Chapter 2

EHW APPLIED TO IMAGE DATA COMPRESSION

Hidenori Sakanashi, Masaya Iwata, and Tetsuya Higuchi
MIRAI Project, Advanced Semiconductor Research Center (ASRC), National Institute of Advanced Industrial Science and Technology (AIST)
{h.sakanashi, m.iwata, t-higuchi}@aist.go.jp

Abstract: In this chapter, EHW is applied to the lossless image compression, and it is implemented in a chip. The current international standard for bi-level image coding, JBIG2-AMD2, is modified by the proposed method to achieve high compression ratios, where compression parameters are optimized by the enhanced genetic algorithm (GA). The results of computer simulations show a 171% improvement in compression ratios with the proposed method compared to JBIG2 without optimization. The experiment shows that when the method is implemented by hardware with an evolvable hardware chip, the processing speed is dramatically faster than execution with software. This chapter also describes activities concerning ISO standardization to adopt part of the technology used in this method to the JBIG2 standard.

Key words: data compression, JBIG2 (Joint Bi-level Image experts Group, 2), ISO/IEC standard, EHW chip.

1. INTRODUCTION

Since the emergence of Desk Top Publication (DTP) in the graphic (imaging, printing and publishing) industry, digital image data has been handled in many ways, such as with digital printers, on-demand printing/publishing (ODP), and so on. On the other hand, the large costs for storage and transfer of an enormous amount of huge images have become a serious problem. For example, the electrophotographic printer must have the large storage and the broad data-bus to process many high-resolution images quickly. In the case

of the on-demand publishing/printing, the large costs involved in transmission and storage pose serious problems to their practical use because the size of the data becomes extremely large for commercial printed matter with high resolutions.

To overcome these problems, image data must be compressed as much as possible and must be restored to its original state very quickly. Unfortunately, traditional data compression methods cannot satisfy these requirements, because the image data used in the printing/publishing industry have distinctive characteristics.

In this chapter, Evolvable Hardware (EHW) is applied to a data compression system. The proposed method has the following 4 features; (1) adoption of JBIG2 (ISO/IEC Int. Stand. 14492, 2001), the latest international standard, as basis, (2) introduction of a simplified initialization procedure for effective analysis in the genetic algorithm (GA) (Holland, 1975), (3) simplification of the evaluation procedure, and (4) enhanced crossover operations. The results of computer simulations show that the proposed method has a compression ratio that is 171% better than that of JBIG2 without optimization.

We have developed an EHW chip for the proposed method. The result of an experiment using the chip has demonstrated that the compression ratio is higher than for conventional data compression chips, and, moreover, that the speed is dramatically faster than with software execution.

2. LOSSLESS COMPRESSION OF HIGH-RESOLUTION GRAPHIC ART IMAGES

2.1 Image Data for Graphic Arts

In general graphic arts technology, color images are basically transformed into four high-resolution bi-level images before going to press, because press machines can only represent two levels (inked or not-inked). These bi-level images correspond to four colors (cyan (C), magenta (M), yellow (Y) and black (K)). Differences in brightness are represented by varying the density and size of the ink dots, known as halftone dots, which are composed from bi-level pixels located on a fixed rectangular grid.

Thus, the graphic arts images have the following features:

- Sets of bi-level images,
- Very high resolution, and
- Large frequency with which the pixel values switch is high in both the horizontal and vertical directions of a raster scan.

Traditionally, because the image data must be compressed in lossless fashion (reversibly) to avoid distortion or degradation in press quality, MH (Modified Haffman), MR (Modified Read) (CCITT Recommendation T.4,

1998) and MMR (Modified Modified Read) (CCITT Recommendation T.6, 1998) methods, which are well-known international standards for facsimile, have been used for encoding. These methods are based on run-length coding. MH uses a one-dimensional model, while a two-dimensional model is adopted in MR and MMR (the principle in MMR is the same as that in MR, but some error correction mechanisms are eliminated to achieve higher compression efficiency). These methods provide fairly good compression for line-art or text images. However, these methods are unsuitable for data where the switching frequency for pixel values is high, like halftone images, because they code the positions where pixel values are switched for each line.

In contrast, JBIG, a template-based arithmetic coding method (Sayood, 2000), was a general-purpose compression method for bi-level images, and JBIG2 (ISO/IEC Int. Stand. 14492, 2001) was standardized as its successor in 2000. JBIG2 was designed to upgrade the lossless JBIG encoding method and to add a lossy compression mode based on pattern matching. As lossy encoding is not relevant to the compression of graphic art images, this chapter focuses only on the lossless encoding mode in JBIG2.

2.2 Lossless Image Compression with JBIG2

The template-based arithmetic coding method is based on the hypothesis that the probability of a given "pixel to be coded" having a specific value depends on the values of a limited number of preceding pixels. In JBIG2, the MQ-Coder is adopted as an arithmetic coder; we shall limit our discussion here to this template-based coding method because a detailed explanation of the MQ-Coder would be beyond the scope of this chapter (the principle of arithmetic encoding and the procedure for the MQ-Coder are detailed in (Sayood, 2000) and (ISO/IEC Int. Stand. 14492, 2001), respectively).

In JBIG2 and its enhanced version called JBIG2-AMD2 (ISO/IEC Int. Stand. 14492/Amd.2, 2003), 16 pixels preceding the pixel to be coded are observed in calculating the probability that it takes a specific value {0, 1}. This probability is used to predict the values of the pixels to be coded, and the accuracy of prediction strongly influences the compression efficiency. Here, these observed pixels are called "reference pixels," and the configuration of their positions is called a "template." Figure 2-1 is a diagram of the template consisting of 16 reference pixels used by JBIG2-AMD2. The question mark in the figure represents the pixel to be coded and is not part of the template. While the positions for the 4 reference pixels are fixed, as marked with a #, there are also 12 special reference pixels called the adaptive template (AT) pixels, indicated in the figure as A_i $\{i = 1, \ldots, 12\}$. AT pixels can arbitrarily change positions within the range marked by the dotted line to achieve higher prediction accuracy and, in turn, higher compression efficiency.

However, it is a very difficult problem to optimize AT pixels according to the characteristics of images to be compressed. Therefore, the next section proposes an extended GA to optimize the configuration of the template of JBIG2-AMD2 quickly and efficiently.

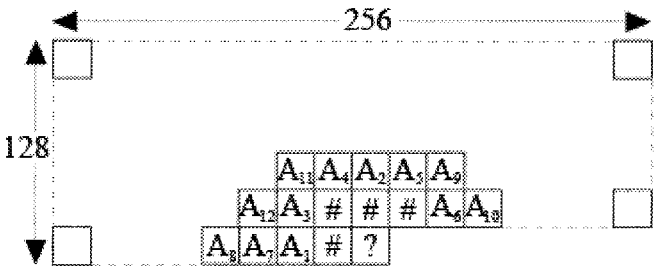

Figure 2-1. Default configuration of JBIG2-AMD2 template

3. EXTENDED GA FOR TEMPLATE OPTIMIZATION

Since the basics and procedures of GA are already described in the previous chapter, this chapter shows some of the modifications and enhancements for the problem of optimizing the JBIG2 templates.

3.1 Coding of Chromosomes and Initialization of a Population

As each AT pixel exists within an area of 256×128, as shown in Figure 2-1, its location can be specified by 15 bits. If the number of AT pixels is M, then the total length of a chromosome is $15M$ bits. The enhanced template proposed in this chapter has 12 AT pixels, so the length of the chromosome is 180 ($=15 \times 12$) bits, as shown in Figure 2-2. In the computational simulation described later, the population consists of 30 chromosomes.

An initial population is usually generated at random, although some GAs adopt a kind of biased initialization, based on certain information related to the problem. For example, in our previous method, the initial population was generated from a seed template by a mutation operation that was derived from a multiple regression analysis (Sakanashi, 2001). However, the computational complexity of the multiple regression analysis was enormous. Accordingly, in this chapter, the seed template is determined by assigning reference pixels one by one based on their degree of correlation to the pixel to be coded.

In order to calculate the degree of correlation between the pixel to be coded and each candidate AT pixel, it is necessary to scan the entire image to check whether the pixels are of the same value. Defining image size as XY, because the number of AT pixel candidates is approximately 256×128, the number of observations and comparisons required will be $XY \times 256 \times 128$. When the image size is small, the number of calculations would not pose a major problem. However, because graphic art images have very high resolutions, this number becomes extremely large. For example, as an A4 image with a resolution of 2400 dpi consists of roughly 20000×28000 pixels, approximately 1.8×10^{13} observations and comparisons would be needed to calculate the correlations.

Thus, to reduce the number of comparisons between pixel values, the degree of correlation is only checked for pixels to be coded, which are stochastically selected at a probability P_{scan}, and the respective candidate AT pixels. Investigating this P_{scan} probability in a second preliminary experiment, we found that a template of sufficient quality to serve as the seed template in initializing the population can be obtained with $P_{scan} = 5000/XY$, which is the P_{scan} value used in this chapter.

In the proposed method, one chromosome in the initial population is the bit string representing this seed template, with the remaining chromosomes being created by mutating this chromosome at twice the mutation rate to be explained later.

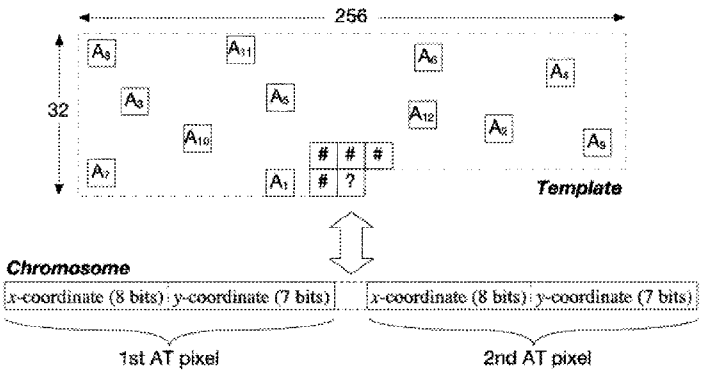

Figure 2-2. Coding of chromosomes

3.2 Random Partial Evaluation

As the objective function to calculate the fitness value of a chromosome, the proposed method uses the inverse of the compressed data size achieved by the template represented by the chromosome. Thus, a chromosome repre-

senting a good template that compresses the image data well will have a higher fitness value.

Incidentally, although it would be possible from an information theory perspective to use entropy as the fitness value rather than the compressed data size, there are two reasons, examined in preliminary experiments, for not doing so: (1) there are no significant differences between the costs for executing the MQ-Coder and for calculating entropy. (2) As the MQ-Coder can dynamically learn the statistics of the given data sequence, the template with the lowest entropy calculated in a static way does not always yield the maximum compression ratio.

Employing the compressed data size for evaluation, however, means that the image data must be repeatedly compressed to obtain the fitness value for each chromosome. The number of times the image data must be compressed will be $N \times G$ times, where N is the population size and G is the number of generations until the termination. Although the easiest way to reduce the evaluation costs is to only use a small part of the image rather than the entire image, evaluation accuracy and the reliability of the fitness value would deteriorate, making it impossible to discover the best template yielding the greatest compression efficiency.

Accordingly, this chapter proposes a procedure to solve this problem, consisting of the following two steps:

- Evaluation of the chromosomes is carried out using a small area of the entire image. The location of this area is changed at random, if a new best chromosome fails to emerge in the population within a given generation interval, $G_{interval}$. (If the location of the area is changed at every generation, the GA will fail because it cannot cope with such drastic fluctuations in the evaluation criteria.)
- When a new best chromosome does emerge, a hillclimb search is executed with this chromosome as the starting point. To avoid over-fit to the small area of the image, in the hillclimb search the pixels for evaluation are chosen from the entire image at a probability $P_{SamplePixel}$.

In this chapter, this procedure is referred to as "random partial evaluation," and the computational simulation described later uses the following parameters: $G_{interval} = 20$, $P_{SamplePixel} = 0.005$, and an area size of 1024×1024 pixels.

3.3 Genetic Operations with Template Crossover

This section proposes a new crossover operator suitable for template optimization. For other genetic operators, this chapter adopts existing methods such as tournament selection (Goldberg, 1991) and bit-wise point mutation. In the computational simulation in the next section, 80% of the chromo-

somes are chosen by tournament selection with a tournament size of 2, and the mutation ratio is 1/180.

In the template representation, there is no notion of order for the reference pixels. That is, if the AT pixels A1 and A2 in Figure 2-2 were exchanged, we would obtain the same compressed data. However, the pixels must be arranged in a certain order within the chromosome representation and this causes a serious problem.

If a simple 1-point crossover method is used, chromosomes with common reference pixels are frequently generated. For example, Figure 2-3 illustrates a case where the chromosomes C1 and C2 are generated by the 1-point crossovers of P1 = A|B|C|D and P2 = C|D|E|F. These chromosomes represent the templates PT1 and PT2, consisting of the set of reference pixels {A, B, C, D} and {C, D, E, F}, respectively. With the crossover point between the second and third pixels, one child, C2 = C|D|C|D, would unfortunately contain only two unique reference pixels, as shown as CT2 in the figure. Because the compression ratio generally tends to be higher when the number of pixels is larger, a chromosome representing a template with overlapping reference pixels will have a poorer evaluation.

Thus, in this chapter, we propose a special crossover procedure called template crossover, as follows:

1. A pair of chromosomes, P1 and P2, is compared to identify any common reference pixels, P_{common}.
2. If present, common reference pixels are removed from P1 and P2, respectively, to produce P1′ and P2′.
3. If the lengths of P1′ and P2′ are 0, P2 is mutated and the process is terminated.
4. Otherwise, reference pixels in P1′ and P2′ are exchanged in a fashion similar to bit-wise uniform crossover, with the results being defined as P1″ and P2″.
5. Finally, P_{common} is concatenated at the ends of both P1″ and P2″, with the results overwriting the original P1 and P2, respectively.

The check in step 3 for the lengths of P1′ and P2′ ensures that identical templates never appear in the population. Moreover, this elimination of redundancy efficiently reduces the search space and so contributes to improve GA search efficiency. Because a template with m reference pixels can be represented in (${}_mP_m = m!$) different ways as chromosomes, the number of redundant evaluations is greatly reduced by removing identical chromosomes.

The parameters of the proposed method mentioned above are summarized in Table 2-1.

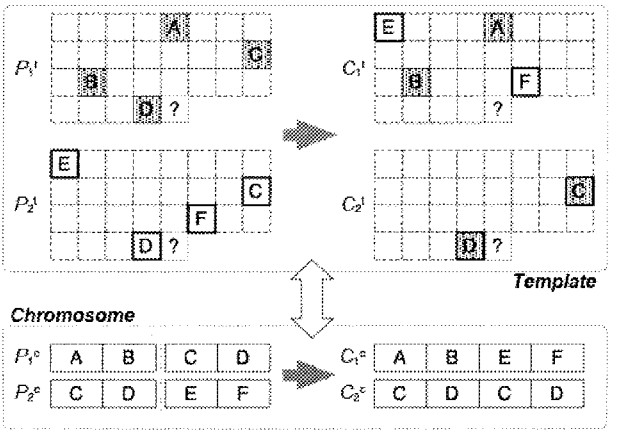

Figure 2-3. Example of a template with overlapping AT pixels being generated by a one-point crossover operation

Table 2-1. Parameter settings

Population size	30
Length of chromosome	180
Max generation	10000
Selection method	Tournament selection
Crossover method	Template crossover
Crossover ratio	0.8
Mutation method	Bit-wise point mutation
Mutation rate	1/180
P_{scan}	5000/[Image size]
Sample area size for evaluation	1024 × 1024
$G_{interval}$	20
$P_{SamplePixel}$	0.005

4. COMPUTATIONAL SIMULATIONS

This section presents the results of the computational simulations executed to examine the performance of the proposed method. This experiment used a set of test images containing the cyan and magenta images of N5, N6 and N8 in SCID (ISO/IEC Int. Stand. 12640, 1997), which were processed by raster image processor (RIP) to decompose the color image into the four bi-level images and to increase the resolution to 2400 dpi. They were chosen as a test image because they have the medium, smallest and greatest entropy levels of the eight images in SCID. Similarly, the cyan and magenta image has a larger level of entropy than the yellow and black images.

Firstly, to verify the effect of extending the GA for template optimization, an experiment was carried out with the 4 conditions shown in Table 2-2, using only the cyan N8 image.

Figure 2-4 shows a graph plotting the mean best fitness values achieved in 3 runs for each condition. As the evaluation areas changed at random periods, the fitness values fluctuated sharply, making it difficult to differentiate the performances across the 4 conditions.

The graph in Figure 2-5 plots the compression ratios rather than the fitness values, and Figure 2-6 is a close-up of Figure 2-5. These show that compression ratios improved as the GA search progressed, which is almost saturated within 1000 evaluations in every condition, and one of our future tasks would be to develop the appropriate termination criteria.

Looking at the contrast between the pairs of conditions [i]+[iii] and [ii]+[iv], which differ in terms of whether initialization was based on correlation analysis, the fact that the results for [iii] and [iv] are respectively better than [i] and [ii] indicates that the proposed population-initialization method effectively boosts the search performance of the GA.

Moreover, we can observe that the results for [ii] and [iv] are better than [i] and [iii] even though the initial populations were the same for the pairs of conditions ([i]+[ii] and [iii]+[iv]). This fact demonstrates the efficiency of the template crossover operator, which was adopted in the conditions [ii] and [iv] but not in conditions [i] and [iii].

Table 2-3 provides the results of the simulations executed to compare the performances of (1) JBIG2-AMD2 with the default template, (2) our method only with the initialization, and (3) the complete proposed method with the enhanced GA (condition [iv]). This table shows that the proposed method can achieve much better compression ratios than the default state of JBIG2-AMD2.

Additionally, the compression ratio achieved by the initialization-only method is about 16.6% ~ 99.7% better than JBIG2-AMD2 with the default template, although there is little difference between them in terms of computational costs. We can say that, from the viewpoint of practical use, it is a very reasonable performance in terms of compression efficiency and processing speed.

Table 2-2. Simulation conditions as a function of the proposed genetic operators

		Template crossover	
		OFF	ON
Initialization with correlation analysis	OFF	[i]	[ii]
	ON	[iii]	[iv]

Table 2-3. Comparison of compression ratios

		JBIG2-AMD2 (Default template)	Proposed method (Initialization-only)	(Condition [iv])
N5	C	6.58	9.47 (+44.0%)	10.82 (+64.5%)
	M	6.42	8.88 (+38.4%)	9.90 (+54.3%)
N6	C	6.10	12.18 (+99.7%)	16.58 (+171.9%)
	M	5.56	10.30 (+85.3%)	13.34 (+140%)
N8	C	4.99	5.90 (+18.3%)	6.51 (+30.5%)
	M	5.15	6.00 (+16.6%)	6.49 (+26.1%)
Process time		33 sec.	45 sec.	1.2 hours

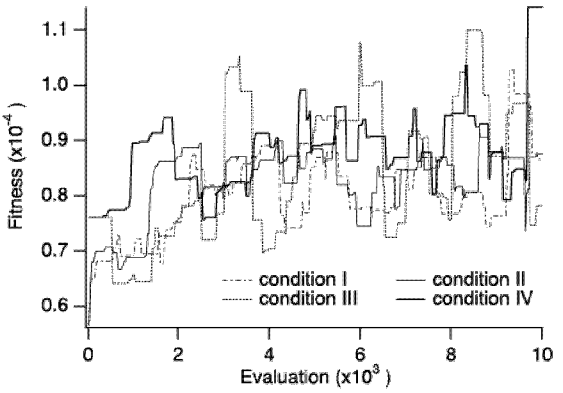

Figure 2-4. Learning curve

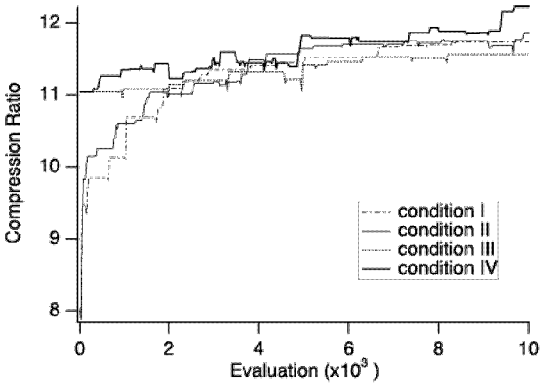

Figure 2-5. Improvements in compression ratios

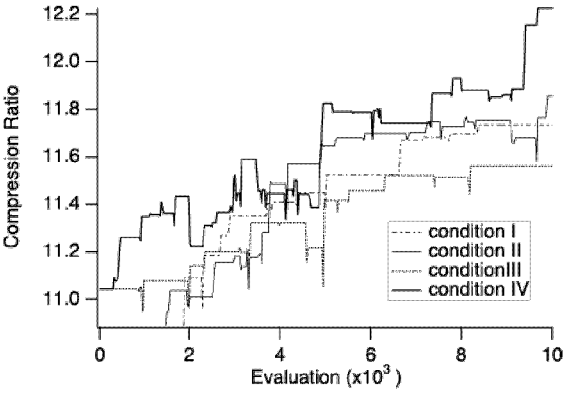

Figure 2-6. Improvements in compression ratios (close-up)

5. IMPLEMENTATION OF THE EVOLVABLE HARDWARE

As described in the previous section, our proposed method provides very high compression efficiency. However, from the practical perspective of use in the graphic industry, it is also necessary to implement high-speed compression/decompression because the data for the graphic art images handled by this method is huge. In this section, we explain the implementation of evolvable hardware (EHW) to speed up this method.

5.1 Architecture

The data compression EHW chip consists of a template optimizer, a data compressor, and an evaluator, as shown in Figure 2-7. In this implementation, both template optimization and evaluation are executed on the host PC. We adopted this kind of implementation because the systems that would employ our proposed compression method, e.g. electrographic printers, are equipped with the latest CPUs that can execute these procedures sufficiently and quickly for practical use. Moreover, processing with high flexibility becomes possible by executing optimization and evaluation on a PC. In addition, when our compression technique is in practical use on electrographic printers, only fast compression and decompression are required. In most cases, optimization and evaluation are executed in advance before the equipment is manufactured, and then the results are embedded in the equipment. For these reasons, this chip is optimized for practical use.

In the trial chip, which we discuss in this section, the area of the template is limited to 32 × 8 pixels with 10 reference pixels (Figure 2-8). In this case, 8 bits are required to specify the location of each reference pixel, so a template can be represented by 80 bits.

A block diagram of the chip is shown in Figure 2-9. The chip mainly consists of an MQ-Coder (ISO/IEC Int. Stand. 14492, 2001), which is the encoder/decoder, an image data memory that stores the input data, a reference buffer, and a context generator (shown as the hatched areas in Figure 2-9).

In order to execute the encoding/decoding procedures in parallel, and thus speed up processing, a feature of this architecture is the incorporation of two MQ-Coders.

The specifications of the sample chip are shown in Table 2-4, and a layout image of the chip is shown in Figure 2-10.

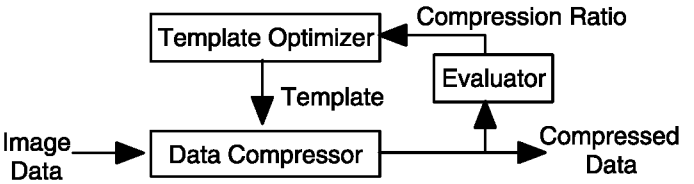

Figure 2-7. Configuration of the data compression system for our method

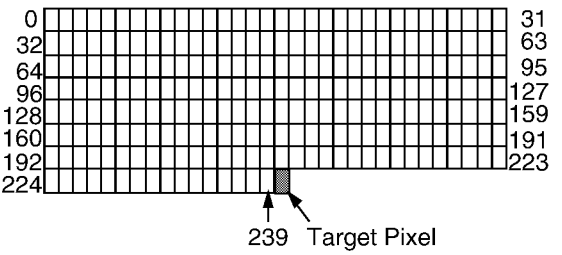

Figure 2-8. The area of the template in the chip

5.2 Elements of the Chip

5.2.1 Image Data Memory

The image data memory stores each line of the image data so that the reference buffers can extract reference areas efficiently from the image data. The size of this memory is 320 words × 32 bits × 9 lines. Each line in the

memory holds one line of image data. If a line of image data is longer than a line of memory, then the image data line is divided into memory-line length units at preprocessing. The PC accesses the memory in 32-bits groups. The lines are updated whenever a process is completed. Accordingly, this memory always holds the areas for extraction by the reference buffers. The memory is divided into two groups at the middle of each line, which are used by reference buffer 1 and 2, respectively.

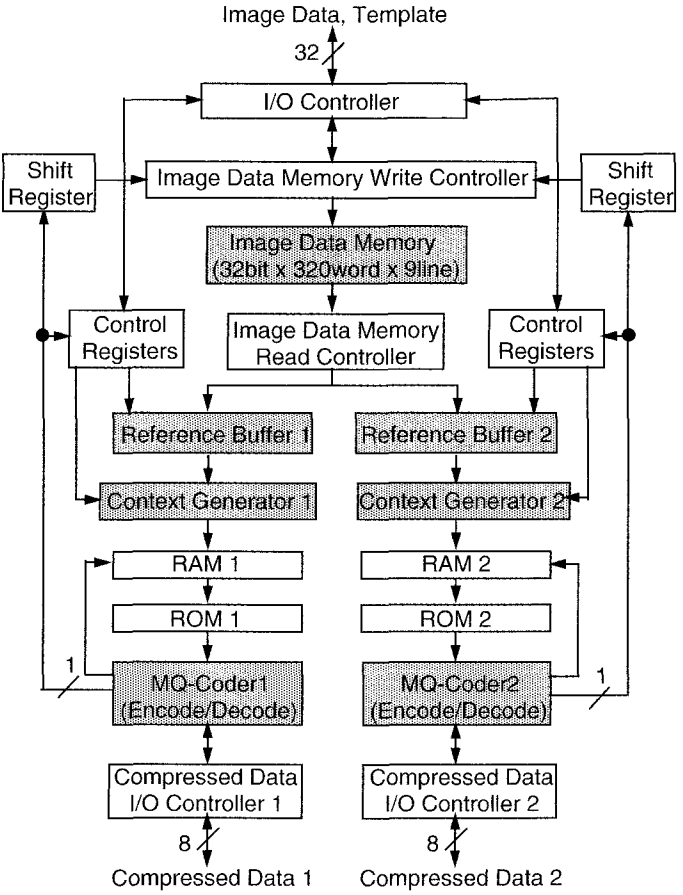

Figure 2-9. Block diagram of the chip

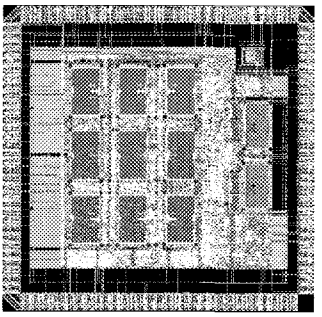

Figure 2-10. Layout image of the chip

Table 2-4. Sample chip specifications

Technology	0.18μm CMOS
Package	160 pin QFP plastic
Die Size	5.14 mm × 5.19 mm
Gate Count	about 56,000
Clock Frequency	133 MHz (maximum)
Supply Voltage	1.8 V (internal), 3.3 V (I/O)
Acceptable Image Size	10,240 × 65,536 (maximum)

5.2.2 Reference Buffer

The reference buffers are buffers for the reference areas of the templates. This chip has two buffers corresponding to the two MQ-Coders. Each reference buffer has two buffers of 64 bits × 8 lines, as shown in Figure 2-11. The data stored in the buffer represents the reference area of a template and is extracted from the image data memory (Figure 2-12). In the extraction procedure, the data is extracted by shifting the reference area in one-word (32 bits) increments, with each data set being stored into buffers A and B in turn. At the edge of an image, the data is clipped so that one-word of data protrudes from the edge, as shown in Figure 2-12. Data outside the image border is set with a pixel value of 0. The data assigning each column, as shown at the top of Figure 2-11, is used to match the data into the columns in Figure 2-12. The data is clipped in this way in order to efficiently process the data across each word. For example, a reference area crossing the border of the image would be processed with the data in the buffer A in Figure 2-11, and an area covering word0 and word1 would be processed with the data in the buffer B. The buffers are updated after the processes using the buffers are complete.

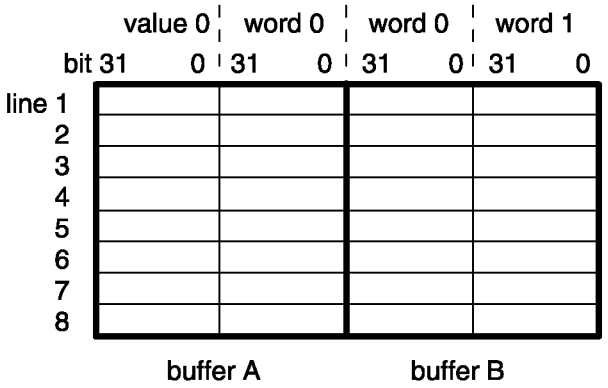

Figure 2-11. A reference buffer

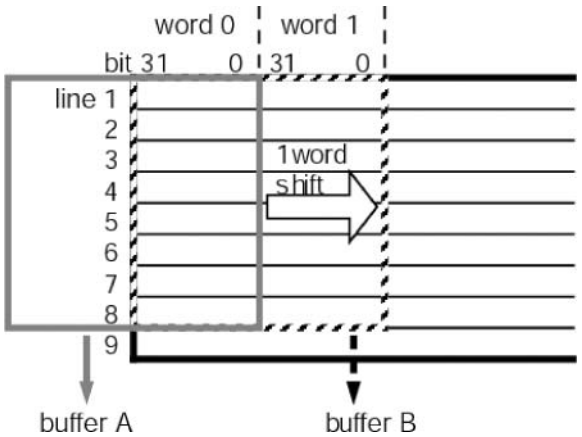

Figure 2-12. Extraction of reference data from the image data memory

5.2.3 Context Generator

The context generator generates a context (10 bits) from a template stored in the register and the data for a reference area in the reference buffer. A context, which is used by MQ-Coder, is the values of the 10 reference pixels for a template. The circuit mainly consists of 10 multiplexers of 240-inputs and 1-output. The multiplexer extracts the pixels specified by the template from the 240 pixels in the reference area of the template (Figure 2-8).

In this method, because the circuit in the context optimizer is optimized for each image, a template is selected in the learning mode as being optimal

for that image. Thus, in this system the context optimizer has the same role as the reconfigurable device in EHW.

5.2.4 Encoder/Decoder (MQ-Coder)

The encoder/decoder used here is the same as that in the international standard for bi-level image encoding, JBIG2. It executes arithmetic encoding using pairs of context (10 bits) and a pixel value (1 bit). The circuit is based on the specification of JBIG2 (ISO/IEC Int. Stand. 14492, 2001) and JBIG2-AMD2 (ISO/IEC Int. Stand. 14492/Amd.2, 2003). The chip has two MQ-Coders executed in parallel. The RAM and the ROM placed in front of each MQ-Coder are also defined in the JBIG2 specifications.

5.2.5 Other Components

There is an I/O controller for the data I/O control between the chip and the PC. A shift register is used in decompression to send the decompressed data immediately into the image data memory and the reference buffers.

5.3 Execution Procedure

5.3.1 Compression

The procedure for compressing one image is as follows:

(1) Set a template in the control register.
(2) Set image data in the image data memory.
(3) Compress the data for one line. (The subprocedure is shown in (a) – (e).)
    (a) Clip data of 64 bits × 8 lines from the beginning of the image data memory and store in the reference buffer. Assign a pixel value of 0 for out of image areas (Figure 2-12).
    (b) Extract 10 pixels as directed by the template from the reference data in the reference buffer using the context optimizer.
    (c) The context (10 bits) and the image data (1 bit) are sent to the MQ-Coder (encoder part) via the RAM and the ROM. The encoded data are sent to the I/O controller.
    (d) Iterate (b) – (c) for the data in one buffer of the reference buffer. In this procedure, the reference area is shifted by 1 bit from the beginning to the end of the buffer, as shown in Figure 2-13.
    (e) Iterate (a) – (d) for the data of one line.
(4) Iterate (2) – (3) and send the compressed data into the I/O controller. The data is picked up from the external PC.

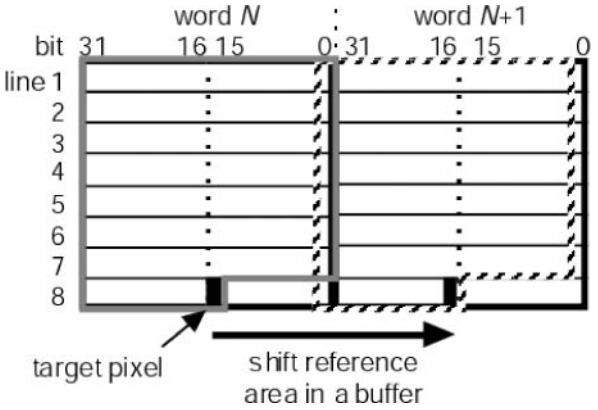

Figure 2-13. Shift of reference area in a reference buffer

5.3.2 Decompression

The procedure for decompressing one image is as follows:

(1) Set a template in the control register. Set all data in the image data memory to 0.
(2) Set the compressed data in the compressed data I/O controller.
(3) Decompress the data for one line. (The subprocedure is shown in (a) – (e).)
   (a) Clip data of 64 bits × 8 lines from the beginning of the image data memory and store in the reference buffer. A pixel value of 0 represents either an out of image area or a pixel that is not decompressed yet.
   (b) Extract 10 pixels directed by the template from the reference data in the reference buffer using the context optimizer.
   (c) The context (10 bits) and the compressed data (8 bit) are sent to the MQ-Coder (decoder part) via the RAM and the ROM. The decoded data is sent to the reference buffer and the image data memory via the shift register.
   (d) Iterate (b) – (c) for the data in a buffer in the reference buffer. In this procedure, the reference area is shifted by 1 bit from the beginning to the end of the buffer, as shown in Figure 2-13.
   (e) Iterate (a) – (d) for the data of one line.
(4) Iterate (2) – (3) and send the decompressed data into the image data memory. The data are picked up from the external PC.

5.4 Performance Evaluation

5.4.1 Evaluation System

We have conducted an experiment to evaluate compression and decompression performance. The architecture of the evaluation system is shown in Figure 2-14. The system consists of a PC, an interface board, and the chip.

The PC has a hard disk and software. The hard disk stores the input and output data. The original data before compression is stored in TIFF format. The software mainly consists of I/O preprocessing and postprocessing, the interface between the board and the PC, the template optimizer, and the evaluator. The I/O preprocessing includes data reading and format conversion. The postprocessing executes header-making and combining compressed data in compression, as well as combining expanded data and conversion into TIFF format in decompression, and writing data into a file. The interface board executes the interface processing between the chip and the PC.

5.4.2 Speed

We evaluated the performance of the chip in terms of the speed of compression and decompression. The clock frequency was 133 MHz. First, we evaluated the processing speed of the chip when using only one MQ-Coder. The compression speed was 3 – 5 clocks/bit, 26.6 – 44.3 Mbit/s. The decompression speed was 5 – 7 clocks/bit, 19.0 – 26.6 Mbit/s. The speeds of this chip are the same as those of an MQ-Coder, indicating that the MQ-Coder represents a bottleneck to the processing speed of the chip. The changes in speed were due to the type of data processed.

Taking these results as a baseline, we next evaluated processing speeds when the two MQ-Coders are used in parallel. If the two MQ-Coders have the same processing speeds, then the total processing speed should be doubled when these are executed in parallel. The maximum speed for compression was 88.6 and the speed for decompression was 53.2 Mbit/s. Although the processing is executed in parallel, because the MQ-Coders process exclusively different data, the actual total speed is determined by the slower MQ-Coder. Thus, the total speed of the chip with parallel processing is twice the speed of the slower MQ-Coder.

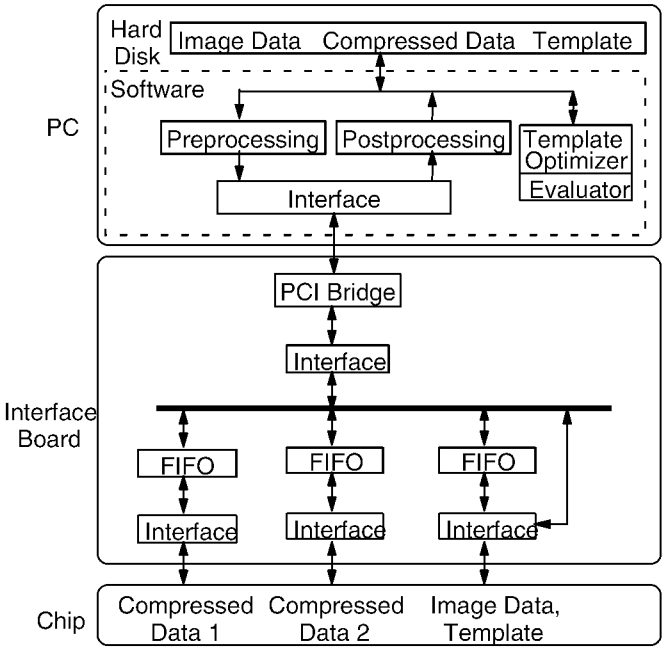

Figure 2-14. Block diagram of the evaluation system for the chip

Next, we evaluated the processing speed of the evaluation system including the PC (Figure 2-14). The speed was 5.1 Mbit/s for compression and decompression. The reason for the slower speeds is that there are too many processing threads executed in the PC. This results in slower PC processing, which is a bottleneck for the total system. Methods of improving this speed are discussed in the next section.

We compared the processing speed of the chip with the software. The compression and decompression speeds of the software using a PC (Pentium 4, 2 GHz) were 0.73 Mbit/s. This shows that the speed executed on the evaluation system is about 7 times faster than when executed on an external PC, and that the speed of the chip is two orders of magnitude faster than the speed of the software.

For reference, we present the processing speed of the JBIG chip (ISO/IEC Int. Stand. 11544, 1993), which is a standard compression method. Although JBIG is a prior version of JBIG2, we have selected this because no chip for JBIG2 currently exists. The JBIG chip is commercialized with the specifications of 1 clock/bit and 115 MHz clock speed. The speeds of the evaluation system and our chip are slower than the JBIG chip. However, it should be noted that the compression ratio obtained by our chip is about 50% better than that of JBIG (Sakanashi, 2001). The JBIG chip is faster because it

compresses and decompresses data in 1 clock/bit. The possibility of 1 clock/bit processing for our chip is discussed in the next section.

As another reference, we briefly show the estimated processing speed of a JBIG2 chip, which has only been designed and evaluated for performance in simulations (Denecker, et al., 2002). The compression speed was 340 Mbit/s, and the decompression speed was 170 Mbit/s. Though the speed is faster than our chip, as we discuss in the next section, if our chip can compress data in 1 clock/bit with 200 MHz clock, then it would also execute compression and decompression at similar speeds to the simulated JBIG2 chip. However, the JBIG2 chip involves a larger template than our chip because its context has 12 pixels, and the reference area is 63×33 pixels. Because our chip is a prototype, we limited the size of the template due to restrictions over die size. We are currently designing new compression circuits using a technique to reduce the circuit size while still implementing a sufficient context size and reference area for practical use.

5.5 Discussion

In this section, we discuss how to improve the processing speed of the evaluation system and the chip.

To exploit the full potential of the chip, we need PC software that controls the chip without making the chip wait. However, the software is at present a bottleneck to the speed of the chip. The main reason for this is that there are too many processing threads (see Section 5.5.1) in the PC with the two MQ-Coders working in parallel, leading to slower PC processing speeds. Consequently, the speed of data transfer to and from the PC cannot match the processing speed of the chip, and much time is required for preprocessing and postprocessing, such as format conversion or file access, creating a bottleneck in the system.

Possible methods for accelerating these processes include using DMA for data transfer and using micro controllers for preprocessing and postprocessing, either on the chip or on the board. The processes will also be accelerated when the speeds of CPUs, PCI buses, and disc access on a PC become faster in the future. Moreover, a speed up in data transfer is expected by fabricating the circuits for the interface board in the chip. A PC with multiple CPUs executing many threads faster would also be effective.

Next, we discuss how to increase the processing speed of the chip. The chip has a write-back procedure into the RAM from the MQ-Coder (Figure 2-9). The MQ-Coder experiences some waiting time because of this procedure. By embedding timing adjustment circuits into the chip, we estimate that a compression speed of 1 clock/bit, 133 MHz per 1 MQ-Coder will be possible. However, the decompression process will still take more than

1 clock/bit, because the process has the write-back procedure into the reference buffer to generate a context. The problem of the additional time required for the decompression process will be a focus for our future research.

6. CONCLUSION

In order to achieve good compression efficiency for very high-resolution images, this chapter has proposed a new lossless compression method with a mechanism to optimize its own parameters using a genetic algorithm (GA). The proposed method has been developed on the basis of JBIG2, the current international standard for bi-level image encoding, and has an improved coding mechanism with an enhanced template of 12 reference pixels (AT pixels). An extended GA is also used in this method, which has the following features: (a) fast and effective mechanisms for initializing the population based on correlation analysis and random sampling of pixels, (b) a random partial evaluation procedure, and (c) template crossover.

The following results of computational simulations prove its advantageous performance: (1) a compression efficiency that was about 23% better than JBIG2, albeit after a long learning period (Section 2.3), (2) a compression efficiency approximately twice that of JBIG2 with default parameter settings with only a slight increase in computational cost, and (3) the new proposed method was able to complete learning about 96 times faster than our previous method using GA.

We have outlined the architecture of an EHW chip and reported the results of experiments to evaluate performance. The comparison of processing speeds with software execution showed that the evaluation system is about 7 times faster, and that the chip is two orders of magnitude faster.

Because of its advantageous performance in image compression, the ISO/IEC JTC 1/SC 29/WG 1 committee determined to adopt part of the technology used in this proposed method as an amendment to the JBIG2 standard. As the next step, we will propose to the ISO TC 130/WG 2 committee that TIFF/IT, the international standard of the file format for the prepress data exchange, adopts our proposed method as a compression method.

Acknowledgment

This work was supported by NEDO and AIST. We thank Dr. Nobuyuki Otsu, AIST fellow, and Dr. Masataka Hirose, the project leader of the advanced semiconductor research center, AIST.

References

CCITT Recommendation T.4. 1998. Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus.

CCITT Recommendation T.6. 1988. Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus.

Denecker, K., et al. 2002. Design of an Improved Lossless Halftone Image Compression Code, Signal Processing. In *Image Communication, vol. 17*, 277-292, Elsevier.

Goldberg, D. E. and K. Deb. 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In *Foundations of Genetic Algorithms*, 69-93, Morgan Kaufmann.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

ISO/IEC International Standard 11544 | ITU-T Recommendation T.82. 1993. Coded Representation of Picture and Audio Information – Progressive Bi-level Image Compression.

ISO/IEC International Standard 12640. 1997. Graphic technology – Prepress digital data exchange – CMYK standard colour image data (CMYK/SCID).

ISO/IEC International Standard 14492 | ITU-T Recommendation T.88. 2001. Information Technology – Lossy/lossless coding of bi-level images.

ISO/IEC International Standard 14492/Amd.2 | ITU-T Recommendation T.88/Amd.2. 2003. Information Technology – Lossy/lossless coding of bi-level images, AMENDMENT 2: Extension of adaptive templates for halftone coding.

Sakanashi, H., M. Iwata and T. Higuchi. 2001. A Lossless Compression Method for Halftone Images using Evolvable Hardware. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 314-326, Springer.

Sayood, K. 2000. *Introduction to Data Compression (2nd edition)*. Morgan Kaufmann.