It returns several bits of the bitstream right aligned.

3. GetBits: To read the next several bits of the bitstream and advances the bitstream read pointer.

4. FlushBits: To advance the bitstream read pointer for several bits and call the function "FillBuffer" to reload the bitstream buffer whenever necessary.

5. ByteAlign: To advance the bitstream read pointer until the number of bits left in the buffer is a multiple of 8.

6. ShowBitsByteAlign: To see the next bytealigned several bits in the bitstream without advancing the bitstream read pointer.

7. PutBits: To append a stream of bits to the bitstream.

Among them, both the functions GetBits and ByteAlign are totally composed of the functions ShowBits and FlushBits. These functions all can be accomplished by bit-wise shift and or.

In bitstream-level, the codewords in a bitstream are generated and extracted sequentially. In the process of encoding or decoding a bitstream, the codeword to be inserted or to be extracted next is unknown until current or previous symbol, or current position of the bitstream pointer is known. Due to this characteristic, some decisions must be made according to current or previous symbol, or current position of the bitstream pointer so as to perform encoding or decoding. So, condition checking is a necessary task to decide next codeword to be inserted or to be extracted and its bit-length.

From the above two paragraphs, the basic operations required to carry out bitstream processing are VLC table-lookup to perform VLD, bit-wise shift and or to access and handle bitstream, and comparisons to perform condition checking. However, there are some problems while processing bitstream because of the characteristic of bitstream. The bit-length of the coming codeword is unknown in most cases, especially for VLC codes. The number of bits processed at the same time is usually limited to 1 or 2 bits.

0-7803-5482-6/99/$10.00 ©2000 IEEE

Therefore, it is difficult for a bitstream-processing module to achieve high throughput.

The bitstream structure is the description about the relationship among codewords and how to concatenate separate codewords to form a complete bitstream. In MPEG-2 video, its structure is hierarchy and top-down with sequence, group of picture, picture, slice, macroblock, and block layer. For MPEG-4 video, a video scene consists of one or several visual objects, which contain one or more video object layers. One instance of a video object layer at a given time is considered as a video object plane. In a video object plane are some video packets, which are composed of data of several macroblocks. In addition to motion and texture data as in MPEG-2 video, shape information of a macroblock is also provided. The order how these data are concatenated in a video packet can be in the order of macroblock or partitioning different types of data, according to the requirement of error resilience.

The bitstream syntax is used to describe the bitstream structure, including codeword descriptions, and some decision-making functions. From the syntax defined in MPEG-4 video standard, the parsing instruction set is defined in order to accomplish bitstream parsing by executing the parsing instructions sequentially. The parsing instructions, each one corresponding to one group, and their parameters are shown in Table 2.

3. PROPOSED ARCHITECTURE AND DESIGN EXAMPLE

Based on the analysis of bitstream structure, only seven types of parsing instructions are sufficient to decode MPEG-4 video bitstream as shown in Table 1. The instruction FLD and VLD are used to extract codewords of different code length in the bitstream. The instruction FOR and FNC determine which parsing instruction should be fetched at next cycle. FNC changes the layer to be processed in the bitstream structure such that all levels of bitstream can be parsed. Both the instruction BRP and BRN are used to decide the execution order of parsing instructions according to the results of comparison operation. The difference between BRN and BRP is that BRP reads data from data memory to perform comparison while BRN doesn't. The instruction CMP performs some basic operations such as addition, subtraction, and shift on previously decoded data.

In order to accomplish these tasks, the proposed core architecture is basically composed of three major units: functional unit (FU), memory management unit (MMU), and instruction decoder (INSTDEC). The FU performs codeword decoding and arithmetic and logic operations required by BRP, BNP, and CMP. MMU comprises several memory modules for storing the parsing instructions and decoded data. Additionally, an address generator (AG) is also included in MMU to generate addresses and control signals for memory modules. INSTDEC decode the parsing instruction and generate corresponding data for FU or AG. The proposed bitstream parsing processor architecture is shown as Figure 1.

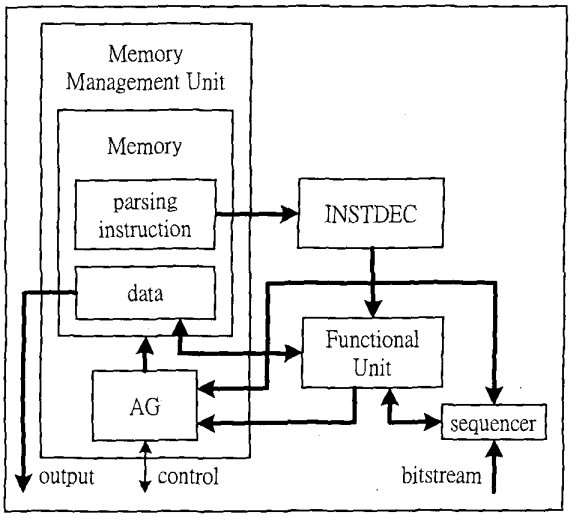

Figure 1. Architecture for MPEG-4 video bitstream parsing.

The decoding flow is described as follows. Bitstream data is fed into a sequencer. To support bit-level processing, instruction FLD is used to extract the output data of sequencer according to data length denoted in parsing instruction. To support VLC decoding, the instruction VLD is used to perform VLC table-lookup. The decoded symbol of FLD or VLD is written into data memory whose address is generated by address generator. The decoded data, which are required by motion, texture or shape decoder in latter stage, can be the outputs of the proposed parsing processor by controlling the AG. To support condition checking, the instruction BRN or BRP is used to check the branch conditions denoted in the instruction field and the comparison result is sent to AG to determine next parsing sequence. When the instruction FOR or FNC is used, AG is controlled to generate correct address for fetching next parsing instruction. Therefore, parsing bitstream in all levels is supported. The instruction CMP performs operation on previously decoded symbol in FU and restores it in data memory.

The superiority of the proposed architecture is described. For an example adopted from MPEG-4 video standard:

```
if (interlaced && field_prediction)
    motion_vector("forward")
```

The first line performs two condition-checking tasks based on two previously decoded symbols. One of the tasks is to compare the symbol "interlaced" with the immediate value "1", and the other one is to compare the symbol "field_prediction" with the immediate value "1". Afterwards, the final result will be true only when both conditions are met. The corresponding first parsing instruction is:

```
BRP ("if", 2, 1, "interlaced" "field_prediction",
"1" "1", "immediate", "=" "=" "&", "No")
```

The execution flow and status of the blocks used in this example are described in Table 3. As the instruction is decoded, the previously decoded data to be compared will be read out from data memory at the first and second cycles. The two branch conditions

are checked at the second and third clock cycles after the corresponding data is read from data memory. In the fourth clock cycle, the final result will be calculated and sent to AG to generate next address of parsing instruction. No deep pipelines are embedded in the proposed architecture. Consequently, there is no memory access latency, which makes it possible to access data and perform comparison simultaneously. Additionally, an extra parsing instruction to indicate ending of a branch or a loop is not required. As long as the address generator detects the ending of a branch or a loop, the correct address will be generated to read next parsing instruction.

4. PERFORMANCE EVALUATION

In the proposed architecture, one clock cycle is required to accomplish variable-length decoding by table-lookup, while a general-purpose RISC spends much more clock cycles [8]. Besides, it can accomplish two branch conditions checking in one branch instruction. While most data are decoded in macroblock-level, this feature can increase throughput of the parsing processor.

The clock cycles required in different architecture are calculated as compared results. The instructions adopted in a general RISC core are referenced in [9]. The comparison of clock cycles is shown in Table 4.

Table 1 summarizes the hit count percentage occupied by each kind of parsing instruction, except for CMP. The results are based on profiling the program on a Sun workstation to determine counts of function calling and variable- and fixed-length decoding for 3 test sequences. In addition, the counts of condition checking are estimated according to the source code and the bitstream structure defined in the standard.

	bream	children	weather	Average
FLD	33.2064	37.8091	31.9215	34.3123
VLD	10.5672	11.0891	10.8586	10.8383
BRP	24.5169	19.2668	23.6464	22.4767
BRN	27.3439	28.3956	29.1476	28.2957
FNC	3.4862	2.905	3.5294	3.3069
FOR	0.8794	0.5344	0.8965	0.7701

Table 1. Hit count percentage occupied by each kind of parsing instruction.

Thus, it is estimated that the operation "Fixed-length decode" will occupy 34%, "Variable-length decode" will occupy 11%, "Branch" will occupy 50% of complete bitstream parsing, and the rest 4% is left for other kinds of operations. In MPEG-4 video Main Profile Level 4, the maximum bitrate can reach 38.4 Mbit/s [1]. Under such circumstance, the resulting required MIPS of each architecture is shown in Figure 2. It's clear that a general-purpose RISC core would spend about 160 MIPS, and the architecture in [8] would spend about 40 MIPS. However, the proposed architecture only takes about 19 MIPS. Obviously, the proposed architecture achieves better performance than RISC-based architecture.

5. FUTURE WORK

The analysis of bitstream structure of MPEG-4 video explores processing requirement and design constraint for bit-level processing. The analysis is suitable for previously developed standards, such as MPEG-1 [4] and MPEG-2 [5], whose bitstream syntax is a subset of MPEG-4's. The proposed architecture is basically memory-based that can be reconfigured for various applications. Although it is for MPEG-4 video application, it can be extended to a universal bitstream parser able to parse bitstream of different standards based on the analysis.

6. REFERENCES

[1] ISO/IEC JTC1/SC29/WG11. *N2502a, Generic Coding of Audio-Visual Objects: Visual 14496-2, Final Draft of International Standard*, Atlantic City, Dec. 1998.

[2] CCITT Study Group XV, TD35. "Draft review of recommendation H.261 video codec for audiovisual services at p×64 kbits/s," *Image Communication*, pp.221-239, August 1990.

[3] "Video coding for narrow telecommunication channels at < 64 kbits/s," *Draft ITU-T Recommendation H.263*, July 1995.

[4] D. L. Gall. "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, Vol. 34, No. 4, pp.46-58, April 1991.

[5] ISO/IEC/JTC1/SC29/WG11 *Draft CD 13818-2 Recommendation H.262 Committee Draft.*

[6] J. H. Li, N. Ling, "Architecture and bus-arbitration schemes for MPEG-2 video decoder," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 5, pp.727-736, August 1999.

[7] T. Onoye, T. Masaki, Y. Morimoto, Y. Sato, I. Shirakawa, "HDTV level MPEG2 video decoder VLSI," pp.727-736, *TENCON'95*.

[8] M. Berekovic, G. Meyer, Y. Guo, P. Pirsch, "A multimedia RISC core for efficient bitstream parsing and VLD," *SPIE'98*.

[9] J. L. Hennessy, D. A. Patterson, *Computer Architecture: A Quantitative Approach*, second edition, Morgan Kaufmann Publishers, Inc., 1996.

Group No.	Parsing instruction	Parameter								
1	FLD	Data length		Next/fld		Immediate/data		Data name		
2	VLD	Data name								
3	FOR	Times to be executed		Immediate/data		Loop length				
4	BRP	If/while	Cond number	Branch length	Data name	Data compared	Immediate/data	Equality	Next	
5	BRN	If/while	Cond number	Branch length	Next bit number	Data compared	Immediate/data	Equality	Next	Bytealign
6	FNC	Address of function		Function length		Immediate/data				
7	CMP	Data name		Operation		Operand		Immediate/data		

Table 2. Parsing instruction set

Clock cycle	FU	AG	Data memory
1		Addr(interlaced)	
2	Compare "interlaced" with 1	Addr(field_prediction)	Read(interlaced)
3	Compare " field_prediction " with 1		Read(field_prediction)
4	Calculate (interlaced & field_prediction)	Addr(next instruction)	

Table 3. Execution flow and status of the blocks used in the example

	DLX [9]	M. Berekovic [8]	Proposed	Speedup with DLX
Fixed-length decode[8]	7/11	1/2	1/2	3.5/11
Variable-length decode	32	5/1	1	32
For	2	2	1	2
Branch (previous data)	3/4	3/4	2/3	1.3/1.5
Branch (next data)	9/24	3/5	1/2	2/13
Function calling	2	2	1	2
Computing	2/3	2/3	2/3	1

Table 4. Performance comparison in clock cycles

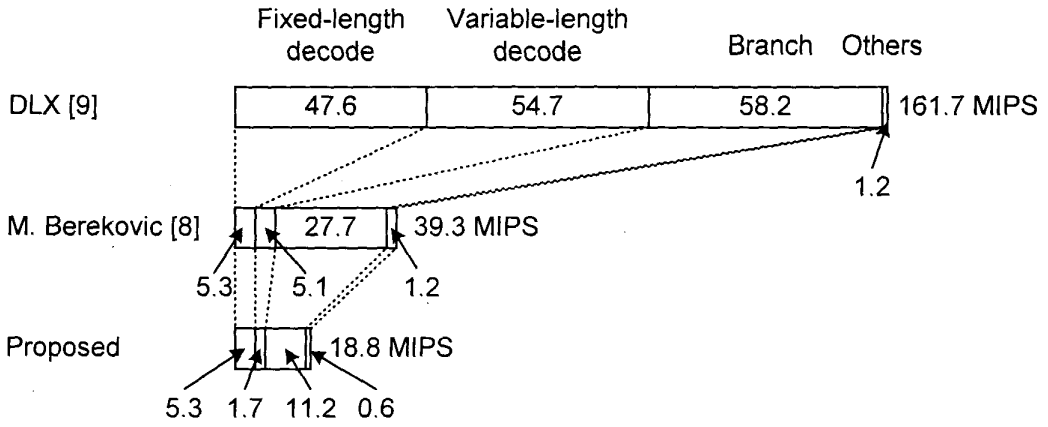

Figure 2. Performance comparison in MIPS