HARDWARE-SOFTWARE CO-IMPLEMENTATION OF A H.263 VIDEO CODEC

S. K. Jang, S. D. Kim, J. Lee, G. Y. Choi and J. B. Ha

Department of Electrical Engineering, Korea Advanced Institute of Science and Technology
373-1 Kusongdong, Yusonggu, Taejon, Republic of Korea
E-mail: skjang@isserver.kaist.ac.kr, jbra@ee.kaist.ac.kr

Samsung Electronics Company Ltd., Korea

ABSTRACT—In this paper, an H.263 video codec is implemented by adopting the concept of hardware and software co-design. Each module of the codec is investigated to find which approach between hardware and software is better to achieve real-time processing speed as well as flexibility. The hardware portion includes motion-related engines, such as motion estimation and compensation, and a memory control part. The remaining portion of the H.263 video codec is implemented in software using a RISC processor. This paper also introduces efficient design methods for hardware and software modules. In hardware, an area-efficient architecture for the motion estimator of a multi-resolution block matching algorithm using multiple candidates and spatial correlation in motion vector fields (MRMC), is suggested to reduce the chip size. Software optimization techniques are also explored by using the statistics of transformed coefficients and the minimum sum of absolute difference (SAD) obtained from the motion estimator.

1. Introduction

As multimedia environment evolved, increasing attention has been drawn to the processing of digital video sequences over the last few years. Recently, H.263 has been standardized by the ITU [1], in order to support videophone, videophone, surveillance camera, and other low bit rate applications. Since video compression algorithms are computationally very intensive and require a large amount of memory I/O operations to reduce intrinsic redundancies of video sequences, they have been implemented by embedded hardware rather than software [2]. But embedded hardware implementation has a limitation on flexibility for meeting the various requirements of wide applications.

As VLSI technologies advances, the processing power of a general-purpose processor increases dramatically. So real-time video processing applications tend to be implemented by software design nowadays with the help of a powerful processor. However, this cannot be the complete solution to fulfill the real-time requirement fully. Consequently, parallel processing and multimedia instruction set extension are generally adopted to enhance the computing power for real-time applications [3,4].

However, there exists a trade-off between hardware and software implementation. Various factors such as flexibility, development cost, power consumption, and processing speed requirement should be taken into account. Hardware implementation is generally better than software implementation in processing speed and power consumption. In contrast, software can give a more flexible design solution and also be more suitable for various video applications.

In order to take advantages of both software and hardware implementation fully, each functional module of the H.263 video codec is studied to determine a proper way for hardware-software partitioning. Based on this study, motion estimation (ME) and motion compensation (MC) parts are implemented in hardware, and the remaining parts are performed in software. A noticeable feature of software-hardware co-design is also found in the cooperation of software and hardware modules. In our system, the ME engine is designed to obtain not only macroblock-based SADs but also block-based SADs. And this SAD information can be efficiently utilized to drastically reduce the computational complexity for software jobs such as discrete cosine transform (DCT) and variable length coding (VLC) by skipping or pruning the DCT. MC is also designed by hardware to apply for both encoding and decoding in a manner of time-sharing. The designed hardware ME and MC engines work concurrently with a RISC processor performing software jobs. The engines communicate with the RISC processor via interrupts and special commands such as start, abort, stop, and resume, and they co-work on a simple hardware-software scheduling policy.

This paper is organized as follows. In Section II, we describe the overview of the implemented systems and consider a hardware-software partitioning. In Section III, efficient small area architectures of hardware modules are described. In Section IV, we investigate software optimization methods by exploiting the statistics of processed data at each part of H.263 codec. The hardwaresoftware scheduling problem is described for practical implementation in Section V, and the verification of the proposed system on an emulation board are given in Section VI. Finally, we draw the conclusions in Section VII.
II. System Architecture

A. System Overview

Fig. 1 shows the overall video communication system. The implemented system consists of a general purpose RISC processor, a unified memory module, and a minimal auxiliary hardware. The portion that the processor reserves for the H.263 video codec is roughly 30% - 40% of the total computing power. All the other parts of the H.324 system, such as G.723, H.263, and H.264, are done in software.

Fig. 1. Architecture of a video communication system.

The software and hardware co-optimization approach is taken to meet both the strict real-time processing requirement and efficient and flexible implementation. The optimum area reduction of the hardware modules and the processing time reduction in the software part are major issues in this paper. Except for a small size SRAM dedicated for the motion estimator, unified memory SDRAM is adopted to store video, audio, and miscellaneous data required for the H.324 system. The hardware modules include motion estimation and compensation, SDRAM controller, host interface, and arbiter. All these modules are implemented into an application specific integrated circuit (ASIC). The RISC core communicates with hardware engines via interrupts, and hardware engines have status registers controllable in the RISC core. The arbiter controls the overall flow in data bus. The SDRAM controller is responsible for data input and output and other necessary operations for the SDRAM.

B. Hardware-Software Partitioning

For hardware-software co-design, software-only codec is implemented first and the bottleneck of processing speed requirement is detoured by embedded hardware modules. Our system aims at low bitrate applications and its target bit rate is about 64kbps for QCIF sequences. In order to decide the hardware implementation of motion estimation and compensation, the following observations are considered. In the full software implementation of the H.263 video codec, it has been known that 120-140 MOPS are roughly required for the target bitrate of 64 kbps of QCIF sequences (with 30 pictures/sec, pre-filtering, 1-7 motion search range, and 3-stage motion search technique) [3]. However, the percentage of usable computing power for the H.263 codec is only 30-40% out of the given RISC computing power (approximately 200 MOPS). Hence, complete software jobs have to be done within 60-80 MOPS. Therefore, at least 60 MOPS should be assigned to hardware in order to meet the real-time requirement of the system.

Since it has been known that motion estimation and compensation are the most computationally intensive among basic functional units for H.263, they are better off to be designed in hardware to meet the real-time requirement. In addition, the standalone implementation of motion-related hardware engines is meaningful in itself, because they are suitable for most of video compression standards without any major modification. In contrast, variable length coding and decoding are recommended for the software approach, because they have various forms depending on video compression standards and have irregular structure not suitable for implementation in hardware.

Even though DCT and inverse DCT (IDCT) are not as complex as motion related engines, they are still major complex parts in an H.263 codec. Therefore, hardware implementation has been preferred for these modules. However, there is still a good chance to reduce the complexity of DCT and IDCT in software utilizing the characteristics of the processed data at low bitrates. Therefore, in this paper, we try to optimize DCT and IDCT in software rather than in hardware.

III. Optimization Methods in Hardware

A. Optimization in Motion Estimation

In hardware optimization, area reduction of the motion estimator is the most important task. Instead of using a full search block matching algorithm (FSBMA), the 3-level multi-resolution block matching algorithm using multiple candidates and spatial correlation in motion vector (MV) fields, called MRMC5-3 (which is almost the same as the HSMA35S in [5]), is adopted. Since MRMC5-3 has much less computational complexity than FSBMA without degrading the estimation performance, its hardware implementation requires a much smaller number of gates. Table I shows performance comparisons between MRMC5-3 and FSBMA.

Table I. Performance comparison between FSBMA and
MRMCS-3 for various QCIF video sequences (64 kbps, 30 Hz, 300 frames).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall monitor</td>
<td>7</td>
<td>73.45</td>
<td>68.85</td>
<td>35.47</td>
<td>68.75</td>
</tr>
<tr>
<td>Container ship</td>
<td>7</td>
<td>74.47</td>
<td>71.74</td>
<td>34.47</td>
<td>71.55</td>
</tr>
<tr>
<td>Mother and daughter</td>
<td>7</td>
<td>73.98</td>
<td>56.23</td>
<td>36.00</td>
<td>56.31</td>
</tr>
<tr>
<td>Foreman</td>
<td>17</td>
<td>29.67</td>
<td>64.35</td>
<td>29.50</td>
<td>65.44</td>
</tr>
<tr>
<td>Coast guard</td>
<td>17</td>
<td>28.27</td>
<td>65.43</td>
<td>28.24</td>
<td>65.58</td>
</tr>
<tr>
<td>Silent voice</td>
<td>9</td>
<td>32.70</td>
<td>67.29</td>
<td>32.74</td>
<td>67.66</td>
</tr>
</tbody>
</table>

As shown in Fig. 2, MRMCS-3 performs a motion search for each 4×4 block with a search range of ±4 on the subsampled image in the upper level, and two search points corresponding to minimum SADs are selected as initial search points in the middle level. Besides these two points from the upper level, the median value of lower level MVs of blocks adjacent to the current block is adopted as an additional initial search point. After performing a local search for each initial search point with a search range of ±2 in the middle level, only one point corresponding to the minimum SAD is selected as the initial search point in the lower level. Finally, in the lower level, MRMCS-3 performs a local search with a range of ±2 on the original image to obtain the final MV.

Fig. 2. Conceptual diagram of MRMCS-3.

For the hardware implementation of MRMCS-3, we adopt an architecture based on a systolic array that is the most popular and superior architecture for implementing a motion estimator in hardware. In a systolic array architecture, the number of processing elements (PE) is proportional to the macroblock size and the search range.

Therefore, a large number of PEs of 64 or 256, is needed to implement a PSBMA in hardware. However, to implement MRMCS-3 in hardware, we adopt a basic searching unit (BSU) that consists of only 3 PEs, flip-flops, multiplexers (MUXs), and simple logic circuits for data flow control. The BSU, shown in Fig. 3(a), searches a 4×4 block, the smallest block size among the three levels, with a search range of ±2 (25 search points), the smallest search range among the three levels.

Since MRMCS-3 has three multi-resolution search levels, and each level has a different block size and search range, three different systolic arrays are required. In the proposed architecture, however, a remarkable reduction in the number of PEs is achieved by using a BSU repeatedly. In the upper level, by dividing a search region of ±4 (81 search points) into four regions containing 25 search points, a search for a 4×4 block can be completed by using a BSU four times. In the middle level, an 8×8 block is divided into four 4×4 sub-blocks. Then, by using the BSU twelve times, a search can be completed for three initial search points. Similarly, in the lower level, a 16×16 block is divided into sixteen 4×4 sub-blocks. Then, by using the BSU sixteen times, a search for the lower level can be completed. For the middle and lower levels, SADs calculated from the BSU are not of the appropriate block size but are partial SADs from 4×4 sub-blocks. Therefore, a calculation of the
real SAD from the partial SADs is needed. For this operation, both a data buffer and adder are required as functional blocks. Fig. 3(b) shows an overall block diagram of the motion estimator described above. It should be noted that the hardware motion estimator easily calculates block SADs (8×8), which are very useful in optimizing software calculation, as well as a macroblock SAD (16×16).

B. Optimization in Motion Compensation

Motion compensation is another computationally intensive part besides motion estimation. Though the algorithm of motion compensation has a fixed form, a gate count reduction is still possible at the architectural viewpoint.

The functional operations of motion compensation are direct memory access (DMA) and interpolation. To cut down on the gate count, we use only one interpolator for interpolation. Once motion compensation starts, 17 or 16 pixels are loaded on the temporal registers for the luminance frame, and 9 or 8 pixels are loaded for the chrominance frame depending on the existence of horizontal half-pel components of the motion vector. If the given motion vector has no vertical half-pel component, the loaded pixels are interpolated along the horizontal direction. Otherwise, pellets from the two lines (one is the currently loaded line and the other is the previously loaded line) are interpolated along the horizontal and vertical directions according to the existence of half-pel components.

By time-sharing, the motion compensation engine is used for both encoding and decoding with different frame memory addresses.

IV. Optimization Methods in Software

There exist two categories in software optimization; execution time and code size (memory size). Since the main concern of H.264 system is real-time application, we regard timing optimization more important than code size optimization. The timing optimization may have four phases; algorithm level optimization, high level language optimization, low level language optimization, and system dependent optimization. Above all, the algorithm level optimization provides a crucial impact on the reduction of computational complexity. Especially, DCT and quantization (DCT/Q) have major computational complexity in software parts of an H.264 codec. It is known that DCT/Q are performed much more frequently than inverse quantization and inverse discrete cosine transform (IQIDCT) in the low bit rate environment. So we adopt an EOB estimation scheme to skip or prune DCT/Q to alleviate their calculation. In EOB estimation, the block-based SAD values obtained from the hardware ME are utilized.

Fig. 4. Histogram of the minimum SADs for coded blocks and not-coded blocks. The data is obtained from the Foreman QCIF sequence (300 frames, 30Hz, Fixed QP). Only luminance inter-blocks are counted. (a) QP=10, (b) QP=20, and (c) QP=30.

A. DCT/Q Calculation Reduction with Predictive Decision of Not-Coded Block

Since DCT has been adopted in many international standards for image and video compression, there have been many research activities to reduce the number of
Fig. 5. Prediction results of not-coded inter-blocks before DCT and quantization. Numbers on the horizontal axis denote OP, and a black bar represents the portion which is predicted as not-coded blocks. (a) Foreman, (b) Coast guard, (c) News, and (d) Silent voice (QCIF, 300 frames, and 30Hz).
operations for DCT and IDCT in the algorithm level. Recently DCT and IDCT optimization methods based on input statistics of zero transformed coefficients were also suggested [6-8]. Since the H.263 codec is working in a low bitrate environment and inter-frame coding is mostly used, DCT coefficients have many zeros compared to still image compression such as JPEG. Therefore, statistics of transformed coefficients are useful for reducing computation time.

When a video sequence is compressed with a bitrate lower than 64 kbps, there exist many not-coded blocks in the H.263 bit stream. In the encoder, since DCT and quantization have to be processed prior to VLC, all blocks must be DCT-transformed and quantized even though such a process has no meaning for not-coded blocks. In order to utilize this phenomenon for improving the encoding speed, the existing DCT/Q skipping method is adopted in inter-frame encoding [6].

The minimum SAD of a given block can be a good prediction criterion for DCT coefficient behaviors [9,10]. In inter-blocks, we can predict a not-coded block by comparing the minimum SAD value with a threshold value related to the quantization parameter. It should be noted that block-based SAD values are available from a motion estimator implemented in hardware. Our specific architecture of a motion estimation engine, which utilizes the RBU repeatedly, facilitates calculating block-based SADs as well as macroblock-based SADs without additional costs. The smaller the minimum SAD of a block is, the higher its possibility of being decided as a not-coded block.

Fig. 4 shows the relationship between the minimum SAD and the number of not-coded blocks for luminance inter-macroblocks. In this figure, we can see that even a simple thresholding based on the experiment can considerably improve the processing speed considerably, because SAD histograms for coded blocks and not-coded blocks have good shapes appropriate for finding a threshold value. We have used $20 \cdot QP$ as the threshold and obtained the miss-prediction probability of lower than about 1% for various test sequences. In other words, we assume that DC and all AC coefficients are quantized to zero if the SAD value is smaller than $20 \cdot QP$. In this case, the corresponding inter-block is treated as a not-coded one and many complex operations such as DCT and quantization, are skipped. The DCT/Q skipping method is also applied to chrominance blocks. Since SAD values of chrominance blocks are to be obtained in software, an additional computational burden is needed. However, the saving of DCT/Q provides much more benefits because most of the blocks are determined as not-coded blocks. Fig. 5 shows the portion of coded interblocks and not-coded inter-blocks. The graphs tell how many inter-blocks are estimated into not-coded ones. As shown in the figure, among all not-coded inter-blocks, half of them are predicted and DCT/Q may be skipped. The skipping ratio is much higher in chrominance inter-blocks than in luminance ones.

### B. Calculation Reduction by Predictive EOB Estimation

DCT/Q calculation may be further reduced by pruning the DCT coefficients. Quantized DCT coefficients generally have many zeros at low bit rates. If we can predict the final position of nonzero quantized DCT coefficients in zigzag scanning in advance, we can reduce DCT/Q calculation further [11-12]. If the estimated maximum EOB lies within the BC rectangle, only the values of corresponding low frequency DCT coefficients are to be calculated, and the high frequency terms are regarded as zeros after quantization.

![Classification of an 8x8 transformed and quantized block.](image)

Fig. 6 shows the classification of a quantized block. For example, if all non-zero coefficients belong to the upper-left $4 \times 4$ rectangular region after quantization, the block is classified to class 8. In this paper, we utilize the SAD value as a reference for class prediction criteria. Table II shows the pruning criteria obtained from experiments so that the degradation of coding performance may be negligible. Fig. 7 shows the overall optimized DCT flow chart according to each class.

Once the class of a quantized block is predicted, a faster processing speed is achieved because the operations of DCT, quantization, and VLC do not need to be applied to all 64 pixels. By using the estimated maximum EOB, dividing operations can be reduced in quantization, and a scanning operation for determining the real EOB can be sk-
Table II. Class prediction criterion for a quantized block.

<table>
<thead>
<tr>
<th>Class</th>
<th>Effective DCT kernel size</th>
<th>Luminance</th>
<th>Chrominance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0</td>
<td>No DCT</td>
<td>$SAD &lt; 20-QP$</td>
<td>$SAD &lt; 20-QP$</td>
</tr>
<tr>
<td>Class 1-3</td>
<td>$3 \times 3$</td>
<td>$20-QP \leq SAD &lt; 21-QP$</td>
<td>$20-QP \leq SAD &lt; 22-QP$</td>
</tr>
<tr>
<td>Class 4</td>
<td>$4 \times 4$</td>
<td>$21-QP \leq SAD &lt; 22-QP$</td>
<td>$22-QP \leq SAD &lt; 24-QP$</td>
</tr>
<tr>
<td>Class 5</td>
<td>$5 \times 5$</td>
<td>$22-QP \leq SAD &lt; 23-QP$</td>
<td>$24-QP \leq SAD &lt; 26-QP$</td>
</tr>
<tr>
<td>Class 6</td>
<td>$6 \times 6$</td>
<td>$23-QP \leq SAD &lt; 26-QP$</td>
<td>$26-QP \leq SAD &lt; 30-QP$</td>
</tr>
<tr>
<td>Class 7</td>
<td>$7 \times 7$</td>
<td>$26-QP \leq SAD &lt; 32-QP$</td>
<td>$30-QP \leq SAD &lt; 56-QP$</td>
</tr>
<tr>
<td>Class 8</td>
<td>$8 \times 8$</td>
<td>$32-QP \leq SAD$</td>
<td>$56-QP \leq SAD$</td>
</tr>
</tbody>
</table>

ipped in VLC. To avoid computational burden of the dividing operation, many dividing operations for quantization are replaced with comparing operations before quantization, because most quantized coefficients are determined as zero after quantization.

![Diagram](image1)

Fig. 7. DCT optimization algorithm.

C. Optimization of IQ/DCT

By exploiting the characteristics of input coefficients, we can reduce the IDCT/IQ calculation complexity [6]. In this paper, we use a fast IDCT algorithm with a butterfly architecture to skip the local signal flow of IDCT when the given coefficients are zero. Therefore, the actual IDCT complexity depends on the probability of zero DCT coefficients, the additional complexity for zero tests, and the complexity of local butterfly calculation. It should be noted that the additional complexity to determine how many non-zero coefficients exist is not considerable, because the zero coefficient information can be obtained in the process of variable length decoding (VLD) with a negligible number of operations.

![Diagram](image2)

Fig. 8. Distribution of EOB values for the Foreman QCIF sequence (fixed QP, 30Hz, 300 frames). EOB of 0 means that the block is not coded.

We have examined the computation reduction effect resulting from zero coefficient tests through simulation on the ARM (advanced RISC machine) platform. Compared to the case without a zero coefficient test, about a 32% reduction of the total IDCT complexity is achieved through zero tests on every row and column at I-D IDCT. Moreover, if EOB is less than 4, we adopt simplified IDCT schemes. Fig. 8 shows EOB statistics for the Foreman QCIF sequence. As shown in the figure, many blocks are
not coded and about 20%-50% of the coded blocks have an EOB smaller than 3. Therefore, the use of simplified IDCT schemes can additionally reduce the IDCT complexity. Fig. 9 shows the overall optimized IDCT flow chart.

![Figure 9. IDCT optimization algorithm.](Image)

**V. Hardware-Software Scheduling of the Video Codec**

The well-defined concurrent operation of hardware and software is another important issue. Thus a simple but reasonable scheduler is prepared to allocate hardware and software jobs properly for video codece. This scheduler controls the operation of MB/MC modules by sending commands such as start, abort, stop, and resume through specialized command registers. For example, if the scheduler writes an ‘ME start’ command to a predefined register to start motion estimation, the motion estimator continues its job for k macroblocks without interruption, and then an interrupt is transmitted to the scheduler after the job is completed. Here, the value of k is specified at the operation unit counter. This feature gives more flexibility so that the scheduler can select a desired amount of hardware job for each term.

To functionally verify hardware and software co-working, scheduling mechanisms for encoding and decoding are independently considered in terms of the processing flow. In the encoder processing flow, a motion vector should be obtained from the motion estimator before performing motion compensation for the corresponding macroblock. Similarly, software jobs, such as DCT and quantization, can not be performed until motion compensation is completed for the corresponding macroblock. In order to meet these conditions, five special purpose registers are used for implementing the encoder scheduling: MBHME, MBHMC, FlagMEBusy, FlagMCBusy, and MBIS. MBHME and MBHMC contain the indices of macroblocks at which the jobs of the hardware ME and MC are completed, respectively. FlagMEBusy and FlagMCBusy show the states of the hardware ME and MC, respectively. MBIS denotes the next macroblock operation index from which the software jobs begin. Fig. 10 shows a basic scheduling diagram for encoding (k = 1). According to this diagram, the hardware and software jobs are concurrently performed as desired. It is also noted that the waiting time of software parts depends on the processing time of the hardware modules. The scheduling of decoding can be achieved in a similar way to that of encoding.

![Figure 10. Hardware-software scheduling mechanism for the implemented encoder.](Image)
VI. Implementation Results

We have implemented the video codec on a reconfigurable emulation board called Aaptis, in order to test and debug it. The emulation board can be configured to an arbitrary logic and has programmable routing devices. On the board, we place a Strong ARM 110 core and memory devices, and prepare two FPGAs (field programmable gate array) devices for hardware implementation. A motion estimator and compensator, SDRAM controller, host interface, DMA engines, and video pre/post-processing parts are implemented in these FPGAs. It should be mentioned that the hardware in FPGAs can be easily implemented in ASIC by using the standard cell library with a minor update of delay characteristics. For QCIF sequences, the ME core excluding DMA requires an 11 MHz clock for the encoding at the frame rate of 30 Hz. The MC requires a 9.5 MHz clock for both the encoding and decoding at the frame rate of 30 Hz, when the probability that a motion vector has half-pel components is assumed to be 0.5. The gate numbers required for the ME and MC hardware are about 25,000 and 15,000, respectively.

In the emulation board, the delay mainly caused by programmable routing devices limits the practical operating frequency to 10 MHz. Due to this intrinsic delay, we limit the memory and system clock of the evaluation system to about 9.5 MHz, and set the corresponding processor core clock to 86 MHz. Hence, through the implementation on the emulation board, we can verify only the functions for the hardware-software co-designed video codec, and may predict its real-time operation in the actual system. It should be noted that in our actual target system, the processor core and memory clocks will be set to 200 MHz and 66 MHz (rather than 86 MHz and 9.5 MHz), respectively.

Table III is the profiling results of the video encoder implemented in the emulation board for two different video scenes captured from a camera. In this table, we use a fixed quantization parameter (QP) of 10, and encode 100 frames in the baseline mode. Since the ME and MC hardware modules (designed to operate at the 66 MHz clock) are working at the slow clock speed of 9.5 MHz in the emulation system, the scheduler consumes a relatively large portion of the total execution time while waiting for the termination of hardware operations before starting software operations. It should also be noted that the processing time of software becomes longer as the input scene has more complex motion. This is because the number of coding blocks increases drastically, and consequentially, the software computing time for DCT, quantization, VLC, inverse quantization, and IDCT increases. The function of the decoder can also be verified in a similar way as in the encoder, with the hardware motion compensator.

Thus, by using 100% of the processor computing power, the maximum attainable frame rate will be about 29 Hz (± 100 frames/3.5 sec) and 16 Hz (± 100 frames/6.3 sec) when disregarding hardware operation. If the processor core clock and system clock are set to 200 MHz and 66 MHz in the target system, respectively, the implemented video encoder is expected to operate about 2.3 ~ 7.0 times faster. When considering that the memory bandwidth exhaustion is the main bottleneck of the processing speed in the emulation system, the speed improvement is more closely related to the system clock speed rather than the processor core clock speed. Therefore, based on the experimental results from the emulation system, it is expected that a 30 Hz encoding of QCIF sequences can be carried out sufficiently with 30% of the total computational power of a RISC processor, in the hardware-software co-implemented H.263 video codec.

<table>
<thead>
<tr>
<th>Functions</th>
<th>Scene 1</th>
<th>Scene 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color space conversion</td>
<td>5.45</td>
<td>37.33</td>
</tr>
<tr>
<td>(RGB 4:2:2 to YUV 4:2:0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Waiting time of software</td>
<td>5.67</td>
<td>38.81</td>
</tr>
<tr>
<td>for ME/MC operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SubBlock</td>
<td>0.64</td>
<td>4.41</td>
</tr>
<tr>
<td>DCT</td>
<td>0.94</td>
<td>6.40</td>
</tr>
<tr>
<td>Quant</td>
<td>0.25</td>
<td>1.71</td>
</tr>
<tr>
<td>VLC</td>
<td>0.33</td>
<td>2.28</td>
</tr>
<tr>
<td>DeQuant</td>
<td>0.03</td>
<td>0.22</td>
</tr>
<tr>
<td>IDCT</td>
<td>0.16</td>
<td>1.12</td>
</tr>
<tr>
<td>AddBlock</td>
<td>0.10</td>
<td>0.68</td>
</tr>
<tr>
<td>Misc.</td>
<td>1.01</td>
<td>6.91</td>
</tr>
<tr>
<td>Total time [sec]</td>
<td>14.60</td>
<td></td>
</tr>
<tr>
<td>Encoded frames/sec [Hz]</td>
<td>6.97</td>
<td></td>
</tr>
<tr>
<td>Measured bitrate [kHz/sec]</td>
<td>11.12</td>
<td></td>
</tr>
</tbody>
</table>

VII. Conclusions

In this paper, we introduce an implementation method for an H.263 video codec based on hardware and software co-design. In order to maximally utilize the RISC computing power reserved for the H.263 codec and to achieve efficient
hardware-software partitioning, motion estimation and compensation parts are implemented in hardware. Novel design methods of ME/MC parts are introduced to minimize the corresponding hardware area. In software optimization, several methods are investigated to reduce the computational complexity mainly for DCT/IDCT by utilizing the SAD information obtained from the hardware motion estimator. The hardware-software co-implementation of H.263 video codec is expected to use 30%-40% of computational power of the 200 MHz RISC processor and 40,000 gates. This video codec will be merged into the H.324 video-conferencing system.

Acknowledgement

The authors would like to thank J. S. Kim, U. Jungho, and J. D. Kim of Samsung Electronics Company for their valuable cooperation.

References


Biographies

Sung Kyu Jung received the B.S. degree in electronics engineering from Pusan National University in 1996 and the M.S. degree in electrical engineering from KAIST in 1998. Currently, he is working toward Ph.D. degree in department of electrical engineering from KAIST. His research interests include image/video compression.

Sung Duk Kim received the B.S. degree in electronic engineering from Kyungpook National University in 1994 and the M.S. degree in electrical engineering from KAIST in 1996. Currently, he is working toward Ph.D. degree in department of electrical engineering from KAIST. His research interests include image/video compression.

Jae Hoon Lee received the B.S. and M.S. degrees in electrical engineering from KAIST in 1996 and 1998, respectively. Currently, he is working toward Ph.D. degree in department of electrical engineering from KAIST. His research interests include image/video compression and representation.

Geon Young Choi received the B.S. degree in electronic engineering from Yonsei University in 1995 and the M.S. degree in electrical engineering from KAIST in 1997. Currently, he is working in multimedia laboratory of Samsung Electronics Company Ltd., Korea. His research interests include image processing and its hardware implementation.

Jong Beom Ra received the B.S. degree in electronic engineering in 1975 from Seoul National University, and the M.S. and Ph.D. degrees in electrical engineering from KAIST in 1977 and 1982, respectively. From 1983 to 1987, he served as an associate research scientist in Columbia University in New York. Since 1987, he has been working as a professor in department of electrical engineering, KAIST. His research interests are digital image processing, video signal processing, 3-D visualization, and medical imaging such as MRI.