ASIP Approach for Implementation of H.264/AVC

Sung Dae Kim, Jeong Hoo Lee, Chung Jin Hyun and Myung Hoon Sunwoo

School of Electrical and Computer Engineering, Ajou University
San 5, Wonchun-Dong, Yeungtong-Gu
Suwon, 443-749 Korea
Tel : 82-31-219-2390
Fax : 82-31-212-9531
e-mail : sunwoo@ajou.ac.kr

Abstract - This paper presents an Application-Specific Instruction Set Processor (ASIP) approach for implementation of H.264/AVC. The proposed ASIP has special instructions for intra prediction, deblocking filter, integer transform, etc. In addition, the proposed ASIP has hardware accelerators for inter prediction and entropy coding. Performance comparisons show a significant improvement compared with existing DSPs. The proposed hardware accelerators have small size and can support real-time video processing. Moreover, the proposed ASIP can handle various multimedia standards. The results indicate that the ASIP approach is one of promising solutions for H.264/AVC.

I. INTRODUCTION

With the rapid progress of semiconductor technology, the market of ASIP 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. However, a programmable DSP should solve the disadvantages, such as cost, size, and power consumption. ASIP can compromise advantages of custom ASIC chips and general DSP chips [1]-[5]. In other words, ASIP 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. The Joint Video Team (JVT) announced H.264/AVC in Dec. 2003 [6]. The new video coding standard H.264/AVC can provide twice as much as higher compression efficiency than MPEG-4. However, it has about 2 times more hardware complexity for a decoder, and about 10 times more hardware complexity for an encoder than the MPEG-4 visual simple profile codec [7]. Because of the hardware complexity, the H.264/AVC codec is usually implemented with ASIC chips or multiple processors, such as ARM and multiple programmable DSPs.

In mobile communication, the implementation of H.264/AVC needs high performance, low power consumption and low cost. It also requires the flexible system which can upgrade without replacing the system. The ASIP approach can be suitable for these requirements. This paper proposes the ASIP implementation of H.264/AVC. Computation-intensive parts in H.264/AVC have been implemented using hardwired accelerators and other parts have been implemented using a programmable DSP.

This paper is organized as follows. Section II analyzes H.264/AVC and describes existing DSP instructions to implement multimedia standards. Section III proposes novel instructions and their hardware architectures and introduces hardware accelerators, and Section IV explains performance comparisons. Finally, Section V contains concluding remarks.

II. ANALYZING H.264/AVC AND EXISTING DSP INSTRUCTIONS FOR VIDEO SIGNAL PROCESSING

A. Analyzing H.264/AVC

H.264/AVC has adopted new features to improve code efficiency. The new features are as follows. H.264/AVC uses several reference frames, variable block size, and a quarter picture element in Motion Estimation (ME)/Motion Compensation (MC). These features enable the encoder to search for the best match for the current frame. However, the memory access and hardware complexity are significantly increased. The past standards should be transmitted the first frame without compression. On the other hand, the H.264/AVC encoder adopts intra prediction, which eliminates the redundancy of intra frame.

The block based structure causes blocking artifacts. H.264/AVC adopts the in-loop deblocking filter to eliminate blocking artifacts. The Exponential Golomb Coding (EGC) and Context Adaptive Variable Length Coding (CAVLC) are also newly adopted features of the H.264/AVC baseline

0-7803-9451-8/06/$20.00

profile. EGC is variable length codes with a regular construction [8]. CAVLC is the method used to encode residual data, 4 x 4 blocks of transform coefficients [9]-[11].

Fig. 1 shows the operation complexity of the H.264/AVC baseline profile [12]. As shown in Fig. 1, MC takes 53% and VLC takes 18.20% of the operation complexity. Since these features require heavy computational loads, the hardware accelerator of each feature is required to implement the H.264/AVC system. Therefore, the proposed ASIP employs the hardware accelerators for these tasks, which can efficiently perform real-time video processing.

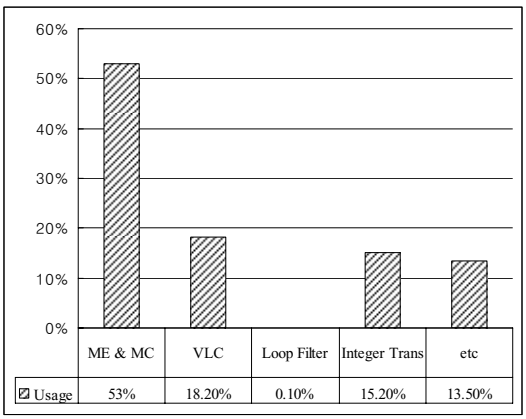


Fig. 1. Complexity analysis of the H.264/AVC baseline profile

B. 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 DCT, IDCT, ME/MC, etc. TMS320c6x of Texas Instruments supports special instructions for multimedia signal processing, such as SUBABS4, AVGx, etc. [13]. 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 additions 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. The TMS320c6x series also support the DOTPU4 instruction which calculates the dot product between four sets of packed 8 bit values. Fig. 2 shows the operation flow of the DOTPU4 instruction. The values in both src1 and src2 are treated as the unsigned 8 bit packed data. The 32 bit unsigned result is written into dst. Four clock cycles are required to execute this instruction.

DCT has a regular computation flow, while ME/MC 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 the 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 [14].

In entropy coding, the code word table is referred according to the number of successive zeros in the input bit stream. 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 [14] [15]. 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.

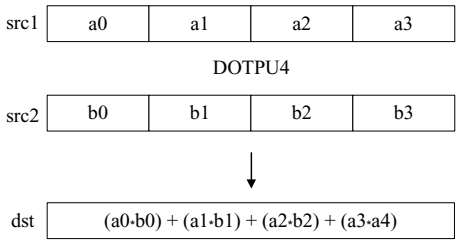


Fig. 2. DOTPU4 instruction in TMS320c64x

III. NOVEL INSTRUCTIONS AND HARDWARE ACCELERATORS

This section presents an overall architecture, new instructions and hardware accelerators for the H.264/AVC codec.

A. Overall architecture of the proposed ASIP

Fig. 3 shows the overall architecture of the proposed ASIP. The proposed ASIP consists of two parts, a programmable DSP part and a hardware accelerator part. The DSP part has a program control unit (PCU), a data processing unit (DPU), and an address unit (AU). The hardware accelerator part has an Inter Prediction Accelerator (IPA) and an Entropy Coding Accelerator (ECA). IPA consists of an ME accelerator and an MC accelerator. ECA has a CAVLC accelerator and an EGC accelerator. The hardware accelerators can operate in parallel with the DSP units.

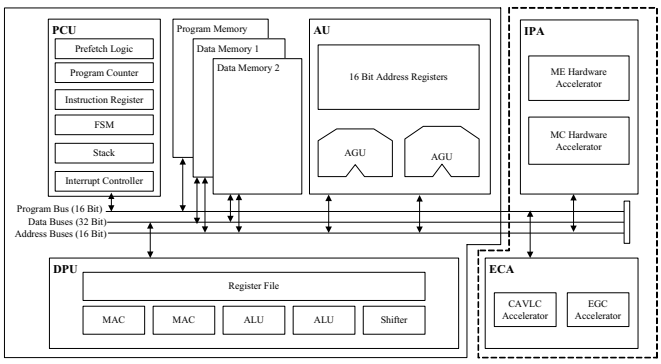


Fig. 3. Proposed ASIP architecture

PCU consists of a prefetch logic, a program counter, an instruction register, an FSM (Finite State Machine), a stack, and an interrupt controller. DPU consists of two Multiply

and Accumulate (MAC) units for two 16-bit by 16-bit multiplications and accumulations, two Arithmetic Logic units (ALU), a barrel shifter and a register file. AU has two address generation units (AGU) for load and store. Each of the internal word lengths is 32 bit. The instruction pipeline consists of six stages, that is, pre-fetch, fetch, decode, execute1, execute2, and execute3. The proposed ASIP has 35 arithmetic instructions, 11 logical and shift instructions, 6 program control instructions, 4 move instructions and 16 special instructions including instructions for H.264/AVC, which will be discussed next.

B. Proposed instructions for in-loop deblocking filter and intra prediction

The in-loop deblocking filter is used to eliminate blocking artifacts as mentioned in Section II. Fig. 4 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 [6]. 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 the 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. 4, but they occur between the packed data within the same register.

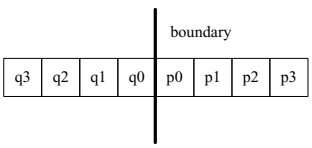


Fig. 4. Block boundary

As mentioned in Section II, the intra prediction eliminates the redundancy of intra frame and inter frame, which has few redundancies between two frames. Fig. 5 shows an identification of samples for 4 x 4 intra prediction. a ~ p in Fig. 5 are predicted using A ~ Q according to the equations defined in [6] and some of equations are represented in equation (6), where A, B, and C represent pixel values, and a pixel value is represented using 8 bits. For a 32 bit architecture, A, B, C and D are stored in one register since a ~ p and A ~ Q in Fig. 5 are 8 bit values.

$$\begin{aligned} &(A + 2 \times B + C + 2) >> 2 \\ &(A + B + 1) >> 1 \\ &(A + 2 \times B + 1) >> 1 \end{aligned} \quad (6)$$

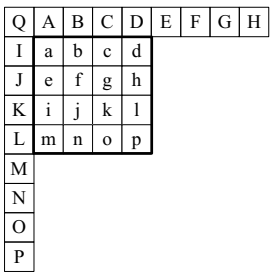


Fig. 5. Identification of samples for 4 x 4 intra prediction

As described in Section II, existing DSPs support only packed operations between two registers. A large number of instruction cycles is required to implement the in-loop deblocking filter and intra prediction with the existing packed instructions that execute packed operations between two registers. Hence, H.264/AVC may require a new instruction to execute packed operations within a register.

Fig. 6 shows the proposed three horizontal addition (hadd) instructions. Three hadd instructions are as follows. The proposed instruction in Fig. 6(a) packs a 32 bit register into four 8 bit data, adds four packed data, and then saturates the result to 8 bit data. Fig. 6(b) is similar with Fig. 6(a). However, the packed data, which is selected by a mask, is one bit shifted to the left. In Fig. 6(c), mask1 selects the data to be added, and mask2 selects the data to be shifted. Intra prediction 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 instruction of the proposed ASIP. Equations (1), (2), (4), (5) can also be implemented using the existing packed multiplication instructions. However, the DOTPU4 instruction in TMS320c6x requires four clock cycles since a multiplication should be executed.

```
dst = hadd(src)
dst = hadd(src:mask)
dst = hadd(src:mask1.mask2)
```

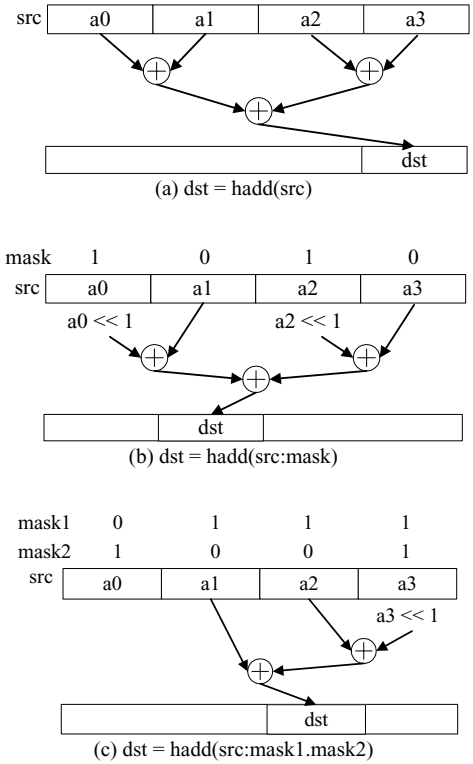


Fig. 6. Proposed instructions for packed additions within one register

C. *Proposed instructions for integer transform*

The 4 x 4 integer transform can be operated using the forward transform as shown in Fig. 7(a). The forward transform is executed with four rows of four packed data. Then, the forward transform is performed again with four columns of four packed data to get the results of the 4 x 4 integer transform. Fig. 7(b) represents an inverse transform. Similarly, the 4 x 4 inverse integer transform can be executed using the operations in Fig. 7(b).

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

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

Each instruction performs the operations of Fig. 7(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. 7. Then, the results are written in another register file consisting of four 32 bit registers. These instructions can be implemented using the adders and eight additional 2 x 1 multiplexers.

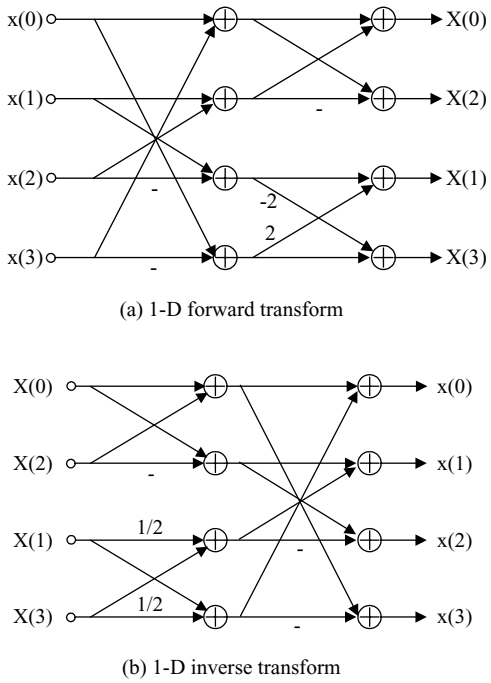


Fig. 7. Operation flow of 4 x 4 integer transform

D. *Hardware architecture for proposed instructions*

Fig. 8 shows the 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. One ALU can perform one horizontal addition instruction and two ALUs can execute fTRAN and iTRAN instructions.

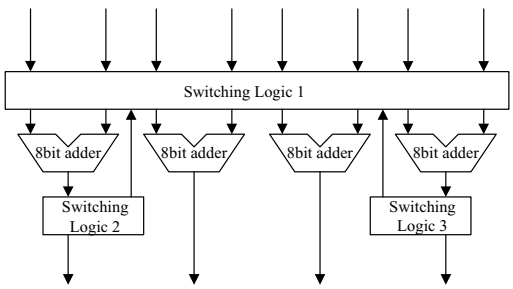
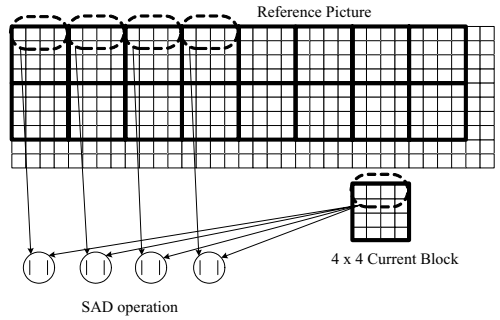


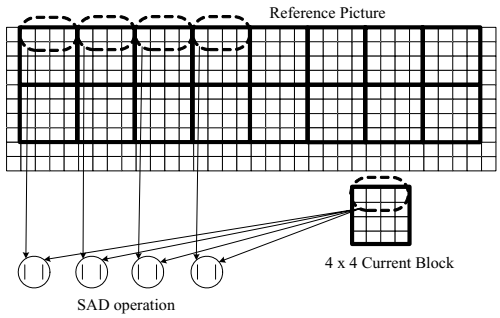
Fig. 8. ALU for the proposed instructions

E. *Proposed hardware accelerator for inter prediction*

As described in Section II, ME/MC should frequently access memory. From a performance point of view and a low power point of view, it is a serious problem. Thus, the sliding window method is used to alleviate this problem [16]. Fig. 9 illustrates the proposed ME operation flow.



(a) ME operation in the first cycle

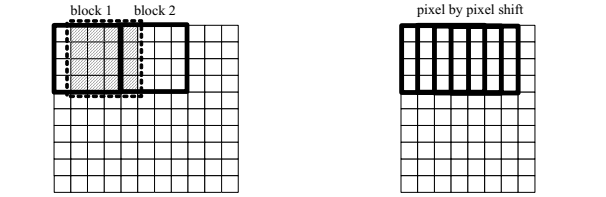


(b) ME operation in the second cycle

Fig. 9. Operation flow of the proposed motion estimation

The proposed ME architecture supports the [+16, -15] search window. In the [+16, -15] search window, 32 4 x 4 blocks exist in a row. In the first cycle, four SADs are simultaneously calculated as shown in Fig. 9(a). Next, the search window shifts left and each operation unit repeats the SAD calculation as shown in Fig. 9(b). SADs of upper four pixels of every block in a row can be obtained after 8 cycles and 32 SADs are stored in buffers. SADs of the second upper are calculated in the same way, and 32 SADs are accumulated with the 32 SADs in buffers, respectively. Then, after 32 cycles, 32 SADs of 4 x 4 blocks can be obtained.

Fig. 10 shows the ME computation flows of existing architectures and the proposed architecture. In Fig. 10(a), the pixel values in the dotted block should be fetched again to calculate SAD of the dotted block after SADs of two adjacent blocks (block 1 and block 2) are obtained. However, if a 4 x 4 block is shifted pixel by pixel as shown in Fig. 10(b), the data in the dotted block in Fig. 10(a) can be reused.



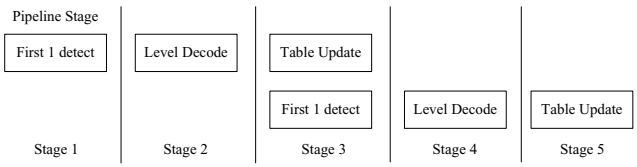
(a) Existing computation flow (b) Proposed computation flow

Fig. 10. ME computation flow

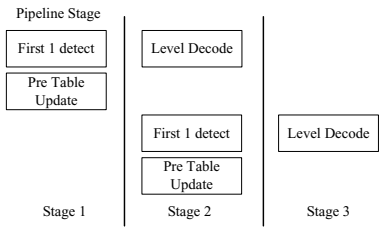
F. Proposed hardware accelerator for entropy coding

The encoder of CAVLC finds the value of the codeword and the length of the codeword in memory according to the data. Therefore, the efficient memory address generator is needed. The decoder of CAVLC is usually implemented with a lookup table. In the decoding process, the level of the nonzero coefficient decoding is an iterative method, which can be implemented without a lookup table.

A generic decoding process for the level of the nonzero coefficient is as follows. First, the decoder obtains the number of successive ones in the input bit stream. Next, the decoder calculates the current symbol length and decodes the current symbol. Finally, the decoder updates the table information used for next symbol decoding. The decoder cannot decode the next symbol until the table information is decided. The generic level of the nonzero coefficient decoding process is shown in Fig. 11(a). Table updating is decided whether the current symbol value is more than the threshold value. Since each table's threshold value has a regular form, we can update the table before current symbol decoding. Hence, the Level Decode stage in the current symbol and the First 1 detect stage in the next symbol can be executed in parallel. Fig. 11(b) shows the proposed level of the nonzero coefficient decoding process. As you can see, we can reduce the computation cycles for level decoding.



(a) Generic level of the nonzero coefficient decoding flow



(b) Proposed level of the nonzero coefficient decoding flow

Fig. 11. Comparison of flows for the level of the nonzero coefficient decoding

If a ROM based look up table is used to implement the runs of zero decoder, the area cost of the table will be expensive. Hence, the ZTEBA (Zero-left Table Elimination by Arithmetic) method was introduced by Chang [10]. In our design, we improve the ZTEBA method. We eliminate the SUB unit and the Saturation unit used in [10] by adding some multiplexers and wires. Fig. 12 shows the block diagram for the proposed runs of zero decoder.

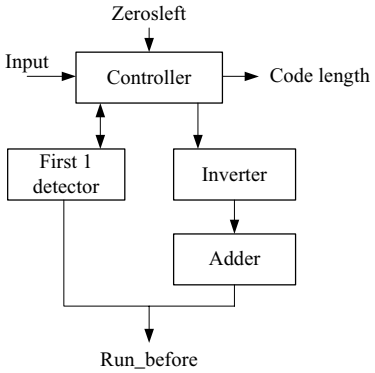


Fig. 12. Proposed runs of zero decoder block diagram

IV. PERFORMANCE COMPARISONS

H.264/AVC can be implemented using ASIP and hardware accelerators. Several core blocks for generating an intra predictor and an in-loop deblocking filter are coded using the proposed application specific instructions and the same blocks are also coded using the existing instructions of TMS320c64x. The proposed architecture can reduce the number of clock cycles for several core blocks for generating an intra predictor about 40% than TMS320c6x. Moreover, the total number of clock cycles to execute the in-loop deblocking filter can be reduced about 20 ~ 25% than TMS320c6x. 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 4 x 4 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 16 x 16 macro blocks. Table I shows the number of the required instructions for 30 frames on existing DSPs [14] [17] and the proposed ASIP. The proposed ASIP having the special instructions can be more efficient than the implementation using instructions of TMS320c55x (SW) and using a coprocessor of TMS320c55x (HW) for the integer transform. TMS320c64x has a VLIW architecture and has eight function units while the proposed ASIP architecture requires only two 32 bit adders.

Table I. Performance comparisons of the 4 x 4 integer transform

	TMS320c55x (SW) [17]	TMS320c55x (HW) [17]	TMS320c64x [14]	**Proposed ASIP**
MIPS	12.8	2.8	1.1	**1.1**

The proposed hardware accelerators have been modeled by Verilog HDL and synthesized using the Samsung SEC 0.18 ㎛ standard cell library by the Synopsys Design Compiler. The proposed ME accelerator can significantly reduce the gate count and the required computation cycles compared with the Samsung architecture [18] and can support much larger search range than the Amphion architecture [19]. Table II shows the comparisons among Samsung, Amphion and our architecture. The Samsung architecture has 64 Processing Elements (PEs) and can support much larger search range than the other architectures. However, it requires much larger computation cycles than the other architectures. The total gate count and the required computation cycles of Amphion are comparable with our architecture. However, the search range of Amphion is much smaller than our architecture.

The proposed hardware accelerator for CAVLC takes average 368 clock cycles for a macro block. In order to achieve the real-time processing requirement for H.264/AVC decoding with HD1080i format, the proposed design should run over 90 MHz. The proposed design can support real-time processing since the maximum operating frequency of the proposed design is about 130 MHz.

Table II. Performance comparisons of the hardware ME architectures

	Clock cycles / frame	Search range	Supported block size	Gate counts
Samsung [18]	12,103,740	H :[-64, +63] V :[-32, +31]	Variable block support	64 PEs
Amphion [19]	406,077	[-8, +7]	Variable block support	61K
Proposed architecture	**456,192**	**[-16, +15]**	**Variable block support**	**76K**

Fig. 13 shows computation times of DSP, ASIC, and ASIP implementations according to the profiling results. Fig. 13(a) shows the computation times of the DSP implementation. If we assume that a single core is used to implement the H.264/AVC algorithm, DSP serially executes all of the algorithm blocks.

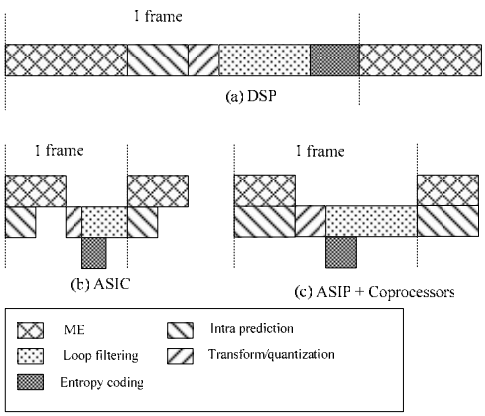


Fig. 13. Computation times of various implementations

Fig. 13(b) shows the computation times of the ASIC implementation. Each block is executed using the dedicated hardware. However, all of the blocks cannot be executed in parallel, since the ME block needs the reconstructed results of the previous frame and the transform block uses ME results. Fig. 13(c) shows the ASIP having accelerators implementation. ME/MC and entropy coding are implemented using the accelerators. The ASIP having accelerators implementation requires more computation times than the ASIC implementation. However, it requires much less computation times than DSP and can support various profiles and standards.

V. Conclusions

This paper has presented efficient instructions to implement the in-loop deblocking filter, intra prediction and integer transform of H.264/AVC. This paper has proposed the hardware accelerators for ME/MC and entropy coding. Three hadd instructions can execute various packed additions within a register. Performance comparisons have shown that the number of clock cycles can be reduced about 20 ~ 25% compared with the existing DSP for the in-loop deblocking filter. The fTRAN and iTRAN instructions can perform the 4 x 4 integer transform in 12 clock cycles. The integer transform can be implemented using much smaller hardware size compared with existing DSPs. The proposed hardware accelerators can efficiently perform ME/MC and entropy coding of H.264/AVC and require minimal hardware. Since the proposed ASIP having the hardware accelerators can concurrently operate, it can handle real-time video processing and can support various multimedia algorithms.

Acknowledgements

This work was supported in part by the NRL (National Research laboratory) program of MOST (Ministry of Science & Technology), in part by the ITSOC program of MIC (Ministry of Information and Communication), and in part by IDEC.

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.,* Sept. 2001, pp. 80-90,.

[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.,* 2000, pp. 181-199.

[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. Circuit Syst.,* May 2004.

[4] Suk Hyun Yoon, Jong Ha Moon, and Myung Hoon Sunwoo, “Efficient DSP Architecture for High-Quality Audio Algorithms,” in *Proc. IEEE Int. Symp. Circuits Syst.,* May 2005

[5] Kim, S.D., Lee, J.H, Yang, J.M., Sunwoo, M.H., and Oh, S.K., “Novel Instructions and Their Hardware Architecture for Video Signal Processing,” in *Proc. IEEE Int. Symp. Circuits Syst.,* May 2005

[6] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14496-10 (E) AVC). July, 2004.

[7] J. Ostermann, T. Wedi, et al.,“Video coding with H.264/AVC: tools, performance, and complexity,” *IEEE Circuits and Systems Magazine,* vol. 4, pp. 7-28, 2004.

[8] Wu Di, Gao Wen, Hu Mingzeng and Ji Zhenzhou, “An Exp-Golomb encoder and decoder architecture for JVT/AVS,” in *Proc. 5th International Conference on ASIC,* 21-24 Oct. 2003 vol. 2, pp. 910-913.

[9] Gisle Bjontcgaard and Karl Lillcvold, “Context-adaptive VLC (CAVLC) coding of coefficients,” Doc. JVT-028, JVT of IS0/IEC MPEG & ITU-T VCEG 3rd Meeting, Virginia, USA, May. 2002.

[10] Hsiu-Cheng Chang, Chien-Chang Lin, and Jiun-In Guo, “A Novel Low-Cost High-Performance VLSI Architecture for MPEG-4 AVC/H.264 CAVLC Decoding,” in *Proc. IEEE Int. Symp. Circuits Syst.,* May 2005.

[11] Yeong-Kang Lai, Chih-Chung Chou, and Yu-Chieh Chung, “A simple and cost effective video encoder with memory-reducing CAVLC,” in *Proc. IEEE Int. Symp. Circuits Syst.,* May 2005.

[12] Woong IL Choi, Byeungwoo Jeon and Jechang Jeong, “Fast motion estimation with modified diamond search for variable motion block sizes,” in *Proc. International Conference on Image Processing,* Sept. 2003, vol. 3, pp. 14-17.

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

[14] *TMS320C64x Image/Video Processing Library,* Texas Instruments Inc., Dallas, TX, 2003.

[15] *BlackfinTM DSP Instruction Set Reference,* Analog Device Inc., Norwood, Mass. 2002.

[16] Thomas Wiegand, Xiaozheng Zhang, and Bernd Girod, “Long-Term Memory Motion-Compensated Prediction,” *Trans. Circuit Syst. Video Technol.,* vol. 9, no. 1, pp. 70-84, Feb. 1999.

[17] *TMS320C55x Hardware Extensions for Image/Video Applications Programmer’s Reference,* Texas Instruments Inc., Dallas, TX, 2002

[18] Jae H. Lee and Nam S. Lee, “Variable Block Size Motion Estimation Algorithm and Its Hardware Architecture for H.264/AVC,” in *Proc. IEEE Int. Symp. Circuits Syst.,* May 2004.

[19] Swee Yeow Yap and john V. McCanny, “A VLSI Architecture for Variable Block Size Video Motion Estimation,” *Trans. Circuit Syst. Video Technol.,* vol. 51, no. 7, July 2004.