AN EFFICIENT EMBEDDED BITSTREAM PARSING PROCESSOR FOR MPEG-4 VIDEO DECODING SYSTEM

Yung-Chi Chang, Chao-Chih Huang, Wei-Min Chao, and Liang-Gee Chen

DSP/IC Design Lab.
Department of Electrical Engineering and Graduate Institute of Electronics Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

ABSTRACT

In this paper, the bitstream parsing analysis and an efficient and flexible bitstream parsing processor are presented. The bitstream parsing analysis explores the critical part in bitstream parsing. Based on the result, the novel approaches to parse data partitioned bitstreams are presented. An efficient instruction set optimized for bitstream processing, especially for DCT coefficient decoding, is designed and the processor architecture can be programmed for various video standards. It has been integrated into an MPEG-4 video decoding system successfully and can achieve real time bitstream decoding with bitstream coded under 4CIF frame size with 30 fps, 8Mbps, which is the specification of MPEG-4 Advanced Simple Profile Level 5.

1. INTRODUCTION

As video coding standards development process keeps going on, no matter for MPEG or H.26x series, more and more coding tools are added to provide more functionalities and better compression performance in a general video coding structure. Thus, its coded data format must be changed. Moreover, commercial video products are now beginning to support several coding standards simultaneously [1]. So, a versatile bitsteam parser for a video coding system implementation will be the trend. A hardwired parser is not a good choice to satisfy the rapidly transition of video coding standards. On the other hand, adaptation of an embedded processor will make the design fast time-to-market. By re-programming the firmware, one can change the specification to match another new coding standard in short time. Moreover, firmware upgrade can be accomplished by replacing the layout of the ROM in fabrication, even no changes if flash is used. Up to now, implementations of video decoder usually embed microprocessors on-chip to be the parsing unit [1][2][3].

Processing of the bitstream often requires bit-level operations such as bit extraction and variable-length decoding functions. Processors designed with 16-bit or 32-bit operations would spend many cycles for a single bit operation. Therefore, it is not efficient to use a general processor for bitstream parsing task. The introduction of processors needs more analysis and optimizations. For MPEG-4 video decoding system, both [4] and [5] propose such a solution. These previous designs emphasize the importance of VLD/FLD operations, and propose enhanced datapath for VLD/FLD.

In this paper, we propose an embedded bitstream processor for MPEG-4 video decoder. Based on our previous work [4] and codeword type distribution analysis, efficient parsing algorithms supporting data partitioned bitstreams are proposed and realized with the proposed bitstream processor. The proposed design can achieve MPEG-4 Advanced Simple Profile (excluding GMC and QMC) Level 5 (720x576, 30fps) real time decoding.

The paper is organized as follows. The analysis of MPEG-4 video bitstream structure is shown in Sec. 2. Based on the analysis, efficient parsing algorithms supporting data partitioned bitstream parsing are presented. The proposed architecture is shown in Sec. 3. The implementation result is presented in Sec. 4 . A conclusion is given in Sec. 5.

2. BITSTREAM PARSING ANALYSIS AND PROPOSED ALGORITHMS

A detailed analysis for MPEG-4 video bitstream structure has been discussed in [4], which shows that there are six classes of operations that occur very often in the bitstream parsing operations. Based on its result, we focus on finding out the most critical part during parsing and speedup it. In addition, to support the error resilience decoding function, a parsing approach for data partitioned bitstreams is also illustrated.

2.1. Codeword Type Distribution

A software model for bitstream parsing is applied for MPEG-4 video decoding. The codeword distribution among several bitstreams is acquired during parsing. Since the computation loading of parsing is proportional to the bit-rate of encoded bitstream, we only perform the analysis on high bit-rate bitstreams. The most part is the DCT coefficient codewrods, which occupies about 70% of the bitstream. The DCT coefficients not only are mass but also occur successively in the bitstream. Besides, if the DCT coefficients cannot be decoded for the MPEG-4 video decoder on time, the decoding system have to be paused and the overall decoding performance is decreased. So, the operations for DCT coefficient decoding have to be optimized.

2.2. DCT Coefficients Decoding Analysis

The operations for the DCT coefficient decoding consist of four parts:

1. *Codeword Identification*: The beginning part of the bitstream is shown and VLC table lookup is performed in the corresponding codebook. If it is valid, the symbol address of the codeword should be generated.

2. *Symbol Lookup*: After the address is generated, the parser reads the symbol entry of the codeword. In DCT coefficient decoding, the looked-up symbol is the (run, level, last) pair.

0-7803-7765-6/03/$17.00

Table 1. Cycle Analysis for I-VOP DCT Coefficients Decoding.

Sequence	General	Enhanced	Percentage
Foreman	210,908	141,364	67.03%
News	122,734	99,394	80.98%
Weather	379,659	210,993	55.57%

Table 2. Cycle Analysis for P-VOP DCT Coefficients Decoding.

Sequence	General	Enhanced	Percentage
Foreman	99,905	80,246	80.32%
News	99,020	54,804	55.35%
Weather	43,611	35,165	80.63%

3. *Symbol Output/Bitstream Update*: The symbol is output whenever the symbol is obtained. Meanwhile, the bitstream is updated by discarding the last decoded codeword.

4. *Branch decision*: The output symbol is checked to see if the DCT coefficient decoding process should be continued. Whether the symbol is a legal symbol, an escape code, or the last coefficient within a block should be checked.

2.3. Proposed DCT Coefficients Parsing Approach

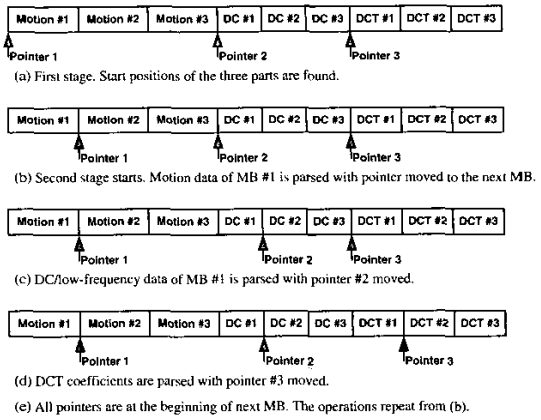
The conventional processor-based implementations [4][5] for DCT coefficients parsing focus on the first to third parts, but ignore the fourth part presented in Sec. 2.2. If we take it into account, the required cycle for VLD in [4] is 4 rather than 1, and that in [5] is about 6 to 10 rather than 4. So, if we can merge this essential branch decision operation with the former 3 parts, the decoding performance can be improved.

A processor emulator with similar instruction architecture to the RISC but slight modification is setup. Then, a firmware program for bitstream parsing is written for simulation. In one case, it's assumed that VLD operation can be finished in single instruction, but the branch decision is required after each DCT decoding. In the other case, the essential branch oepration is merged with the VLD instruction such that the codeword decoding and branch condition checking can be accomplished within one cycle. The comparison on required cycle to parse on I- and P-VOP is shown in Table. 1 and Table. 2. It is shown that the improvement with the new merged instruction is between 20% and 50% of processing cycles. Thus, it's desirable to merge the essential branch decision operation with the VLD operation for a processor-based parser.

2.4. Proposed Data Partitioned Parsing Approach

A P-VOP video packet is composed of three parts: *Motion part*, which keeps the motion data of all MBs, *DC and low-frequency DCT related data*, which contains DC values and the AC prediction flag, and *DCT coefficients*. In each part, the data is ordered in one MB after another. Since data within one MB is divided into three different locations, either a large buffer to store the previously decoded data or parsing several times to obtain the necessary codeword in the video packet is required. However, the former appraoch costs too high, while the latter one is inefficient.

We propose an cost-effective algorithm to parse the data partitioned bitstream efficiently. It's shown in Fig. 1. The parsing is composed of two stages. In the first stage, the whole video packet is only watched and stored to find the starting positions of the three parts described above. After the starting positions of the three parts are found, the second stage parsing starts. At first, the motion data of the first MB are parsed. Then the DC/low-frequency data of the first MB, followed by the DCT coefficients of the first MB, are decoded. The data in the three parts are decoded alternatively until all MBs in the video packet are parsed. With the proposed appraoch, the required buffer size can be reduced greatly. While decoding one frame with CIF size, only one packet buffer with maximum packet size, which is 8K bits at MPEG-4 Advanced Simple Profile Level 5, and one side buffer with size about 700 bits are required. Compared with conventional implementation, which may needs 43K bits, the proposed algorithm is more cost-effective.



(a) First stage. Start positions of the three parts are found.

(b) Second stage starts. Motion data of MB #1 is parsed with pointer moved to the next MB.

(c) DC/low-frequency data of MB #1 is parsed with pointer #2 moved.

(d) DCT coefficients are parsed with pointer #3 moved.

(e) All pointers are at the beginning of next MB. The operations repeat from (b).

Fig. 1. Proposed Data Partitioned Bitstreams Decoding Approach.

Table 3. Cycle Overhead for Proposed Data-partitioned Parsing Algorithm from Software Simulation.

Sequence	Combined	Data Partitioned	Overhead	Overhead(%)
Bream	100,528	111,399	10,871	10.81%
News	56,928	59,443	2,515	4.42%
Weather	93,344	100,460	7,116	7.62%

The cycle overhead for the proposed algorithm is shown in Table. 3. We encode the sequence in either data partitioned mode or non-data-partitioned mode, and use the emulator to parse it to count the required cycles. It is shown that the overhead is tolerable with respect to the total required cycle.

3. ARCHITECTURE DESIGN

3.1. Features

From the analysis results in previous section, the proposed bitstream processor has the following features:

Single-cycle DCT coefficient decoding: As the analysis in the previous section, the most critical section of our design, the DCT coefficient decoding, involves two memory operations in one decoding cycle. One is the read operation for symbol lookup, and the

other is the write operation for data output. If we want the coefficient decoding to be completed in one cycle, there should be two buses for one read and one write operation simultaneously on the memory. Thus, we try to include a two-port RAM model into our design for the two simultaneous memory operations.

Conditional executions: The analysis about the parsing operations in [4] shows the branch instruction occupies a large proportion. The occurrence of the branch is so often, but the target for the jump usually consists of a single operation such a VLD or a FLD. In order to eliminate the branch overhead, we introduce the conditional executions in modern DSP and micro-controllers [6]. By conditioning the execution with a flag (a one-bit register), every instruction can be controlled more freely than the branch architecture.

Bit-array operations: One-bit data is often found in the bitstream. General operations such as comparisons use one-bit signals, too. One-bit memory access operations, such as 1-bit load and 1-bit store, are provided.

VLC table programmable: To provide more flexibility and for supporting other video coding standards in the future, the VLC tables is programmable.

3.2. Proposed Instruction Set

We design the instruction set to provide the above features. It can be divided into five categories according to its functionalities.

Arithmetic instructions: The Boolean logic operations, 16-bit addition and subtraction are available. One 8-bit multiplier is also included. Special functions such as absolute value, conversion from sign-magnitude to 2's complement and bit-field extraction are also available to use.

Bitstream operation instructions: There is a set of enhanced bitstream operations, including fixed-length decoding and variable-length decoding. To optimize the DCT coefficient decoding, as mentioned in the previous section, one special variable-length decoding instruction called 'REP.VLDS' is used for repeatedly decoding. With the help of conditional execution, it will execute DCT coefficient parsing repeatedly automatically until parsing for a series of DCT coefficient codeword is finished.

Branch operations: Most of the RISC branch instructions are replaced by the conditional execution. The branch instruction set is reduced to jump with or without linking the return address to registers, and the jump to address indicated by register.

Comparison: In parsing applications, the branch condition generation often consists several data comparisons with Boolean operations to each other. To make the comparison more efficient, the comparison can use a logic operation such as AND/OR on a conditional flag register and the current comparison result, and write the logic result back to the conditional flag register.

Memory operations: 16-bit, 32-bit, and single bit load/store pairs of memory access operations are available. The single bit load is simply masked the unnecessary bits from the loaded data word, while the store operation is realized by adjusting the input bit at correct position and writing back.

3.3. Bitstream Processor Architecture

The block diagram for the bitstream processor is sown in Fig. 2. The processor is composed of four stages: Instruction Fetch(IF), Execution(EXE), Memory Load(MEML), and Write Back/Memory Store(WBMS). The instruction is fetched from the program memory by the program counter, and buffered by a register. At the

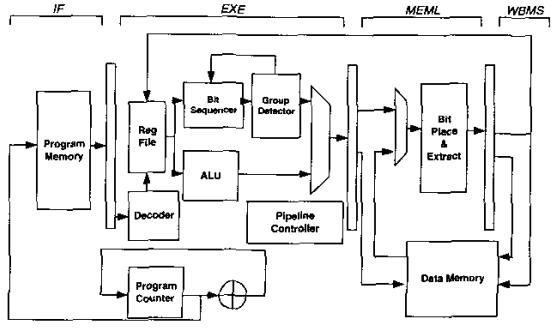


Fig. 2. Block Diagram of Proposed Bitstream Processor.

execution stages, a bit sequencer provides the basic bitstream functions such as show-bit and flush-bit operations. The group detector [7] works as an address generator by taking the most front bits of the bitstream from the sequencer and calculates the symbol address. Simultaneously it sends back the number of bits to be discarded to the sequencer to update the bitstream. Meanwhile, the ALU provides some arithmetic functions such as addition, subtraction, 8-bit multiplication and logic operations. The multiplexer at the end of the EXE stages selects the output between the group detector and the ALU. The group detector is applied here to perform codeword identification and provide VLC table programmability.

At the MEML stages, data read address generated at the previous stage appears at the read address port of the data memory. Once the data is read, it passes the bit placing and extracting block for bit replacement or bit extracting instructions. After the bit placement, some instructions need to write the data back to the memory, and the data write address is applied at the write address port. Meanwhile the data will be written back to the register file, which contains 32 16-bit registers, through the write port. A pipeline controller exists for monitoring the execution of each stage. It is responsible for clearing the pipeline registers if bubble has to be inserted, or stalling the pipeline when necessary.

To integrate the proposed design to a video decoding system, an interface module is designed. It handles parameter control by acquiring the parsed parameters and storing them in registers. The parameters for the decoding units are refreshed for every MB decoding. It's designed with hardwired state machine. A FIFO is provided inside the interface module for the decoded DCT coefficients from the bitstream processor. The interface module stalls either the following decoding units or the bitstream processor according to whether the FIFO is empty or full.

3.4. Stream Handler

To support the compressed domain data partitioning parsing discussed in Sec. 2.4, a video packet buffer to store the packet data and three addressing pointers for locating the start positions are required. In our design, the two components are included in stream handler, which is an interface between the bitstream sequencer and the external bitstream data input. Its block diagram is shown in Fig. 3 The video packet buffer is addressed by three addressing registers, which are corresponding to Pointer 0, 1 and 2, respectively. The processor can access the external bitstream by using

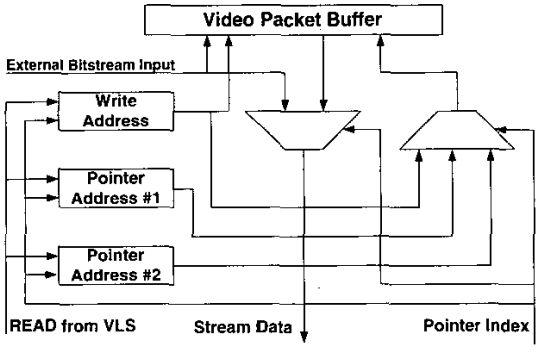


Fig. 3. Block Diagram of Stream Handler.

Table 4. Cycle Overhead for Proposed Data-partitioned Parsing Algorithm from Hardware Simulation.

Sequence	Combined	Data Partitioned	Overhead	Overhead(%)
Bream	204,516	225,104	20,588	10.07%
Weather	213,680	230,025	16,345	7.65%

Pointer 0, and the bitstream input data will be bypassed. Meanwhile, the stream handler writes the bitstream input data word into the video packet buffer to save the packet data. By using the Pointer 1 or 2, the addressing 1 or 2 will be activated to address the packet buffer and send the data to the processor. Once the whole packet has been parsed, a signal is passed to the stream handler to reset the addressing registers to prepare next packet parsing.

4. IMPLEMENTATION

The proposed bitstream processor is integrated into the decoding unit to form an MPEG-4 video decoding system. With TSMC 0.35um 1P4M technology, it operates at 33MHz under 3.3V to achieve MPEG-4 Advanced Simple Profile Level 5 (4CIF, 30fps) real-time decoding. Its gate count is 32,603.

The overhead for parsing data partitioned bitstream twice from hardware simulation is shown in Table. 4. As discussed in Sec. 2.4, the overhead is negligible. The comparison results with other implementations are shown in Table. 5. The proposed design achieves highest programmability with least required DCT coefficient decoding cycle and small memory requirement.

5. CONCLUSIONS

The MPEG-4 video bitstream parsing analysis and an efficient and flexible bitstream parsing processor are discussed in this paper. The bitstream parsing analysis explores that the most critical part in bitstream parsing lies in DCT coefficient codeword decoding. We propose approaches for DCT coefficients and data partitioned bitstreams. Based on the analysis results and proposed approaches, an efficient instruction set optimized for bitstream parsing is presented, and the processor architecture is proposed and implemented. It has been integrated in to an MPEG-4 video decoding system successfully and can achieve real time bitstream decoding with bitstream coded under 4CIF frame size with 30 fps, 8Mbps. This is the specification of MPEG-4 Advanced Simple Profile Level 5.

Table 5. Comparison between Different Parser Implementation.

Implementation	Proposed	[4]	[5]	[8]
Architecture	Processor	Processor	RISC Extension	Hardwire
Programmability	Yes	Compile time	Yes	No
DCT coefficients decoding cycle	1	4	6 to 10	1
Gate count	32K (RAM included)	24K	9K	20K
Memory size	1K Byte	1K Byte	8K Byte	50 Byte

6. REFERENCES

[1] Sigma Designs, "EM8470/EM8471/EM8475/EM8476 MPEG-4 decoder for set-top, DVD, and streaming applications," in *Sigma Designs Product Brief*, 2000.

[2] Toshiba, "MPEG-4 video decoder LSI TC35274," in *Tentative Technical Data Sheet*, 2000.

[3] Amphion Semiconductor Ltd., "CS6750-MPEG-4 video decoder," in *Data Sheet*, 2002.

[4] Y. C. Chang, H. C. Chang, and L. G. Chen, "Design and implementation of a bitstream parsing coprocessor for MPEG-4 video system-on-chip solution," in *International Symposium on VLSI-Technology, Systems, and Applications (VLSI-TSA'2001)*, 2001, pp. 188–191.

[5] M. Berekovic, H. J. Stolberg, M. B. Kulaczewski, and P. Pirsch, "Instruction set extension for MPEG-4 video," *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, vol. 23, no. 1, pp. 27–49, Oct. 1999.

[6] N. Seshan, "High VelociTI processing [Texas Instruments VLIW DSP architecture]," *IEEE Signal Processing Magazine*, vol. 15, no. 2, pp. 86–101, 117, Mar. 1998.

[7] B.J. Hsieh, Y. S. Lee, and C.Y. Lee, "A new approach of group-based VLC codec system with full table programmability," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 2, pp. 210–221, Feb. 2001.

[8] Y. C. Chang, W. H. Ji, and L. G. Chen, "A memory-efficient MPEG-4 simple scalable profile decoder with optimized motion compensation," in *3rd Workshop and Exhibition on MPEG-4 (WEMP4)*, 2002.