Journal of VLSI Signal Processing 41, 183–191, 2005

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, Taiwan

Received February 13, 2003; Revised May 11, 2004; Accepted July 30, 2004

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.

Keywords: MPEG-4, video decoding, bitstream parsing processor, data partitioned bitstream parsing

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, the coded data format for newer standard must be changed for either better compression technique or advanced coding options. Moreover, commercial video products are now beginning to support several coding standards simultaneously [1]. Various standards differ from their bitstream formats. The operation to extract the data hidden in the bitstream is called parsing. So, a versatile bitstream 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 because it lacks of flexibility. On the other hand, adaptation of an embedded processor will reduce the time-to-market. By re-programming the firmware, one can change the specification to match another new coding standard in a short time. Moreover, firmware upgrade can be accomplished by replacing the layout of the ROM in fabrication. Up till now, implementations of video decoder usually embed microprocessors on-chip to be the parsing unit [1–4].

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. Since bitstream parsing is bit-serial operation and is the first stage of the whole decoding task, the overall decoding system performance is determined by its throughput and efficiency. The introduction of processors needs more analysis and optimizations. For MPEG-4 video decoding system, both [6] and [5] propose such a solution. In [6], the instruction set extension for MPEG-4 bitstream parsing is proposed for general purpose RISC. In [5], a processor along with the instruction set dedicated for bitstream parsing is proposed. These previous designs emphasize the importance of variable-length code decoding(VLD)/fixed-length code decoding(FLD) operations, and propose enhanced datapath for VLD/FLD. In addition, a bitstream processor for object based MPEG-4 profile is also proposed in [7, 8].

In this paper, we propose an embedded bitstream processor for MPEG-4 video decoder. Based on our previous work [5] 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 Global Motion Compensation(GMC) and Quarter-pixel Motion Compensation (QMC)) Level 5 (720×576, 30fps) real time decoding.

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

2. Bitstream Parsing Analysis and Proposed Algorithms

A detailed analysis for MPEG-4 video bitstream structure has been discussed in [5], 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 accelerating 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. Most of the bits are DCT coefficient codewords, which occupies about 70% of the bitstream. The DCT coefficients not only occupy large portion but also occur successively in the bitstream. Besides, if the DCT coefficients cannot be decoded for the MPEG-4 video decoder in time, the decoding system has to be paused and the overall decoding performance is decreased. So, the operations for DCT coefficient decoding have to be optimized.

2.2. Proposed DCT Coefficients Parsing Approach

From hardware implementation viewpoint, 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.
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.

The conventional processor-based implementations [5, 6] for DCT coefficients parsing focus on the first to third parts, but ignore the fourth part presented. If we take it into account, the required cycle for VLD in [5] is 4 rather than 1, and that in [6] 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.

We perform simulation on this idea. 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 is 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 operation is merged with the VLD instruction such that the codeword decoding and branch condition checking can be accomplished within one cycle. The comparison on average required cycle to parse an I- and P-VOP is shown in Tables 1 and 2. The term "enhanced" means to merge the branch operation with VLD. It is shown that the improvement with the new merged instruction is between 20 and 50% of processing cycles. Thus, it is desirable to merge the essential branch decision operation with the VLD operation for a processor-based parser.

Table 1. Cycle analysis for I-VOP decoding.

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

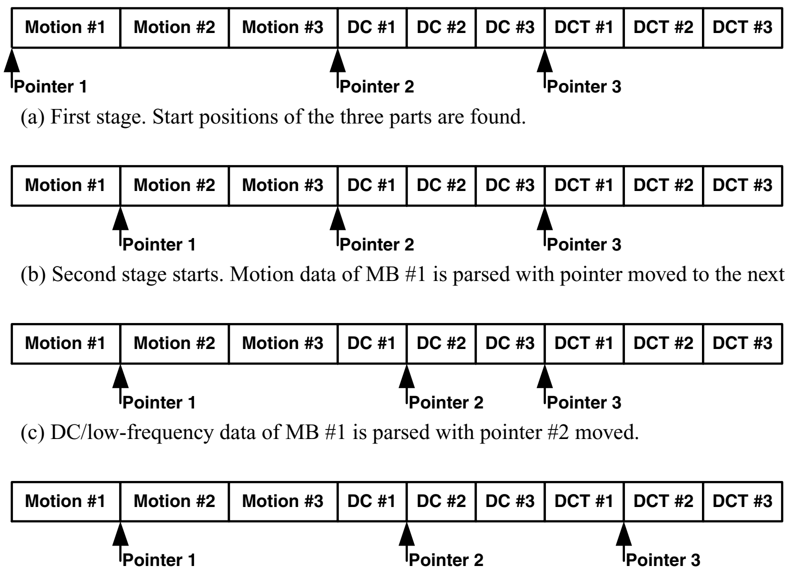
Table 2. Cycle analysis for P-VOP decoding.

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

2.3. Proposed Data Partitioned Parsing Approach

In MPEG-4 video standard, the bitstream structure for a P-VOP can be data partitioned or combined. In combined mode, the data is arranged in MB order. All data for a specific MB is put together. In data partitioned mode, 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, we have two approaches traditionally. One is to allocate a large storage space in either external or internal memory to store all the previously decoded motion data and DC data. These data can be read out only when the corresponding DCT coefficients for the block are decoded. The other one is to parse a video packet several times to obtain the necessary codeword for each MB. However, the former approach 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 approach, the required storage size can be reduced greatly. While decoding one frame with CIF size, only one packet buffer with maximum packet size,



(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).

Figure 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

which is 8 K bits at MPEG-4 Advanced Simple Profile Level 5, and one side buffer with size of about 700 bits are required. Compared with conventional implementation, which may demand 43 K bits, the proposed algorithm is more cost-effective.

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 cycles.

3. Architecture Design

3.1. Proposed Instruction Set

From the analysis results in Tables 1 and 2, the most critical part during bitstream parsing is DCT coefficient decoding. The DCT coefficient decoding involves two memory access operations in one decoding cycle. One is the read operation for symbol lookup, and the other is the write operation for data output. So, we target at single-cycle DCT coefficient decoding. Besides, the analysis about the parsing operations in [5] shows that the branch instruction occupies a large proportion. The occurrence of the branch is quite 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 [9]. By conditioning the execution with a flag (a one-bit register), every instruction can be controlled more freely than the branch architecture. Moreover, to provide more flexibility and for supporting other video coding standards in the future, the VLC tables are programmable.

In order to meet the above requirement, the instruction set is designed and can be divided into five categories according to its functionalities. The *bitstream operation instructions* contains a set of enhanced bitstream operations, including fixed-length decoding and variable-length decoding. To optimize the DCT coefficient decoding, as mentioned above, 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 since most of the RISC branch instructions are replaced by the conditional execution. The code for DCT coefficients parsing is simply shown as follows. The r1 stores the front 16-bit of the bitstream.

DCT_decoding:
REP.VLDS r1, AD_DCT_COEF

The *arithmetic instructions* contains Boolean logic operations, 16-bit addition and subtraction. An 8-bit multiplication 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. The *branch instructions* only contain jump with or without linking the return address to registers, and the jump to address indicated by register. In parsing applications, the branch condition generation often consists of several data comparisons with Boolean operations to each other. To make the comparison more efficient, the *comparison instructions* use logic operations 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. The *memory access instructions* contains 16-bit, 32-bit, and single bit load/store pairs of memory access operations. The single bit load is simply to mask 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. The bit-array operations, including one-bit signal comparison and memory access, are supported with the proposed instruction set.

3.2. 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 execution stages, a bit sequencer provides the basic bitstream functions such as show-bit and flush-bit operations. Meanwhile, the ALU provides some arithmetic

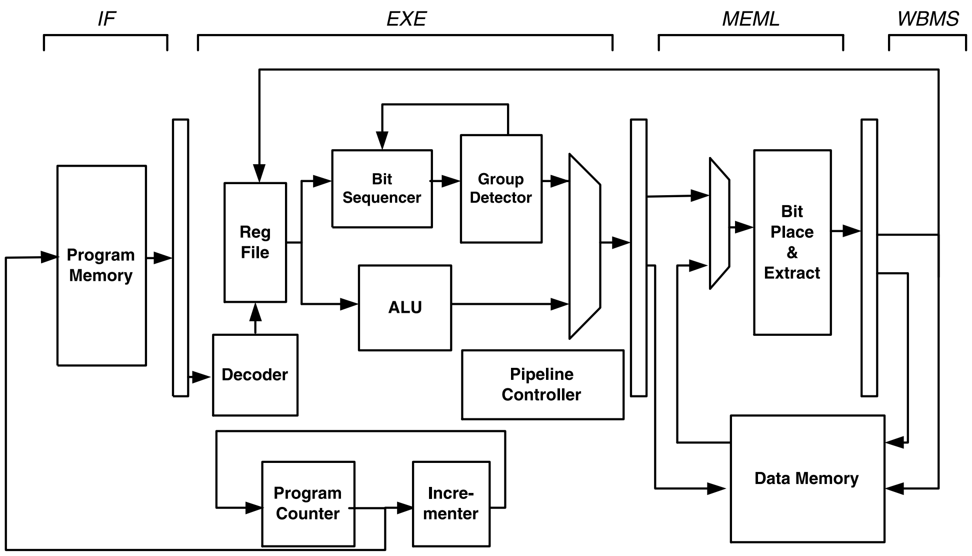


Figure 2. Block diagram of proposed bitstream processor.

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. 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.

The group detector [10] works as an address generator by taking the most front bits of the bitstream from the sequencer and calculates the symbol address. It is composed of 16 group detector cells since MPEG-4 TCOEF table has 15 groups of codewords. Each cell compares the bitstream to check if it is valid in this group. The group detector determines the valid group and calculates the symbol address. Simultaneously it sends back the number of bits to be discarded to the sequencer to update the bitstream. So, the group detector is applied here to perform *codeword identification* and provide VLC table programmability. Next, the *symbol lookup* and *branch decision* are executed when we load data from memory. Finally, the *symbol output* is accomplished by writing data back. When the REP.VLDS instruction is executed, the PC will stall. REP.VLDS will output one symbol in one clock cycle. At the same time, it checks the branch condition. If branch is taken, REP.VLDS breaks. If branch is not taken, REP.VLDS goes on. Branch decision is made by check the exception code field of the loaded symbol.

To integrate the proposed design to a video decoding system, an interface module is designed as shown in Fig. 3. It handles parameter control by acquiring

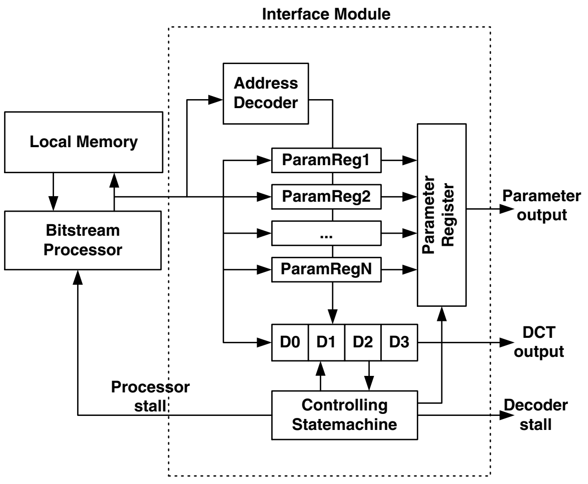


Figure 3. Block diagram of interface module.

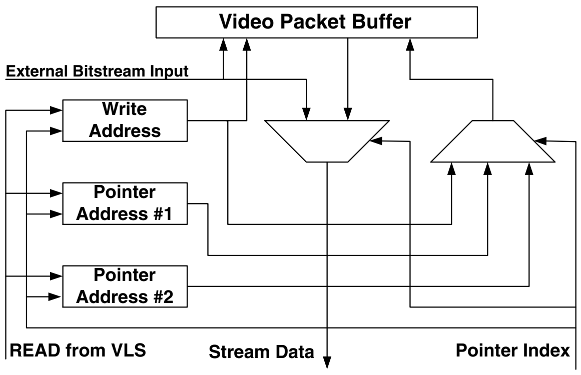


Figure 4. Block diagram of stream handler.

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

the parsed parameters and storing them in registers. The parameters for the decoding units are refreshed for every MB decoding. It is designed with hardwired state machines. 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.

3.3. Stream Handler

To support the compressed domain data partitioning parsing discussed in Section 2.3, 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

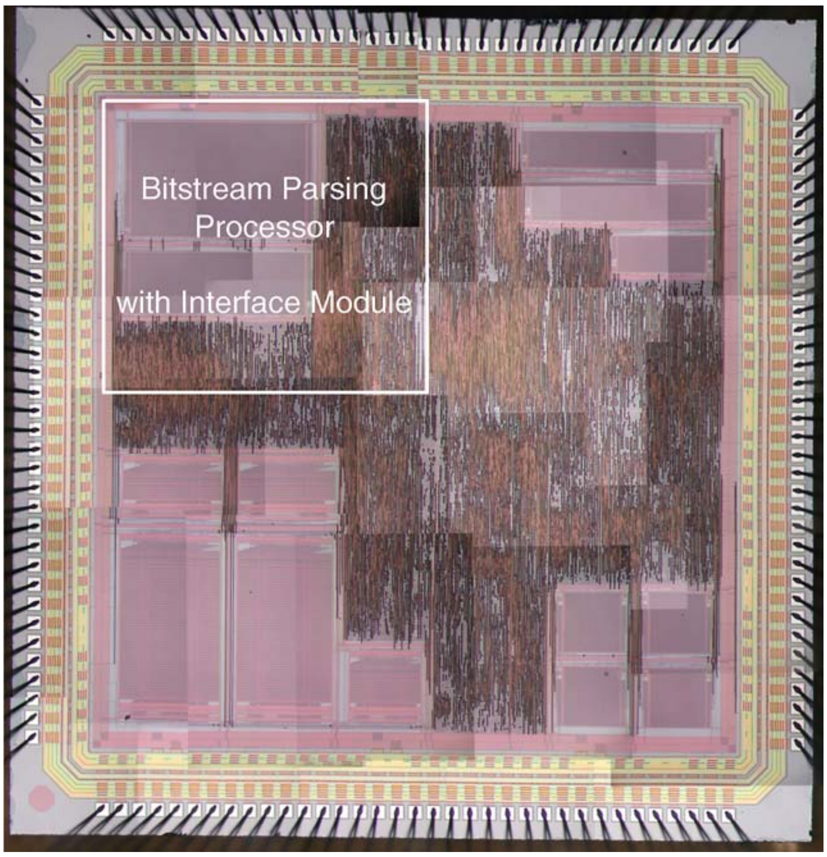


Figure 5. Chip implementation of the bitstream parsing processor embedded in an MPEG-4 video decoder.

Table 5. Comparison between different parser implementation.

Implementation	Proposed	[5]	[6]	[11]	TI C6X
Architecture	Processor	Processor	RISC Extension	Hardwire	DSP
Speed (MHz)	33	40	200	12	∼200
Technology (um)	0.35	0.35	0.25	0.35	–
Programmability	Yes	Compile time	Yes	No	Yes
DCT coefficients decoding cycle	1	4	6 to 10	1	10
Gate count	32 K (RAM included)	24 K	∼9 K	∼20 K	–
Memory size	∼1 K Byte	∼1 K Byte	∼8 K Byte	∼50 Byte	33.8 KB

is shown in Fig. 4. 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 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 and Comparison

The proposed bitstream processor is successfully integrated into the decoding unit to form an MPEG-4 video decoding system. With TSMC 0.35 um 1P4M technology, it operates at 33 MHz under 3.3 V to achieve MPEG-4 Advanced Simple Profile Level 5 (4CIF, 30fps) real-time decoding. Its gate count is 32,603. The chip implementation is shown in Fig. 5. The overhead for parsing data partitioned bitstream twice from hardware simulation is shown in Table. 4. As discussed in Section 2.3, the overhead is negligible.

The comparison results with other implementations are shown in Table 5. We compare the proposed design with [5, 6, 11], and TI C6X DSP [12, 13]. Among them, [5, 6], and C6X are programmable architecture, while [11] is a dedicated design. Chang et al. [5] is programmable at compile time. As discussed in Section 2, DCT coefficients parsing occupy most part in bitstream parsing. Only the proposed architecture can achieve single-cycle DCT coefficients decoding, which has same performance with the dedicated state-machine implementation. Besides, the proposed design achieves high performance with low operating frequency. So, 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, 8 Mbps. This is the specification of MPEG-4 Advanced Simple Profile Level 5.

References

1. Sigma Designs, "EM8470/EM8471/EM8475/EM8476 MPEG-4 decoder for set-top, DVD, and streaming applications," Product Brief, 2000.
2. Toshiba, "MPEG-4 video decoder LSI TC35274," Tentative Technical Data Sheet, April 2000.
3. Amphion Semiconductor Ltd., "CS6750-MPEG-4 video decoder," Data Sheet, April 2002.
4. J. Dunlop, A. Simpson, S. Masud, M. Wylie, J. Cochrane, and R. Kinkead, "Semiconductor IP core for ultra low power MPEG-4 video decode in system-on-silicon," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003, vol. 2, pp. 681–684.
5. 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.
6. 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.
7. K.A. Jacob, "MPEG-4 Main Profile Decoder Partitioning with respect to Bit Stream Processing," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2001, vol. 1, pp. 281–284.
8. J. Hormigo and K.A. Jacob, "Bit Stream Processor for Object Based MPEG-4 Profiles," in *IEEE Asilomar Conference on Signals, Systems and Computers*, 2001, vol. 2, pp. 1241–1245.
9. N. Seshan, "High VelociTI Processing [Texas Instruments VLIW DSP architecture]," *IEEE Signal Processing Magazine*, vol. 15, no. 2, pp. 86–101, 117, March 1998.
10. 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, Feb. 2001, pp. 210–221.
11. 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.
12. S. Sriram and C.Y. Hung, "MPEG-2 Video Decoding on the TMS320C6X DSP Architecture," in *IEEE Thirty-Second Asilomar Conference on Signals, Systems and Computers*, 1998, vol. 2, pp. 1735–1739.
13. TI, "MPEG-2 Video Decoder: TMS320C62x Implementation," Application Report, March 2000.

Yung-Chi Chang was born in Kaohsiung, Taiwan, R.O.C., in 1975. He received the B.S. and M.S. degrees from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., in 1998 and 2000, respectively, where he is currently pursuing the Ph.D. degree in the Graduate Institute of Electrical Engineering. His research interests include video coding algorithms and VLSI architectures for image/video processing.
watchman@video.ee.ntu.edu.tw

Chao-Chih Huang was born in Taiwan, R.O.C., in 1977. He received the B.S. and M.S. degree in electrical engineering from National Taiwan University in 2000 and 2002, respectively. In Oct 2002, he has joined the multimedia team of Realtek Taiwan, to be a system design engineer and researched on video coding algorithms. His research interests include video compression/coding and image processing.
gcc@video.ee.ntu.edu.tw

Wei-Min Chao was born in Taoyuan, Taiwan, R.O.C., in 1977. He received the B.S. and M.S. degrees from the Department of Electronics Engineering, National Taiwan University in 2000 and 2002 separately. His research interests include video coding algorithms and VLSI architecture for image and video processing.
hydra@video.ee.ntu.edu.tw

Liang-Gee Chen was born in Yun-Lin, Taiwan, in 1956. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, in 1979, 1981, and 1986, respectively. He was an Instructor (1981–1986), and an Associate Professor (1986–1988) in the Department of Electrical Engineering, National Cheng Kung University. In the military service during 1987 to 1988, he was an Associate Professor in the Institute of Resource Management, Defense Management College. In 1988, he joined the Department of Electrical Engineering, National Taiwan University. During 1993 to 1994 he was a Visiting Consultant of DSP Research Department, AT&T Bell Lab, Murray Hill. In 1997, he was a visiting scholar of the Department of Electrical Engineering, University of Washington, Seattle. During 2001 to 2004, he was the first director of the Graduate Institute of Electronics Engineering (GIEE) in National Taiwan University (NTU). Currently, he is a Professor of the Department of Electrical Engineering and GIEE in NTU, Taipei, Taiwan. He is also the director of the Electronics Research and Service Organization in Industrial Technology Research Institute, Hsinchu, Taiwan. His current research interests are DSP architecture design, video processor design, and video coding systems.

Dr. Chen has served as an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology since 1996, as Associate Editor of IEEE Transactions on VLSI Systems since 1999, and as Associate Editor of IEEE Transactions on Circuits and Systems

II since 2000. He has been the Associate Editor of the Journal of Circuits, Systems, and Signal Processing since 1999, and a Guest Editor for the Journal of Video Signal Processing Systems. He is also the Associate Editor of the Proceedings of the IEEE. He was the General Chairman of the 7th VLSI Design/CAD Symposium in 1995 and of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He is the Past-Chair of Taipei Chapter of IEEE Circuits and Systems (CAS) Society, and is a member of the IEEE CAS Technical Committee of VLSI Systems and Applications, the Technical Committee of Visual Signal Processing and Communications, and the IEEE Signal Processing Technical Committee of Design and Implementation of SP Systems. He is the Chair-Elect of the IEEE CAS Technical Committee on Multimedia Systems and Applications. During 2001–2002, he served as a Distinguished Lecturer of the IEEE CAS Society. He received the Best Paper Award from the R.O.C. Computer Society in 1990 and 1994. Annually from 1991 to 1999, he received Long-Term (Acer) Paper Awards. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on circuits and systems in the VLSI design track. In 1993, he received the Annual Paper Award of the Chinese Engineer Society. In 1996 and 2000, he received the Outstanding Research Award from the National Science Council, and in 2000, the Dragon Excellence Award from Acer. He is a member of Phi Tan Phi.
lgchen@video.ee.ntu.edu.tw