ASIP Instructions and Their Hardware Architecture for H.264/AVC

Jung H. Lee, Sung D. Kim, and Myung H. Sunwoo

Abstract—**H.264/AVC adopts new features compared with previous multimedia algorithms. It is inefficient to implement some of the new blocks using existing DSP instructions. Hence, new instructions are required to implement H.264/AVC. This paper proposes novel instructions for intra-prediction, in-loop deblocking filter, entropy coding and integer transform. Performance comparisons show that the required computation cycles for the in-loop deblocking filter can be reduced about 20 ~ 25%. This paper also proposes new instructions for the integer transform. The proposed instructions can execute one dimension forward/inverse integer transform. The integer transform can be implemented using much smaller hardware size than existing DSPs.**

Index Terms—**Multimedia, H.264, ASSP, Instruction**

I. INTRODUCTION

With the rapid progress of semiconductor technology, the market of Application-Specific Signal Processor (ASSP) is dramatically growing. Once algorithms have been fixed, custom Application-Specific Integrated Circuit (ASIC) chips have been implemented to reduce the cost, size, and power consumption of systems. However, custom ASIC solutions have been found inadequate to upgrade standards since they should be redesigned. With the rapid increase in clock speed it has become feasible to keep the functionality entirely in a programmable DSP, greatly improving time-to-market and allowing faster changes and upgrades. ASSP can compromise advantages of custom ASIC chips and general DSP chips [1][2][3]. In other words, ASSP chips adopt high performance and low power of ASIC chips and flexibility of DSP chips.

Multimedia signal processing technology has been developed with the progress of semiconductor technology. Technology related to multimedia signal processing has been standardized as MPEG-2, MPEG-4, H.261, H.263, etc. Recently, the Joint Video Team (JVT) announced H.264/AVC in Dec. 2003 [4]. The new video coding standard H.264/AVC can provide twice as much as higher compression efficiency than MPEG-4. However, it is hard to implement H.264/AVC since it requires 10 times more hardware complexity for an encoder, and 2 times more hardware complexity for a decoder than those of MPEG-4. H.264/AVC has a number of control parts that has been implemented using programmable processors, such as ARM and DSP [5], and computation-intensive parts have been designed using hardwired accelerators.

Existing DSPs has various application specific instructions for multimedia algorithms. However, newly adopted features of H.264/AVC, previous instructions are not efficient to support H.264/AVC. This paper shortly introduces new features of H.264/AVC. Then it proposes novel instructions and their hardware architecture to efficiently implement the new features on ASSP.

This paper is organized as follows. Section 2 analyzes H.264/AVC and describes existing DSP instructions to implement multimedia standards. Section 3 proposes novel instructions and their hardware architectures, and Section 4 explains performance comparisons. Finally, Section 5

Manuscript received October 20, 2005; revised December 3, 2005.
School of Electrical and Computer Engineering, Ajou University San5, Wonchun-Dong, Yeongtong-Gu, Suwon, Korea
E-mail : sunwoo@ajou.ac.kr

contains concluding remarks.

II. Existing DSP Instructions for Video Signal Processing

Existing DSPs support various instructions to execute packed operations between two registers. These operations are used for various video signal processing, such as motion estimation and compensation, DCT/IDCT, etc. TMS320c6x of Texas Instruments supports special instructions for multimedia signal processing, such as SUBABS4, AVGx, etc [6]. The SUBABS4 instruction calculates absolute differences of four pairs of the packed data. The AVG4 instruction calculates averages of the packed data in two registers. After addition operations of four packed data, four results are shifted a bit to the left for division, and 0.5 is added to each result for rounding. TMS320c6x series also supports the DOTPU4 instruction which calculates the dot product between four sets of packed 8 bit values. The values in both src1 and src2 are treated as unsigned, 8 bit packed data. The 32 bit unsigned result is written into dst. Four clock cycles are required to execute this instruction. Fig. 1 shows the operation flow of the DOTPU4 instruction.

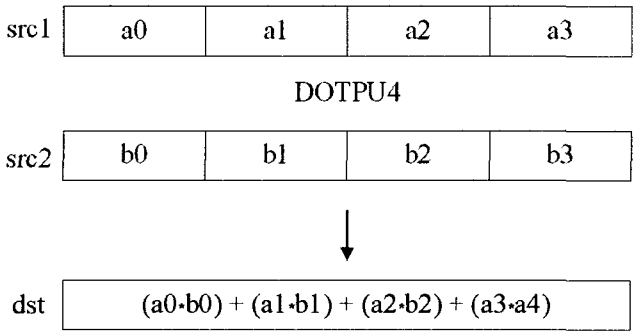


Fig. 1. DOTPU4 instruction in TMS320c64x.

DCT has a regular computation flow, while motion estimation/compensation and entropy coding have control based computations. TMS320c55x has a coprocessor for DCT computations, and it requires 2.8 MIPS for DCT computations to achieve a processing speed of 30 fps for the QCIF format. TMS320c6x having eight function units requires 1.1 MIPS to implement DCT of 30 QCIF fps video data using DSP instructions [7].

In entropy coding, the code word is obtained based on the number of successive zeros in the input bit stream in the code word table. Moreover, packed compare operations are required. To execute these operations, TMS320c64x supports the LMBD and CMPEQ/GT/LT instructions, and the Blackfin DSP of Analog Device supports the ONES instruction [7][8]. The LMBD instruction counts the number of zeros in a register. The CMPEQ/GT/LT instructions compare pairs of 8 bits or 16 bit packed data.

III. Novel DSP Instruction and Their Hardware Architecture

This section describes ASSP instructions and their hardware architectures for H.254/AVC codecs.

1. Proposed Instructions for In-loop Deblocking Filter, Intra-prediction and Entropy Coding

The in-loop filter is used to eliminate blocking artifacts as mentioned in Section 2. Fig. 2 shows 8 pixels of neighboring 4 x 4 blocks. The 8 pixel values are decided according to the boundary strength (bS), which represents the difference of two neighboring blocks, using p0 ~ p3 and q0 ~ q3. The equations calculating pixel values are defined in the specification [4]. The equations can be classified into five categories as follows.

$$p2+p1+p0 \quad (1)$$
$$p2+2\times p1+2\times p0 \quad (2)$$
$$2\times p3+3\times p2+p1+p0 \quad (3)$$
$$2\times p1+p0 \quad (4)$$
$$(p0+q0+1)>>1 \quad (5)$$

p0 ~ p3 are the packed data in a register, and q0 ~ q3 are also the packed data in another register. Then, equation (1) shows additions of three packed data in one register. Equation (2) represents one bit shift left operations of two data followed by additions of three packed data in the same register. Equation (3) shows one bit shift left operation of data and a multiplication operation of data

followed by additions of four packed data. Equation (4) shows one bit shift left operation of the packed data followed by an addition of two packed data. Equation (5) shows an addition of the most significant byte (MSB) of one register and least significant byte (LSB) of the other register followed by one bit shift operation.

Even though these computations are packed operations, these operations do not occur between two registers as shown in Fig. 2, but they occur between the packed data within the same register.

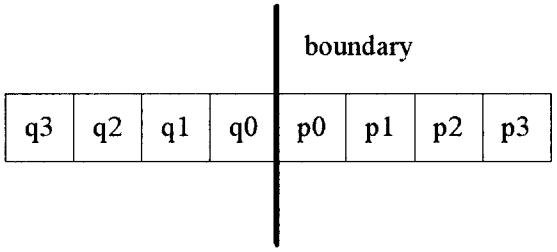


Fig. 2. Block boundary.

As mentioned in Section 2, the intra-prediction eliminates redundancy of intra-frame and inter-frame, which has few redundancies between two frames. Fig. 3 shows an identification of samples for 4x4 intra prediction. a ~ p of Fig. 3 are predicted using A ~ Q according to the equations defined in the specification [4] and some of equations are represented in equation (6). In equation (6), A, B, C, represent pixel values, a pixel value is represented using 8 bits.

Q	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				
M								
N								
O								
P								

Fig. 3. Identification of samples for 4 x 4 intra prediction.

CAVLC of entropy coding requires calculating the number of ones or zeros in a codeword. These operations can perform the packed compared operation followed by the packed addition operations. These packed additions are not the additions between registers but the additions of the packed data within a register. As described in Section 2, existing DSPs support only packed operations between two registers. A large number of instruction cycles is required to implement the in-loop deblocking filter, intra-prediction and CAVLC with the existing packed instructions, which execute packed operations between two registers. Hence, H.264/AVC may require a new instruction to execute packed operations within a register.

Fig.4 shows the proposed three horizontal addition (hadd) instructions.

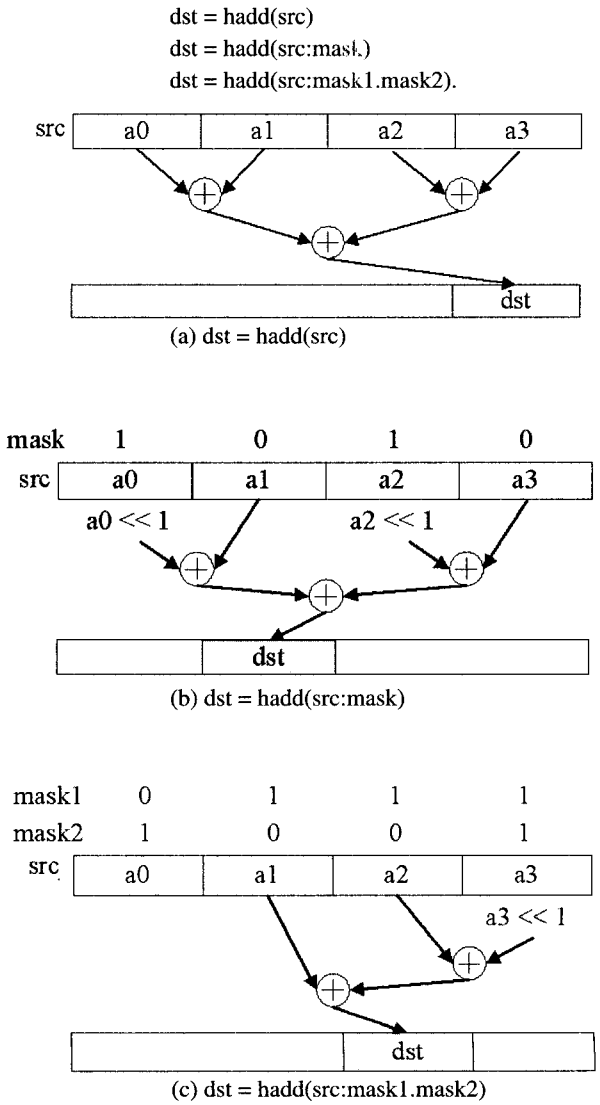


Fig. 4. Proposed instructions for packed additions within one register.

Three hadd instructions are as follows. The proposed instructions in Fig. 4(a) pack a 32 bit register into four 8 bit data, add four packed data, and then saturate the result to 8 bit data. Fig. 4(b) is similar with Fig. 4(a). However, the packed data, which is selected by a mask, is one bit shifted to the left. In Fig. 4(c), mask1 selects the data to be added, and mask2 selects the data to be shifted. Intra-prediction, entropy coding, and equations (1), (2), (4), (5) of the in-loop deblocking filter can be implemented using the proposed instructions. Equation (3) can be implemented using the packed multiplication instruction such as the DOTPU4 instruction of TMS320c6x or the PMUL of proposed ASSP. Equations (1), (2), (4), (5) can also be implemented using existing packed multiplication instructions. However DOTPU4 instruction requires four clock cycles since the multiplication should be executed

2. Proposed Instructions for Integer Transform

The 4 x 4 integer transform can be performed using the forward transform as shown in Fig. 5(a). The forward transform is performed for four rows of four packed data. Then, the forward transform is performed again for four columns of four packed data to get the results of the 4 x 4 integer transform. Fig. 5(b) represents an inverse transform. Similarly, the 4x4 inverse integer transform can be executed using the operations in Fig. 5(b).

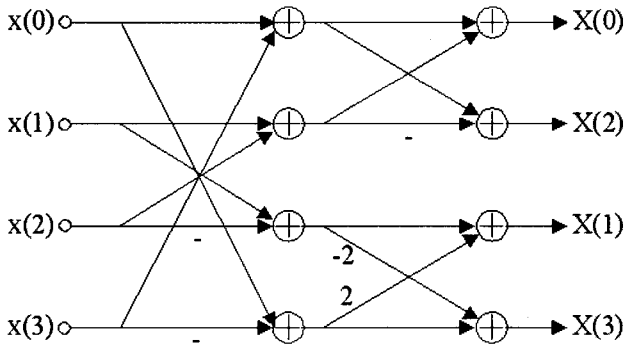
This paper proposes novel instructions to efficiently execute the forward/inverse 4x4 integer transform as follows.

dst = fTRAN (src)
dst = iTRAN (src).

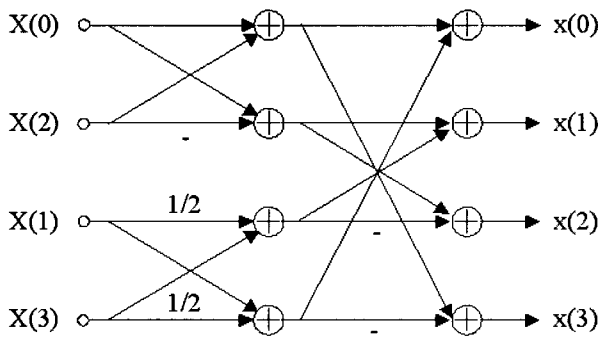
Each instruction performs the operation of Fig. 5 (a) and (b). These instructions read a 32 bit general register in one register file, which consists of four 32 bit registers, and execute the operation flow in Fig. 5. Then, the results are written in another register files consisting of four 32 bit registers. These instructions can be implemented using existing adders and eight additional 2 x 1 multiplexers.

3. Hardware Architecture for Proposed Instructions

Fig. 6 shows the Arithmetic Logic Unit (ALU) for the proposed instructions. Switching Logic 1, 2, and 3 which only consist of eight 2 x 1 multiplexers and two 1 x 2 de-multiplexers, are only the additional hardware for the proposed instructions. 1 bit shift operation in Figure 4 is executed using a multiplexer. One ALU can perform one horizontal addition instruction. Only two continuous ALU operations can execute fTRAN and iTRAN instructions.



(a) 1-D forward transform



(b) 1-D inverse transform

Fig. 5. Operation flow of 4 x 4 integer transform.

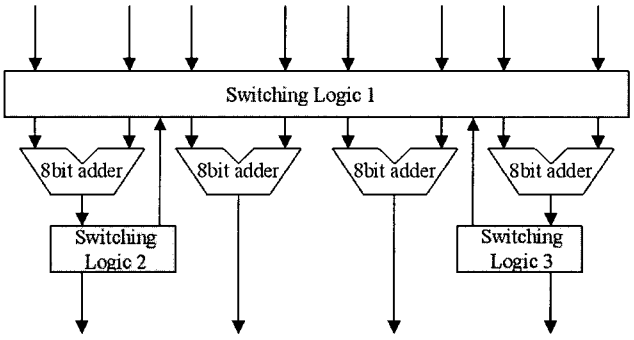


Fig. 6. ALU for the proposed instructions.

IV. Performance Comparisons

Several key blocks for generating intra-predictors of the in-loop deblocking filter are coded using the proposed instructions and the same blocks are also coded using existing instructions of TMS320c64x. The proposed architecture can reduce the number of clock cycles for several key blocks for generating intra-predictors about 40% than TMS320c6x. Hence, the total number of clock cycles taken to execute in-loop deblocking filter can be reduced about 20 ~ 25% since several coded blocks take about 34% of total computation cycles. TMS320C64x supports the DOTPU4 instruction that executes packed multiplications of two registers and adds four results in four cycles. The computation cycles of TMS320c64x can be reduced, since it supports the DOTPU4 instruction. Hence, other DSPs, which do not support the special instruction, require more instructions.

The fTRAN and iTRAN instructions can be executed in one cycle. Hence, 12 clock cycles are required to execute the 4x4 integer transform using the proposed instructions and about 1,140,480 instructions for 30 frames ((24 cycles x 16 blocks) x 99 macro blocks x 30 frame) are required for QCIF images, since a QCIF image has 99 16x16 macro blocks. Table I shows the number of the required instructions for 30 frames on existing DSPs and the proposed ASSP. The proposed ASSP having the special instructions can be more efficient than the implementation using instructions of 55x (SW) and using a coprocessor of 55x (HW) for the integer transform. TMS320c64x has a VLIW architecture and has eight function units while the proposed architecture requires only two 32 bit adders.

Table. 1. Performance comparisons of 4x4 integer transform

	55x (SW)	55x (HW)[9]	64x [7]	Proposed ASSP
MIPS	12.8	2.8	1.1	1.1

V. Conclusions

This paper proposes efficient instructions and their hardware architecture to implement the in-loop deblocking filter, intra-prediction, entropy coding, and integer transform of H.264/AVC. Three hadd instructions can execute various packed additions within a register. Performance comparisons shows that the number of clock cycles can be reduced about 20 ~ 25% compared with the existing DSP for the in-loop deblocking filter. The additional hardware requires only eight 2 x 1 multiplexers and two 1 x 2 demultiplexer. This paper also proposes new instructions, fTRAN and iTRAN, for the integer transform. The integer transform can be implemented using much smaller hardware size compared with existing DSPs.

References

[1] Jae S. Lee, Young S. Jeon, and Myung H. Sunwoo, "Design of new DSP instructions and their hardware architecture for high-speed FFT," in *Proc. IEEE Workshop on Signal Processing Syst.*, pp. 80-90, Sept. 2001.

[2] J. Glossner, J. Moreno, M. Moudgill, J. Derby, E. Hokenek, D. Meltzer, U. Shavadron, and M. Ware, "Trends in compilable DSP architecture," in *Proc. IEEE Workshop on Signal Processing Syst.*, pp. 181-199, 2000.

[3] Jeong H. Lee, Jong H. Moon, Kyung L. Heo, Myung H. Sunwoo, Sung K. Oh, and In H. Kim, "Implementation of Application Specific DSP for OFDM Systems," in *Proc, IEEE IEEE Int. Symp. Circuti Syst.* (ISCAS'2004), May 2004.

[4] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 (E) AVC). July, 2004.

[5] *H.263 Encoder: TMS320C6000 Implementation*, Texas Instrument Inc., Dallas, TX, 2000.

[6] *TMS320C6000 CPU and Instruction Set Reference Guide*, Texas Instruments Inc., Dallas, TX, 2000.

[7] *TMS320C64x Image/Video Processing Library*, Texas Instruments Inc., Dallas, TX, 2003.
[8] *BlackfinTM DSP Instruction Set Reference*, Analog Device Inc., Norwood, Mass. 2002.
[9] TMS320C55x Hardware Extensions for Image/Video Applications Programmer's Reference, Texas Instruments Inc., Dallas, TX, 2000.

Junghoo Lee He received the B.S. degree in electronic engineering from Ajou University, Suwon, Korea in 2002. He is currently working toward the Ph.D. degree with School of Electrical and Computer Engineering, Ajou University. His main research interests include SOC design and application-specific DSP chip design for communications and multimedia applications.

Sungdae Kim He received the B.S. degree in electronic engineering from Ajou University, Suwon, Korea in 2003. He is currently working toward the Ph.D. degree with School of Electrical and Computer Engineering, Ajou University. His main research interests include SOC design and application-specific DSP chip design for communications and multimedia applications.

Myung H. Sunwoo He received the B.S. degree in electronic engineering from the Sogang University in 1980, the M.S. degree in electrical and electronics from the Korea Advanced Institute of Science and Technology in 1982, and the Ph.D. in electrical and computer engineering from The University of Texas at Austin in 1990.
He worked for Electronics and Telecommunications Research Institute (ETRI) in Taejon, Korea from 1982 to 1985 and Digital Signal Processor Operations Division, Motorola, U.S.A from 1990 to 1992. Since 1992, he has been a Professor with the School of Electrical and Computer Engineering, Ajou University in Suwon, Korea. He is a Director of the National Research Laboratory sponsored by the Korean Government. His research interests include VLSI architectures, SOC design for multimedia and communications, and application-specific DSP chip design.
He is the author of more than 100 journal and conference papers and holds two U.S. patents. He served as a Technical Program Chair of the IEEE Workshop on Signal Processing Systems (SIPS) in 2003. He has been a VLSI Systems and Application Technical Committee (VSATC) Member of the IEEE Circuits and Systems (CAS) Society since 1996 and a Design and Implementation of Signal Processing Systems (DISPS) Technical Committee Member of the IEEE Signal Processing Society since 2004. He has also served on the Technical Program Committee of various international conferences including the IEEE Workshop on SIPS and the IEEE International SOC Conference. He served as an Associate Editor for the IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2002-2003) and is serving as a Guest Editor for the Journal of VLSI Signal Processing (Kluwer, 2004). Currently, He is a Senior Member of IEEE and a Chair of the IEEE CAS Society of the Seoul Chapter.