Design of Parallel Image Compression Circuits for High-speed CMOS Image Sensors

（高速度 CMOS イメージセンサのための並列画像圧縮回路の設計）

Yukinari Nishikawa (member)†, Shoji Kawahito (member)‡‡, Masanori Furuta‡‡, Toshihiro Tamura‡‡

Abstract We investigated parallel image compression circuits suitable for integration in high-speed CMOS image sensors. We compared the coding efficiency and hardware complexity of several image compression algorithms that use 2-D DCTs using simulation and logic synthesis, and found that using 4×4 point 2-D DCT-based coding methods reduced hardware complexity and improved coding efficiency. We developed a parallel processing architecture for on-sensor image compression that uses a processing element array and a data-buffering scheme for parallel data-output. We constructed a prototype 256 × 256 pixel high-speed CMOS image sensor chip that integrates 16 image compression-processing elements and uses 0.25-μm CMOS technology. The area of the image compression circuits is 80% of the image array with 15 μm square pixels. The entire chip could be operated at a clock frequency of 53.6 MHz, and high-speed images compressed by a factor of four could be read out at 10,000 fps using a 32-bit parallel bus.

Key words: High-speed image sensor, Image compression, Parallel processing

1. Introduction

High-speed image sensors are required for various applications, such as the observation of high-speed phenomena, analysis of high-speed machinery, broadcasting, and sports applications. Recently, the performance of high-speed image sensors has been greatly improved by active pixel CMOS image sensor technology1). There are two types of high-speed image sensors, namely, sensors with analog outputs and sensors with digital outputs. At present, high-speed CMOS image sensors with parallel analog outputs are faster than those with digital outputs. However, digital-output high-speed image sensors are needed in order to achieve small camera heads and low-cost camera systems. The bottleneck of digital high-speed image sensors is the data transfer from the sensor chip to an external memory. A solution to this problem is on-sensor image compression, which can be realized in the CMOS image sensor technology. On-sensor image compression is also useful to increase the recording time in memories of limited size because the devices that are currently available for directly storing high-speed images are expensive semiconductor memories.

A few types of image sensor chips with data compression functions have been reported2)3)4). These feature analog domain processing for image compression and are not intended for high-speed data transfer. An image compression system for a high-speed camera using a wavelet compression algorithm has been reported5). This system is implemented with an FPGA and is not considered for on-sensor integration. The authors have reported image compression circuits suitable for high-speed image sensors6)7). However, detailed discussions on appropriate image compression algorithms, the actual architecture, and the design of the on-sensor image compression circuits have not yet been reported.

In the present paper, parallel image compression circuits for high-speed image sensors are discussed with respect to coding efficiency, processing speed, and suitability for on-sensor integration. An on-sensor parallel processing architecture using a processing element array and an output data buffer for parallel data output is proposed. Based on a 4 × 4 point 2-D DCT-based compression algorithm, which is the best suited for the
proposed architecture, a prototype CMOS image sensor integrating column parallel image compression circuits is designed to estimate the performance and the area.

In the following, the image compression algorithms, the design of the processing elements, and the sensor chip design are described.

2. Compression algorithm suitable for high-speed image sensors

Fig. 1 shows the proposed column parallel processing architecture for on-sensor image compression. An array of image compression processing elements (ICPEs) directly processes the image data from the image array. An image compression algorithm to implement the ICPE should meet the following requirements:

1. Small circuit configuration to place the processing element in the limited area of the image array periphery,
2. High data processing speed,
3. Sufficient image coding efficiency.

There are two major image compression methods for moving pictures; intraframe and interframe codings. The interframe coding provides high compression ratio if a sophisticated hybrid coding with motion compensation is used. However, this requires frame memories and complicated circuits. There exist several possible intraframe compression algorithms. Among these, transform coding algorithms using a two-dimensional discrete cosine transform (2-D DCT) are suitable for implementing the parallel image compression architecture shown in Fig. 1 because 2-D DCT-based image compression circuits can be implemented with relatively small circuits while achieving a relatively high coding efficiency. The pixel-block-based algorithm is well suited to parallel processing with the ICPE array. The most suitable algorithm for the parallel image compression system shown in Fig. 1 in terms of the block size of the 2-D DCT and the following entropy coding method remains a topic for discussion. In this section, several 2-D DCT-based image compression methods are compared in order to investigate the most suitable method for the high-speed parallel image compression system.

2.1 Comparison of 2-D DCT based compression methods

The 2-D DCT-based intraframe compression algorithms have been extensively investigated. For example, the mean square error (MSE) of various image transform coding methods as a function of block size are reported in Reference (10). The MSE of the 2-D DCT decreases as the block size decreases, and the coding efficiency of the 4×4 point 2-D DCT alone has been considered to be worse than that of a 2-D DCT having a block size of 8×8 or greater. Therefore, the 4×4 point 2-D DCT has been employed for limited applications, such as adaptive block size transform coding. However, it is worthwhile to investigate the coding efficiency of the 4×4 point 2-D DCT because compact circuits and the resulting high processing speed can be expected based on the parallel image compression architecture of Fig. 1.
In order to explore high coding efficiency in the $4 \times 4$ point 2-D DCT-based compression algorithm, while maintaining the advantage of low hardware complexity, a special zigzag scanning method using four blocks, as shown in Fig. 2, and the application of 1-D Huffman coding and a simplified quantization matrix are introduced. One of the weaknesses of the $4 \times 4$ point 2-D DCT is the relatively short zero run length. The zigzag scanning shown in Fig. 2 can lengthen the zero run length and improve the coding efficiency. A 2-D Huffman coding using the non-zero AC coefficients and the zero run length is considered to be useful for achieving high coding efficiency. However, the table for the 2-D Huffman table is complicated, and this coding is not always suitable for parallel image compression architecture. As a hardware-efficient algorithm, a coding method in which 1-D Huffman tables are used for the DC, the non-zero AC, and the zero run length, is introduced. The Huffman table used for the zero run length in the 1-D Huffman coding is shown in Table 2. The 2-D Huffman table and the table for DC components are the same as those used in JPEG. The Huffman table for non-zero AC components in the 1-D Huffman coding is the same as that used for the DC components.

Another piece of area-consuming hardware is a quantizer for the 2-D DCT coefficients. In the coding method using the $4 \times 4$ point 2-D DCT, a simple quantizer matrix given by

$$Q = q_c \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 2 & 4 & 4 \\ 2 & 2 & 4 & 4 \end{bmatrix}$$

is used, where $q_c$ is the quantization coefficient to control the compression ratio.

In order clarify the advantages of the proposed coding method for parallel image compression circuits using the $4 \times 4$ point 2-D DCT, the zigzag scanning method with four blocks, the 1-D Huffman coding, several coding methods, including the proposed algorithm, are compared with respect to coding efficiency. Table 1 shows five methods using the 2-D DCT-based compression evaluated here. Method D8Z8SH2 is the same algorithm as that used for JPEG. Methods D4Z416H1 and D4Z416H2 in Table 1 employ a special zigzag scanning using four blocks, as shown in Fig. 2. The other methods use a zigzag scanning using a single block. For Huffman coding, the 2-D and 1-D Huffman coding methods are compared.

### Table 1 2-D DCT-based compression methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>DCT</th>
<th>Zigzag scanning</th>
<th>Huffman table</th>
</tr>
</thead>
<tbody>
<tr>
<td>D8Z8SH2</td>
<td>8 x 8</td>
<td>8 x 8</td>
<td>Two-dimensional</td>
</tr>
<tr>
<td>D4Z416H1</td>
<td>4 x 4</td>
<td>4 x 16</td>
<td>One-dimensional</td>
</tr>
<tr>
<td>D4Z416H2</td>
<td>4 x 4</td>
<td>4 x 4</td>
<td>Two-dimensional</td>
</tr>
<tr>
<td>D4Z416H2</td>
<td>4 x 4</td>
<td>4 x 16</td>
<td>Two-dimensional</td>
</tr>
<tr>
<td>D8Z8SH1</td>
<td>8 x 8</td>
<td>8 x 8</td>
<td>One-dimensional</td>
</tr>
</tbody>
</table>

### Table 2 Huffman table for ZRL.

<table>
<thead>
<tr>
<th>ZRL Value</th>
<th>Huffman code</th>
<th>Amplitude Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>2,3</td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>4,...,7</td>
<td>1110</td>
<td>2</td>
</tr>
<tr>
<td>6,...,15</td>
<td>11110</td>
<td>3</td>
</tr>
<tr>
<td>16,...,31</td>
<td>111110</td>
<td>4</td>
</tr>
<tr>
<td>32,...,63</td>
<td>1111110</td>
<td>5</td>
</tr>
</tbody>
</table>

Fig. 3 Test Still Pictures.

### 2.2 Comparison of the compression performance

Fig. 4 shows the simulation results for the five methods listed in Table 1. The compression ratio is controlled by the quantization coefficient. The image quality is measured by the Peak Signal-to-Noise Ratio (PSNR). The test pictures used in the simulation are shown in Fig.3.

Method D8Z8SH2 using the $8 \times 8$ point 2-D DCT and 2-D Huffman table, which is used for JPEG, yields a high compression ratio over a wide compression range. Method D4Z416H1 using the $4 \times 4$ point 2-D DCT, $4 \times 16$ point zigzag scanning, and 1-D Huffman coding has better PSNR than that of method D8Z8SH2 at a compression ratio of less than ten. The performance of method
D4Z44H2 is the worst among the five methods. This means that 2-D Huffman coding is useful for the 8×8 point 2-D DCT but is not suitable for the 4×4 point 2-D DCT. In the comparison between methods D4Z416H1 and D4Z416H2, the 1-D Huffman coding has slightly better performance. This is useful information for hardware implementation because the 1-D Huffman table is much simpler than the 2-D Huffman table.

In the low-compression-ratio range, a relatively short zero run length (ZRL) frequently appears. In the 1-D Huffman table for ZRL, shown in Table 2, the code word for short ZRL is relatively short compared to that of the 2-D Huffman table. This helps to achieve a better PSNR in method D4Z416H1 than those of the other methods in the low-compression-ratio range.

In high-speed cameras, large image degradation due to the image compression is not preferable. Therefore, the performance in the high-SNR region or relatively low-compression region is particularly important. From Fig.4, the 4×4 point 2-D DCT and the appropriate choice of entropy coding methods are useful for the image compression systems in high-speed image sensors because a smaller block size leads to increased compactness of the hardware, and a high coding efficiency is expected in the low-compression-ratio range.

3. Design of Processing Elements for Image Compression

Image compression processing elements (ICPEs) are designed with two methods, D4Z416H1 and D8Z88H2, which provide relatively high coding efficiency, in order to compare the circuit size.

3.1 2-D DCT circuits

2-D DCT circuits are core parts of the ICPE. A row-column decomposition is useful for implementing 2-D DCT circuits of relatively small size. Using the row-column decomposition, the 2-D DCT is defined as follows:
\[ F = C f C^t \]  \hspace{1cm} (2)

where \( f \) is an input pixel block matrix, and \( C \) is a cosine kernel matrix. For 4\times4 points, \( C \) is given by

\[
C = \begin{bmatrix}
  a & a & a & a \\
  b & c & -c & -b \\
  a & -a & -a & a \\
  c & -b & b & -c \\
\end{bmatrix}
\hspace{1cm} (3)
\]

where \( a = \cos(\pi/4) = 1/\sqrt{2}, \ b = \cos(\pi/8) \), and \( c = \cos(3\pi/8) \).

Fig.5 shows the block diagram of the 2-D DCT processing unit using the row-column decomposition. The 2-D DCT processing unit consists of two 1-D DCT processors, a multiplexer, and two transposition memories. The input data are fed into the first DCT processor, and the output data are stored in one of the intermediate memories. The intermediate data are then transposed in the memory. The output of the other intermediate memory is used for the 2nd 1-D DCT processor. Using this architecture, the 2-D DCT can be performed with \( N \) steps in the \( N\times N \) point 2-D DCT. A distributed arithmetic (DA) method is useful for implementing the 1-D DCT processor. Fig.6 shows a block diagram of the 1-D DCT processor using the DA method. In the DA method, the sum-of-product operations required for the 1-D DCT are implemented using ROMs by performing the operations with each bit of the input. The 1-D DCT operation for executing Eq. (2) is expressed as

\[
\begin{bmatrix}
  F(0, k) \\
  F(2, k) \\
  F(1, k) \\
  F(3, k)
\end{bmatrix} =
\begin{bmatrix}
  a & a & 0 & 0 \\
  a & -a & 0 & 0 \\
  0 & 0 & b & c \\
  0 & 0 & c & -b
\end{bmatrix}
\begin{bmatrix}
  f(0, k) + f(3, k) \\
  f(1, k) + f(2, k) \\
  f(0, k) - f(3, k) \\
  f(1, k) - f(2, k)
\end{bmatrix}
\hspace{1cm} (4)
\]

For example, \( F(2, k) \) is given by

\[ F(2, k) = b(f(0, k) - f(3, k)) + c(f(1, k) - f(2, k)) \]  \hspace{1cm} (5)

This operation is performed in each bit of the input. The bit slices of \( f(0, k) - f(3, k) \) and \( f(1, k) - f(2, k) \) have combinations of \( 3 \times 3 \). The address of 9 ways is assigned to the ROM that stores the 10-bit results of the operation of Eq. (5). The ROM outputs are accumulated in a bit-shift manner from the LSB to the MSB using an accumulator and a register.

The hardware sizes of the 4\times4 point and 8\times8 point 2-D DCT circuits designed with the DA method are compared by means of logic synthesis. The number of cells and the number of nets for the 4\times4 point and 8\times8 point 2-D DCT circuits after logic synthesis are estimated as shown in Fig.7. The number of nets and the number of cells in the 4\times4 point 2-D DCT circuits are approximately 1/4 of those of the 8\times8 point 2-D DCT circuits. In the 4\times4 point 2-D DCT, two bits and four bits of input are processed in parallel for the first and second 1-D DCT, respectively, and the throughput is 20 clock cycles. In the 8\times8 point 2-D DCT, the throughput is 80 clock cycles. Since the 8\times8 point 2-D DCT covers a pixel area that is four times larger than that of the 4\times4 point 2-D DCT, the processing performance of the two methods is the same.

3.2 Quantizer, Zigzag Scanner and Huffman Encoder

A block diagram of the ICPE is shown in Fig.8. A quantizer for the DCT coefficients usually requires a digital divider or equivalent hardware for precise control of the bit rate and division by a quantization ma-
trix. However, as discussed later, the on-sensor image compression does not require elaborate control of the bit rate, and the quantizer is implemented by barrel shifters. To accomplish this, the quantizing coefficient, and the components of the quantization matrix, as shown in Eq. (1), are chosen to be of power 2. Using the barrel shifters, the quantization can be performed in a single step.

The zigzag scanner is implemented with memories and a sequencer to generate the address of the memory based on the scanning order. Two sets of RAMs are used for the complementary operations.

The zigzag scanned data are encoded by a Huffman coder. The block diagram of the Huffman coder with 1-D Huffman coding for both DC and AC coefficients is shown in Fig. 9. The input 2-D DCT coefficients are classified into the DC, the non-zero AC (NZAC), and the zero AC (ZAC) coefficients, and these coefficients are assigned to each of coding circuits. Differential coding is used for the DC coefficient, in which the difference of the present DC coefficient from that of the adjacent block is coded. The 12-bit DC differential code is divided into 12 size categories using a size table, and the size is encoded by a Huffman table. The amplitude is calculated in parallel, and a DC codeword composed of the size and the amplitude is obtained. The NZAC coefficients are encoded in the same manner as the DC differential code.

For the ZAC coefficients, the ZRL that is the number of the ZAC coefficients preceding the next NZAC coefficient is calculated at the run-length counter. In the 1-D Huffman coding, the ZRL is coded by the Huffman table shown in Table 2. An important feature of the proposed on-sensor image compression system is multi-bit, or parallel output. For the parallel output, as described later, the code length information of the encoded coefficients is extracted in the Huffman coder using code length tables. The code length, as well as the code word, is transferred to the output buffer memories.

A Huffman coder using 2-D Huffman coding for AC coefficients is similar to Fig. 9, except that a 2-D Huffman table is used for NZAC coefficients and ZRL. The 2-D Huffman table is complicated and the resulting ROM table size becomes large. The total ROM size required for the 1-D Huffman coding of AC coefficients is approximately 1/8 of that of the 2-D Huffman coding.

3.3 Comparison of hardware complexity

The ICPEs using methods D4Z416H1 and method D8Z88H2 are designed, and the number of logic cells and the number of nets calculated by logic synthesis are compared. The results are shown in Fig. 10. The number of cells and the number of nets in method D4Z416H1 are approximately 1/2.5 of those of method D8Z88H2. In method D8Z88H2, the components of the quantization matrix are chosen to be of power of 2, and the quantizer is also designed with a barrel shifter. The hardware complexity of method D8Z88H2 if the quantization matrix for JPEG is used will become larger than that of Fig. 10. Compared to method D4Z416H1, method D8Z88H2 requires twice the memory at input and output. Both the ICPE using method D8Z88H2 and that using method D4Z416H1 have the same processing steps of 80 clock cycles for 8×8 and 4×16 pixel blocks, respectively.


A prototype high-speed image sensor integrating parallel image compression circuits is designed. Method D4Z416H1 is used for the ICPE array because it is the most suitable for implementing the ICPE with respect to the coding efficiency in the low-compression ratio range and the small circuit size.

Fig. 11 shows the chip layout and Table 3 shows the specifications of the designed CMOS image sensor. The pixel array consists of 256 (H) × 256 (V) pixels, each of size 15×15 µm², and a global electronic shuttering function. The pixel outputs are first read through a noise canceller array, and the noise-cancelled signals are then
converted to digital data by a 10-bit cyclic analog-to-digital (A/D) converter array. The architecture of the ADC is the same as that reported in Reference (11).

The data path for one channel of the image compression circuits integrated at the designed CMOS image sensor is shown in Fig.12. The timing chart of the data path is shown in Fig.13. The entire image compression system works as a single instruction multiple data stream (SIMD) type of parallel processing architecture. The pixel array, noise canceller, column parallel ADC and ICPEs and input/output buffer memories are controlled by a single controller. A unit operation cycle to process 4×16 pixels in each ICPE consists of readout cycles of four rows of the image array, which corresponds to a total of 80 basic clock cycles. To complete the image compression process within 80 clock cycles, each data path operates in a pipeline fashion, as shown in Fig.13.

The compressed data are transferred to an external memory system via a 32-bit bus. This multi-bit, or parallel, output is specially designed in high-speed image compression circuits. Since the output data of the Huffman coder are variable length codes (VLCs), the data must be restored in the fixed width (32-bit) buffer memory, as shown in Fig.14(a). This operation is performed by the VLC-to-FLC (fixed length code) converter in Fig. 12. Using the total code length of the previous code words from the beginning, the size of the bit shifting is determined. The new code word is shifted by a barrel shifter, and the shifted code word is merged with the previous data stored in a register by logical OR operation. The buffer memory size to store uncompressed data for 4×16×16×10 = 10,240 (bits), which corresponds to 320 words of 32-bit width. In the prototype design, the minimum compression ratio is assumed to be four, and the buffer memory size is reduced to 80 words × 32 bits. The 80 words of the compressed data are read with 80 clock cycles. As a result, the data output rate can be increased by a factor of four compared to the case without data compression. In other words, the proposed system takes advantage
of data compression for the high-speed data transfer by a fixed factor of four. However, this does not mean that the compression ratio is limited to four. Fig.14(b) shows the output data stream of the designed image sensor. If the compressed data size is smaller than 1/4 of the uncompressed data size, the latter part of data word is occupied by zero, as shown in Fig.14(b). The data words of zero can be skipped when the compressed data are stored in the external storage, and the fully compressed data are finally stored.

The image compression ratio depends on the size of the output buffer memory. The maximum performance can be expected if the memory size for a full frame is prepared, but this requires a large silicon area. Fig.16 shows the compression ratio of the test high-speed picture sequences shown in Fig.15, for two cases of output buffering: with a frame memory and with a memory for a 4x16 pixel block in each ICPE. Although the frame buffering has higher performance, the advantage of the pixel block buffering is not so great.

The size of one ICPE is designed to fit in 16 horizontal pixels, 240 μm with a 15-μm pitch pixel size. Using logic synthesis and automated placement and routing of standard cells, the total length of the ICPE, including the output frame buffer memories is approximately 2.7 mm, as shown in Fig.11. Including the input buffer memory, the total size is approximately 3 mm, which is 80% of the image array.

To achieve 10,000 fps for a 256 × 256 pixel image sensor, with the sampling time of 4.5 μs for global shuttering, all of the image data must be read in 95.5 μs. The required basic clock frequency, $f_c$, is given by

$$f_c = \frac{80 \times (256/4)}{9.55 \times 10^{-8}} \approx 53.6 \text{ MHz}.$$ (6)

The logic synthesis and the placement and routing of standard cells to construct the physical layout has been performed so as to satisfy the operation at 53.6 MHz. Successful operation without timing violation has been confirmed by post layout simulations.

The impact of the on-sensor compression in the present design is that a high-frame rate (10,000 fps)
image with a reasonable image resolution (256×256) can be captured with a 54-MHz clock and a 32-bit bus, which is compatible with a high-speed CameraLink, and the image data of the sensor output can be stored directly in the memories of personal computers via CameraLink. The advantage of on-sensor compression can be exploited in high-resolution (Megapixels) image sensors.

5. Conclusion

The present paper describes parallel image compression circuits suitable for on-sensor integration. A parallel processing architecture with a one-dimensional processing element array is suitable for integrating the image compression circuits at the periphery of the image sensor. Transform coding, especially with a 2-D DCT for a relatively small pixel block is suitable for the parallel image compression circuits considering both the small hardware size and the coding efficiency. A prototype high-speed CMOS image sensor integrating parallel image compression circuits showed the effectiveness of the proposed on-sensor image compression for achieving a high-frame-rate image sensor with a reasonable increase in the silicon area.

[References]

13) "http://www.opencores.org/"