Design and Implementation of a Bitstream Parsing Coprocessor for MPEG-4 Video System-on-chip Solution

Yung-Chi Chang, Hao-Chieh Chang, and Liang-Gee Chen
DSP/IC Design Lab, Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

ABSTRACT

In this paper, the hardware-oriented bitstream structure analysis and an efficient and flexible bitstream parsing processor are presented. The analysis of MPEG-4 video bitstream structure based on RISC model explores requirement and design constraint for bitstream-level processing. It shows that conventional RISC is not efficient enough for bitstream parsing. An efficient instruction set optimized for bitstream processing is designed and the hardware architecture can be reconfigured for various applications. Compared with 160 MOPS required by a RISC, the proposed architecture needs only about 27 MOPS to parse an MPEG-4 video bitstream at high bit-rate as about 40 Mbit/s, which is about 6 times speedup. The impact of the proposed architecture on video applications is to enhance and extend the processing for bit domain translation and related real time applications.

INTRODUCTION

Nowadays most video data are represented and stored in digital form. Digital form data possess the advantages of further processing easily. Moreover, a lot of data compression technique can be applied for digital data. Data compression is essential because of limited communication channel and finite storage media space. After compression, the original digital data is transformed to compressed format called bitstream. A bitstream is composed of hundreds to millions of single bit whose value can be one or zero. It contains necessary information, including coded data and header information, for decoder to transform it into its original uncompressed format, with a little distortion indistinguishable to human. Parsing is to extract header information and coded data correctly according to the compression rules. A parser can be regarded as a pre-processing unit in a video decoding system, as shown in Figure 1 [1].

Generally speaking, the implementations of bitstream parser can be divided into two classes. One is the dedicated decoder based on FSM [3][4]. It can achieve higher performance and more cost-effective with the penalty of lacking flexibility. A programmable architecture is the other solution. The versatile VLD chip design proposed by Yang [5] can parse bitstreams of H.261 and JPEG. Additionally, the extensions of a RISC core for bitstream parsing and VLD proposed by Berekovic [6] can enhance the performance of bitstream processing for a conventional RISC.

In previous designs for MPEG-2 [2] or other video coding systems, a dedicated decoder is usually adopted as a front-end decoder. In newer system, like 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.

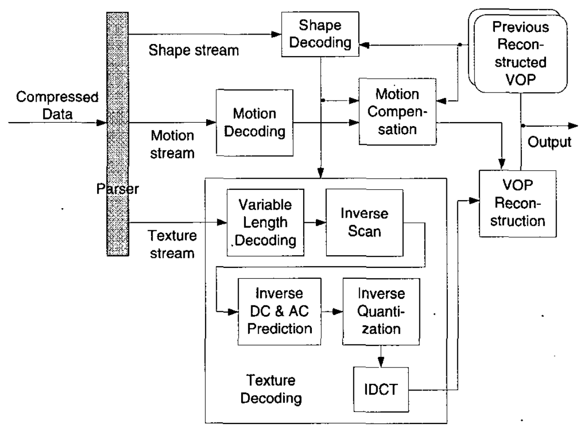

FIGURE 1. MPEG-4 Video Decoding System

BITSTREAM STRUCTURE ANALYSIS

A bitstream is composed of several codewords, which represent some information or symbols while delivering multimedia data. The code lengths of codewords can be fixed or variable. In this section, how they form a bitstream and are handled will be presented.

Bit-level Processing

To access and handle a bitstream, several functions are required to perform bit-level processing. To see a piece of bitstream, the piece should be right aligned. When the piece is read out and next piece of bitstream is required, the bitstream pointer should be advanced. Sometimes it is required to advance the bitstream pointer until the number of bits left in the buffer is a multiple of 8. To append several bits to the bitstream, the bits to be appended should be left or right shifted to the correct location. In brief, these functions all can be accomplished by bit-wise SHIFT and OR.

0-7803-6412-0/01/$10.00©2001 IEEE.

Bitstream-level Processing

One or several bits form a codeword, which is the information hidden in the bitstream. The bitstream structure is the description about the relationship among codewords and how to concatenate separate codewords to form a complete bitstream. From the bitstream structure, a real bitstream can be generated by concatenating several codewords sequentially. Therefore, in the process of decoding a bitstream, the codeword to be extracted next is unknown until current or previous symbol, or current position of the bitstream pointer is known. Due to this characteristic, the bitstream-level processing becomes more complex because some decisions must be made according to current or previous symbol, or current position of the bitstream pointer such that bitstream parsing can be performed smoothly.

The bitstream syntax is used to describe the bitstream structure, including codeword descriptions, layer structure, and some decision-making functions. Basically, the bitstream structure defined in MPEG-4 video coding standards [1] is hierarchy. A video scene contains one or several visual objects, which are composed of 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 include 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 bitstream syntax used in MPEG-4 video bitstream structure are listed in the following:

Fixed-length code: A codeword whose length is known exists in the bitstream.

Variable-length code: The codeword is variable-length coded. So, Its code length and value can be acquired by VLC table lookup.

Layer transformation: This syntax exists when the following part of a bitstream represents the information of another layer.

Loop: A series of codeword appears in the bitstream for several times.

Decision-making: The following bitstream structure must be decided. The branch condition may depend on the value of previously extracted codeword or the coming codeword.

Code calculation: The value of previously extracted codeword will be modified for future condition checking.

From the above paragraphs, it is clear that VLC table-lookup, decision-making, and comparison are the key operations required to accomplish bitstream-level processing.

COMPLEXITY ANALYSIS

Analysis Approach

As the characteristics of bitstream-level processing have been explored, the computational complexity of MPEG-4 video bitstream parsing is analyzed by applying a RISC-based computation model. The computational complexity is defined as the required clock cycle count for bitstream parsing. Since the necessary bitstream syntax has been found out, each syntax can be regarded as a task performed while parsing a bitstream. Thus, the total complexity of bitstream parsing can be acquired by calculating the product of each task count and clock cycle count each task needs. It is shown in the following equation:

$$\sum_{i=1}^{7} (cycle\ \ count\ )_i \times (no\ \ of\ \ task\ \ count\ )_i$$

To find out each task count, MPEG-4 video software decoder is profiled on workstations so as to obtain number of times some functions called. Besides, statistical analysis is used for some tasks. To find out clock cycle count of each task, the instruction set of a general RISC core [7] is analyzed.

Analysis Result

The analysis result is shown in Table 1. The data are based on the test sequence bream in size 720x480 with three objects, 300 frames. The task item " Decision-making (P)" represents that the branch condition depends on the value of previously extracted codeword, while the item " Decision-making (N)" stands for that the branch condition depends on the value of previously extracted codeword. The task "Code calculation" is ignored here due to its low appearance frequency. From the table, half of MOPS is spent on decoding fixed-length and variable-length codes, while other half is spent on Branch. The total MOPS required by a general RISC core to perform bitstream parsing is about 160 MOPS.

TABLE 1. Complexity Analysis Result

Task	Cycle Count	Task Count %	MOPS	%
Fixed-length decode [6]	8/13	33.21	49.01	30.89
Variable-length decode	19/26	10.57	33.42	21.07
Decision-making (P)	6/9	24.52	25.85	16.29
Decision-making (N)	11/14	27.34	48.05	30.28
Layer Transformation	4	3.48	1.96	1.24
Loop	3	0.88	0.37	0.23
Total	---	100	158.67	100

ARCHITECTURE DESIGN

Proposed Instruction Set

To enhance bitstream processing, it is necessary to propose a new instruction set to parse a bitstream more efficiently. From the analysis result, the parsing efficiency for a RISC core is degraded for extracting fixed-length and variable-length codewords and checking branch conditions. The proposed instruction set is listed in the following:

FLD: This instruction is used for fixed-length decoding. Besides, advancing the bitstream read pointer until the next bytealign position is supported.

VLD: This instruction is used for variable-length decoding.

FNC: This instruction is used for layer transformation.

CAL: This instruction is used for code calculation. The operation to be performed can be arithmetic or logical operations.

FOR: This instruction is used for loop.

BRANCH: This instruction can support up to two branch conditions checking.

Proposed Architecture

From the proposed instruction set, the architecture is composed of four major units: functional unit (FU), address generation unit (AG), instruction decoder (INSTDEC), and a sequencer. The proposed parsing processor architecture is shown as Figure 2.

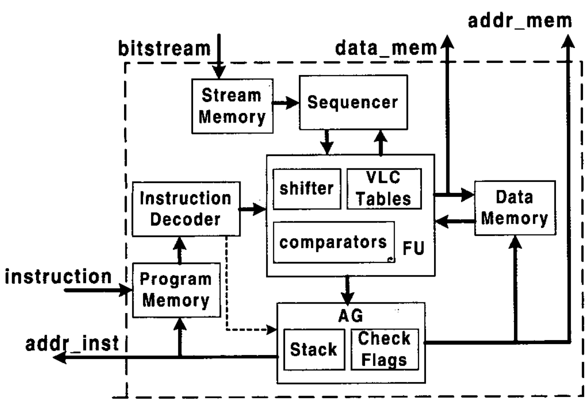

FIGURE 2. Parsing Processor Architecture

FU performs bit-level shift for fixed-length decoding, arithmetic operation such as addition and subtraction for code calculation, comparison and bit-wise operation for branch. Thus, a shifter and VLC tables are included in it. Besides, 2 comparators to perform branch condition checking are embedded here.

AG generates memory addresses and control signals to indicate read or write mode for program memory and data memory. It is regarded as the controller of the whole paring processor. To support all-layer parsing, stack implementation is included here for three circumstances: layer transformation, looping, and branch instruction with conditions met.

INSTDEC decodes the executing parsing instruction in order to generate necessary data and control signals for sequencer, FU, and AG. The sequencer shifts the incoming bitstream to the position where the segment to be parsed in fed into FU.

The parsing flow is described as follows. At first, bitstream and parsing instructions are loaded into stream memory and program memory, separately. Bitstream is fed into a sequencer. FLD extracts codeword from output stream of sequencer according to data length denoted in parsing instruction. Instruction VLD performs VLC table-lookup. The decoded codeword of FLD or VLD is written into data memory whose address is generated by AG. 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. BRANCH checks the branch conditions denoted in the instruction field and the comparison result is sent to AG to determine next parsing sequence. When FOR or FNC is executed, AG is controlled to generate correct address for fetching next parsing instruction. CAL performs operation on previously decoded symbol in FU and restores it in data memory.

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 [7]. Besides, it can accomplish two branch conditions checking in one branch instruction. The performance comparison in clock cycle count of each task is shown in Table 2. The proposed instruction set outperforms RISC's in code-extraction.

TABLE 2. Performance Comparison

Task	32-bit RISC [10]	[6]	Proposed	Speedup with RISC
Fixed-length decode [6]	8/13	1	1	8/13
Variable-length decode	19/26	4	1	19/26
Decision-making (P)	6/9	5/8	2/4	1.5/2.25
Decision-making (N)	11/14	6/9	2/3	3.67/5.5
Layer Transformation	4	4	1	4
Loop	3	2	1	3
Code Calculation	2/3	2/3	2/3	1

In MPEG-4 video Main Profile Level 4, the maximum bitrate can reach 38.4 Mbit/s and the MB rate is 489600 MB/s

[1]. Under such circumstance, the resulting required MOPS of each architecture is shown in Figure 3. It's clear that a general-purpose RISC core would spend about 160 MOPS, and the architecture in [6] would spend about 64 MOPS. However, the proposed architecture only takes about 27 MOPS. About 6 times speedup with 32-bit RISC is achieved.

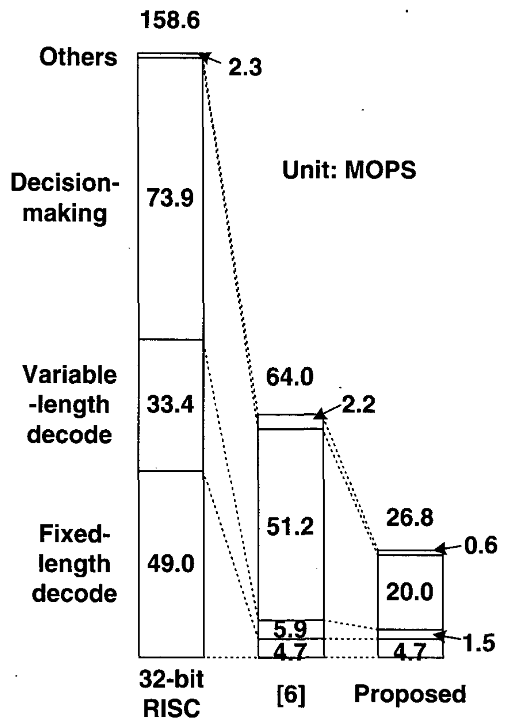

FIGURE 3. Performance Comparison in MOPS

IMPLEMENTATION

The VLC tables required for VLC decoding are embedded in the system. Software to transform VLC tables defined in standards to HDL-code has been developed. About 24K gates are used for the architecture except for the memory. Among them, less than 7K gates are for VLC tables. The summary of chip features is listed in Table 3. The chip layout is shown in Figure 4.

TABLE 3. Chip Summary

Technology	TSMC 0.35um CMOS 1P4M
Chip Area	3.02 x 3.00 mm^2
Die Area	2.23 x 2.21 mm^2
Gate Count	24,517 (memory excluded)
Memory	9088 bits
Transistor Count	227,224 (memory included)
Speed	40 MHz
Power	250 mW @ 40 MHz, 3.3 V
Package	100 CQFP

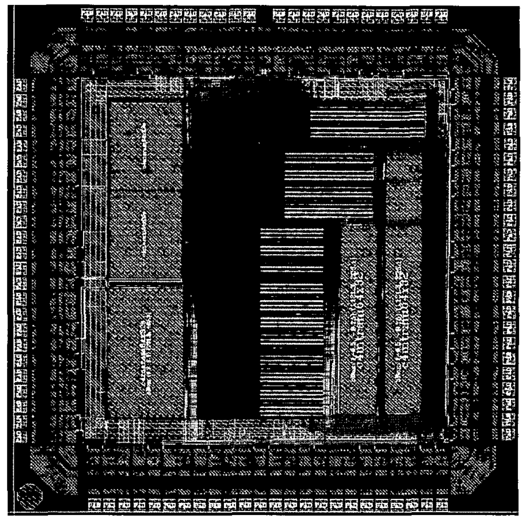

FIGURE 4. Chip Layout

CONCLUSION

The analysis of characteristics of bitstream-level processing and an MPEG-4 video bitstream parsing processor architecture is presented in this paper. The processing power of bitstream processing is shown from the analysis. An instruction set optimized for bitstream parsing is proposed according to the analysis result. The proposed architecture is based on a new proposed instruction set for bitstream parsing. The evaluation result shows the required MOPS of the proposed architecture is only 27 MOPS, which is much lower than that of a general RISC core, while the programmability and flexibility are supported.

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] ISO/IEC/JTC1/SC29/WG11 *Draft CD 13818-2 Recommendation H.262 Committee Draft.*
[3] 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.
[4] T. Onoye, T. Masaki, Y. Morimoto, Y. Sato, I. Shirakawa, "HDTV level MPEG2 video decoder VLSI," pp.727-736, *TENCON'95*.
[5] K.-M. Yang, F. Fujiwara, T. Sakaguchi, A. Shimazu, "VLSI architecture design of a versatile variable length decoding chip for real-time video codecs," *IEEE Region 10 Conference on Computer and Communication Systems*, pp.551-554, September 1990.
[6] M. Berekovic, G. Meyer, Y. Guo, P. Pirsch, "A multimedia RISC core for efficient bitstream parsing and VLD," *SPIE'98*.
[7] J. L. Hennessy, D. A. Patterson, *Computer Architecture: A Quantitative Approach*, second edition, Morgan Kaufmann Publishers, Inc., 1996.