Evolvable Hardware Chip for High Precision Printer Image Compression

Hidenori Sakanashi [1], Mehrdad Salami [1], Masaya Iwata [1], Shogo Nakaya [2]
Tsukasa Yamauchi [2], Takeshi Inuo [2], Nobuki Kajihara [2] and Tetsuya Higuchi [1]

[1] Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba 305-0047, Japan
{h_sakana, m_salami, miwata, higuchi}@etl.go.jp

[2] RWCP Adaptive Device NEC Laboratory

Abstract

This paper describes a data compression chip for the high-precision electrophotographic printer using Evolvable Hardware (EHW). EHW is a new hardware paradigm which combines Genetic Algorithm (GA) and reconfigurable hardware technology such as FPGA (Field Programmable Gate Array). In EHW, GA is used to search for the most desirable hardware structure to a given task. If the task requirement changes, GA is invoked to get a better hardware structure and EHW is reconfigured that way. In data compression, EHW is used to implement the most adequate compression method directly in hardware according to the characteristics of the target image. The EHW-based compression chip attains approximately twice the compression compared with the international standard called JBIG. This chip is the first EHW-chip to lead to a commercial product.

Introduction

The electrophotographic (EP) printing is the next generation technology in printing and publishing industry to print books with a high-precision photo quality.

One A4-size EP image of 1200 dpi requires 70 MBytes for storage, and the EP printer processes hundreds different pages with the speed of 100 page/min. It means that, for printing a book with 100 pages, 7 giga bytes of image data must be transferred to the printer in the speed of 1800 Mbyte/min. On the other hand, the data transfer speed of the usual hard-disk drive is only 300 Mbyte/min. Thus, the EP printers must employ data compression techniques. They are required (1) to compress image data efficiently, and (2) to reconstruct the compressed data very fast. However, the traditional techniques are insufficient both in the compression ratios and the decompression speed.

This paper proposes a data compression chip using Evolvable Hardware (EHW) that enables on-line reconfiguration of the hardware structure. In EHW, genetic algorithm determines how the hardware structure should be reconfigured whenever a new hardware structure is needed for a better performance. EHW can adaptively reconfigure its hardware structure to compress the target images more efficiently. Besides, it can change compression method during compressing one image. As a consequence, we obtained three times higher compression rate than that of the current printing machine, and five times faster compression speed than the exhaustive search method with similar performances. Moreover, EHW can meet the speed requirement of the printer because the expansion of data is done by EHW hardware. This chip is determined to be adopted in a commercial EP printer.

In the rest of this paper, section 2 explains the foundation of EHW. The data compression in EP printing is mentioned in section 3. In section 4, the data compression method adopting EHW is described in detail, and the results of computational simulations show that EHW exhibits the superior performance to the other methods. Section 5 shows the design of a chip implementing based on this method [†], and section 6 concludes this paper.

Evolvable Hardware

This section explains genetic algorithm (GA) and Evolvable Hardware (EHW) in which GA is combined with the reconfigurable hardware.

Genetic Algorithm

Genetic Algorithm (GA) was proposed to model adaptation of natural and artificial systems through evolution (Holland 1975), and is well known as one of the most powerful search procedures (Goldberg 1989). The canonical GA has a population of chromosomes, each of them is obtained by encoding a point in the search space. Usually, they are represented by the strings of binary characters.

At an initial state, chromosomes in the population are generated at random, and processed by many operations, such as *evaluation*, *selection*, *crossover* and *mutation*. The latter three operations are called the genetic operations, and one cycle of the evaluation and the genetic operation is counted as a generation. The evaluation assigns the fitness values to the chromosomes, which indicates how well the chromosomes perform as solutions of the given problem. According to the fitness values, the selection determines which chromosomes can survive into the next generation. The crossover chooses some pairs of chromosomes, and exchanges their sub-strings at random. Finally, the mutation randomly picks some positions in

[†] Mitsubishi Heavy Industry Ltd. will use a similar architecture as a basis for the data compression chip in the next model of their electrophotographic printer.

the chromosome and flips their values.

The major advantages of GA are its robustness and superior search performance in many type of problems without a prior knowledge (Davis 1991). However, it is rarely reported that GA is used for industrial applications or commercial products, because of the cost for fitness evaluation. If the evaluation can be executed very quickly by the specific hardware device, however, the most serious problem of GA can be solved, and we can use GA more effectively. Evolvable Hardware is based on this concept, and utilizes the robust capability of GA by reducing its computational cost.

Brief Summary of Evolvable Hardware

Evolvable Hardware (EHW) is a hardware device which is built on software-reconfigurable devices, e.g. PLD (Programmable Logic Device) and FPGA (Field Programmable Gate Array). The hardware structure of EHW can be reconfigured autonomously by using GA in order to attain better hardware performance in a changing environment (Higuchi et al. 1992). The hardware structure is determined by downloading a binary bit string into the device. This string is called *architecture bits*, and it is regarded as chromosome of GA as shown in Fig.1. Once a good chromosome is found by GA, it will be downloaded into software-reconfigurable devices.

EHW inherits two important features: fast computation of hardware device and adaptability of GA. EHW has three advantages over the traditional hardware and software systems (Yao and Higuchi 1996): First, as mentioned above, EHW can *autonomously* improve the performance by changing its hardware configuration with GA. Second, it processes the information much faster than the software system, and can realize the *real-time* computation. Third, the reconfigurable-device can change its functionality in an *on-line* fashion during execution. EHW can also deal with a new application area to which the inflexible traditional hardware systems are not efficient (Murakawa et al. 1996, Bennett et al. 1996). In particular, EHW exhibits a superior performance in dynamically changing environment because it can reconfigure its hardware structure to adapt to the changes. The image compression is also one of them. Depending on the nature of the image data to be compressed, EHW implements the most effective compression configuration in its hardware structure.

Electrophotographic Printing

This section describes the motivation to apply EHW to the electrophotographic (EP) printer as the data compression system.

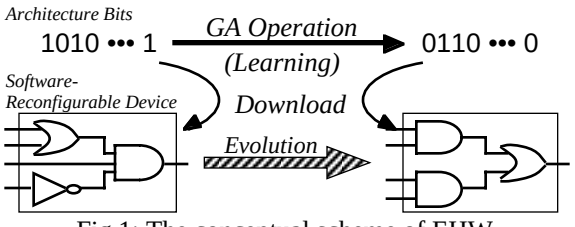

Fig.1: The conceptual scheme of EHW.

Electrophotographic Printing, Halftone Dot Image and Data Compression

EP printing is a new technology in the field of printing and publication, and we can easily have many printed matters of photo-quality at anytime with very low cost. The EP printer is required to be much smaller than the offset printers, and to work much faster than color copier.

In the EP printing, the original data created on the desktop publishing (DTP) systems is divided into four halftone dot images, which are binary image data and correspond to four colors (cyan, magenta, yellow and black). In halftone images, gray levels are represented using binary pixel patterns. The total size of the four images is much larger than the size of the data in the DTP system. One A4-size color image of 1200 dpi requires about 70 Mbytes for storage. A digital printer prints hundreds sets of hundreds different pages in the speed of 100 page/min. Therefore, it must have huge memory devices and very fast data-transfer bus. In order to reduce the costs for storage and communication, the halftone dot images must be compressed as much as possible and the compressed data must be expanded faster than the printing speed. Besides, the images must be compressed in the lossless way in which the expanded data is completely the same as the original data.

Many techniques have been proposed for lossless image compression in the field of information processing. Some of them are realized only by software, but they cannot satisfy the speed requirement of the EP printer. The others are implemented in a chip and work very fast in compression and decompression. They are used in various industrial products, such as facsimile. However, they are not suitable for the EP printer, because their adaptation mechanisms are too simple for the complex characteristics of the halftone dot image. It is very difficult for them to realize the efficient compression and the fast compression & decompression, simultaneously.

Therefore, this paper proposes to apply EHW to the EP printer as the data compression unit. Since EHW is implemented in a chip, it can compress and decompress image data very quickly. Moreover, since EHW has powerful ability of adaptation caused by GA, the compression chip with EHW can efficiently compress the halftone dot images in the printer.

Compression of Halftone Dot Image

Currently, the EP printer uses the data compression chips based on Lempel-Ziv method. Its advantage is fast compression and decompression, but its compression ratio is poor because Lempel-Ziv method is not theoretically optimized for compressing image data.

Image data consists of many pixels, and each pixel tends to tightly relate with its neighboring pixels. Then, we can predict the pixel value using the values of its neighboring pixels, and we don't have to represent the pixel if

Table 1: Two components of the proposed method and their two function modes.

		Learning Mode	Compression Mode
Template Generator	Statistical Calculation Genetic Algorithm	Search for best template	---
	Template Memory	Store discovered templates	Output stored templates
Data Compressor		Calculate compression ratio	Output compressed data

the prediction is correct. Namely, the precision of prediction strongly influences the compression ratio. The pattern of the values of pixels used for prediction is called the *context*. The large context allows the precise expectation, but it increases the computational cost. Therefore, we must carefully select the *template* which is a set of relative locations of the reference pixels from the currently coded pixel. However, the different images have different optimal templates, and the template may have to change even in one image to improve the compression ratio.

JBIG (Joint Bi-level Image coding experts Group), which is the international standard method specialized to bi-level image data, basically adopts the method mentioned above (CCITT 1993). It has a template consisting of 10 reference pixel positions (Fig.2). In the figure, black and hatched rectangles respectively represent the positions of a currently coded pixel and the reference pixels. A hatched rectangle with thick edges, called the *adaptive template*, can move in the area indicated by the rectangles with crosses during compressing one image to capture any variable structure that might exist in the image, using the simple statistical calculation (Sayood 1996).

JBIG is developed to mainly compress the binary facsimile images, but the halftone dot images of EP printing have quite different characteristics. In the halftone dot images, the adjacent pixels don't relate with each other strongly. Therefore, JBIG can not compress those images well, because (1) the configuration of template is not suitable, and (2) the simple mechanism to move the adaptive template can not work well (Forchhammer 1993). Therefore, to resolve the above two difficulties, EHW is applied to the data compression system of the EP printer. The chip implementing this method will compress the halftone dot images much better than others without affecting the compression speed. The next section explains this approach in details.

Lossless Image Data Compression by Genetic Algorithm

As shown in Fig.3, the proposed method in this paper consists of two parts, the template generator (TG) part with GA and the data compressor (DC) part (Salami 1998). This section explains them, and shows the result of simulations.

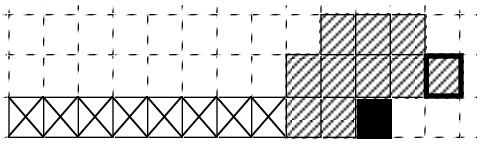

Fig.2: Configuration of reference pixels (template) of JBIG.

Data Compressor Part

This system has two operating modes, the learning mode and the compression mode (Table 1). In both modes, the image data is divided into units named the *stripe*, consisting of L lines of an image. In the learning mode, DC part searches for the optimal template of each stripe by GA, and the discovered templates are stored in the memory. Using those templates discovered in the learning mode, the compression mode compresses the image data. In the compression mode, GA doesn't work for generating templates, and the stored templates are used for compressing the corresponding stripes.

DC part also consists of 2 components, the context selector and QM-Coder (Pennebaker et al. 1988). The context selector determines the contexts corresponding to the coded pixels, using the template sent from TG part (as mentioned in the next sub-section). Fig.4 illustrate an example of the template. Similar to Fig.2, black and hatched rectangles respectively represent the positions of a currently coded pixel and the reference pixels. QM-Coder sequentially encodes all pixels in a stripe using the contexts corresponding to them.

DC part is much more flexible than JBIG. It can change the relative locations of *all* reference pixels, while JBIG changes only one. Besides, the pixels can separately move within much wider area of 32x8 (Fig.4). Because of the these enhancements in the context selection, this system can easily deal with the halftone dot image where the adjacent pixels don't relate with each other. It can also compress the ordinal facsimile images similar to JBIG.

Template Generator Part

As mentioned in the previous section, the compression ratio is strongly influenced by the template specifying which pixels are used to generate contexts. TG part searches for the optimal template in every stripe using GA.

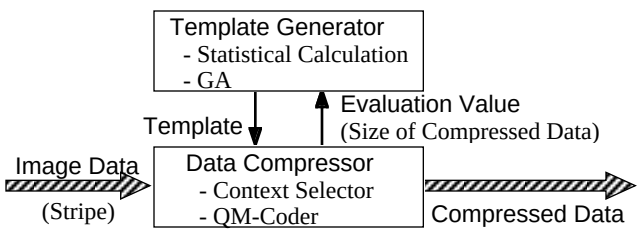

Fig.3: Framework of the proposed system.

A template is defined to have 10 pixel locations, and each location is selected from the 32x8 area. One location can be represented by 8 bits ($2^8 = 32 \times 8$), and 80 bits are required to indicate a template with 10 locations. In the rest of this paper, the 8 bits specifying one location is called the *slot*.

In the learning mode, the initial population of GA is generated using the *initial template*, which is prepared by the simple statistical calculation on the first stripe, based on the following steps:

1. The default template of JBIG is set as the initial template.
2. The compression ratio is calculated by using the initial template.
3. Among 10 slots in the initial template, the most useless slots are determined. (The most useless slots specifies the position of the reference pixel which records the largest compression ratio even if it is removed from the template.)
4. Among all possible locations in the 32x8 area, the most efficient location is selected. (The most efficient location records the largest compression ratio when it is assigned to the previously selected slot.)
5. The new template is created by assigning the selected location to the selected slot.
6. If the compression ratio using the new template is better than the initial template, the initial template will be exchanged by the new template and go to step-3.
7. Finish the procedure.

The discovered initial template is copied to one chromosome in the population of GA, and its mutated copies are assigned to other chromosomes. This procedure is simple and the result of the statistical calculation is not always the best template, but it accelerates the search speed of GA, compared with the random initial population.

The initial template discovered by the statistical calculation is stored in the template memory and GA runs from the second stripe. In each stripe, the chromosomes in the population are sent to the context selector one by one as templates and the size of the compressed stripe from QM-Coder will be returned as the fitness value. The number of generations for one stripe is represented by the parameter G, and GA searches for the optimal template minimizing the size of the compressed stripe. The best template discovered by GA is stored in the template memory, and GA moves to the next stripe. The stored templates in the template memory are used for compressing their corresponding stripe in the compression mode.

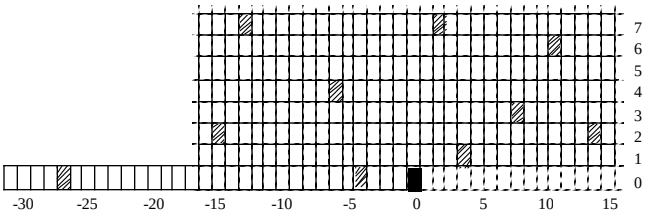

Fig.4: Configuration of reference pixels (template) of the proposed system.

Enhanced Representation

From the viewpoint of GA, the data compression is a nonstationary problem because the target stripe for compression changes periodically. It is well known that the nonstationary problem is difficult for GA (Grefenstette 1992). Using the structured GA (Dasgupta and McGregor 1992), therefore, chromosome representation is enhanced to allow GA effectively follows the changing environment.

The basic idea of this enhancement is to keep the previously good templates in the population even if the environment changes suddenly. In one image, there may be many local areas with similar textures, and their optimal templates tend to have many common locations of reference pixels. Hence, it is important for the chromosomes in the population to keep the previously good locations for the next stripes.

The enhanced representation have $M \geq 10$ locations of the reference pixels (slots), and M additional bits indicating which slots are active to generate a template (Fig.5). The inactive slots are never influenced by the genetic operations except the selection, and can keep the previously good locations. The active slots are changed to search for the better locations of the reference pixels. Additionally, two more genetic operations are introduced to handle the inactive slots, *slot copy* and *slot swap*. Using slot copy, the contents of one randomly selected active slot is copied to one of the inactive slots. The slot swap operation chooses one active slot and one inactive slot at random, and exchange their activation bits. These two additional operations of the slot copy and the slot swap correspond to memorization and remembrance. The probabilities that a chromosome is changed by these operators are given by two parameters, the slot-copy ratio and the slot-swap ratio.

Simulation Results

This section shows the results of some computational simulations, before explaining the the architecture of the EHW chip implemented for the data compression.

For the simulation, we prepare a printer image (halftone dot image; JIS/ISO standard color image data N2), and 8 facsimile images (standard images of CCITT (International Telegraph and Telephone Consultative Committee)). The printer image is divided in 8 sub-images, and they are compressed separately.

In order to compare the performance of the proposed system, the following 4 methods are also applied to the images: (1) JBIG, (2) Lempel-Ziv method ("compress" command of Unix), (3) the statistical method, and (4) the sequentially exhaustive search. Among them, the simple statistical method (3) are same as the proposed method, but it doesn't use GA. In the method (3), after the statistical calculation in the first stripe, the same template is used for compressing the following stripes. In the method (4), only

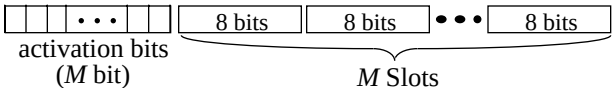

Fig.5: Enhanced representation of chromosome.

Table 2: Compression ratios of various method.

	Printer Image	Fax Image
(1) Lempel-Ziv	3.34	8.41
(2) JBIG	3.35	14.67
(3) Statistical calculation	6.38	19.49
(4) Exhaustive search	6.61	---
Proposed method	6.52	19.82

Table 3: Parameter setting of the proposed system.

Total generation	162	Population size	20
Tournament size	2	Crossover ratio	0.8
Mutation ratio	0.03	Number of slots	15
Slot-copy ratio	0.1	Slot-swap ratio	0.1
Number of lines in one stripe (L)			40
Number of slots in enhanced representation (M)			15

one location in a template can be optimized by the exhaustive search. The search area is same as the proposed method. The reason why two or more locations can not be changed at once is to avoid the combinatorial explosion.

Table 2 shows the compression ratios of the above methods. The parameter setting of the proposed system and GA is shown in Table 3. Unfortunately, the compression ratio of the proposed method on the printer image is worse than the method (4), but it can finish the compression much faster than the method (4), as mentioned below. The following 3 important facts can be concluded from the simulations: First, among method (1), (2) and (3), the method (3) shows the best performance for compressing both images. This fact means that the enhanced context selector introduced for the halftone dot images can deal with the ordinal facsimile images. Namely, the proposed context selector has the ability to process the general image data.

Next, the proposed method can exhibit better compression ratio than the method (3). The only difference between their procedures is that the proposed method executes GA after the second stripe. The improvement of the performance demonstrates the efficiency of the adaptability of GA during compressing one image. Under the parameter setting of Table 3, the difference between two methods are very small, but the compression ratio of the proposed method is expected to improve by more generations and larger population. When it is implemented in the hardware, we may ignore such increased computational costs.

Finally, the proposed method presents less compression ratio for compressing the printer data than the method (4). Table 4 shows the more detailed results, and we can understand the proposed method exhibits better compression ratio on 4 out of 8 images. It means that the search performance of the GA is not always inferior to the sequential exhaustive search method. Moreover, from the viewpoint of the computational cost, the proposed method can finish the task about *ten times* faster than the exhaustive search.

Data Compression Chip

We are developing the data compression chip based on the method explained in the previous sections.

Fig.6 illustrates the architecture of the EHW data-compression chip. The chip mainly consists of NEC V830 RISC processor (32 bit, 100 MHz), the data compression hardware, and 3 registers. Through the registers, V830 and the compression hardware can communicate with each other. V830 controls the chip, runs GA calculation in the learning mode, and interfaces with the host computer.

The data compression hardware have 5 components and 3 controllers (Fig.7). The image data is temporally stored in the line memory of 10240 bits x 10 lines and sent to the reference buffer. It sequentially send the currently coded pixels and extracts the corresponding 32x8 pixels sent to the context generator. Using the template, the context generator creates contexts from the received 32x8 pixels. The contexts are sent to QM-Coder with the currently coded pixels. In the compression mode, QM-Coder sends a sequence of the compressed data to the external data storage and outputs the size of the compressed data to V830 in the learning mode.

Those components are configured to process the image data as fast as possible, because it must calculate the evaluation values of GA in the learning mode. In the case of the computer simulation in the previous section, the evaluation of chromosomes in the population spent about 95% of the total computational time. Therefore, by implementing the data compression part in hardware, the total speed of this chip including the learning mode becomes drastically fast. For example, in the above simulations, the proposed method spent about fifty minuets for learning and compressing a 1184x6464 image on the Ultra Sparc Station (144 MHz). On the contrary, assuming that QM-Coder works at 40 MHz, the hardware can finish the same process within a minute. The volume of the hardware for the proposed system is greater than JBIG system, but the extra hardware allows the system to search for the optimal template adaptively. This chip can not expand the compressed data yet, but we are starting to design the new chip which can compress the image data and expand the compressed data.

Conclusions

This paper described a data compression chip for the high-precision electrophotographic printer using Evolvable Hardware (EHW), which is a new hardware paradigm combining Genetic Algorithm and the reconfigurable hardware. The computer simulations showed that the EHW-based system can exhibits much better compression ratios than the international standard.

The salient contribution of this paper is the reveal of the true importance of hardware reconfigurability. While software-reconfigurable devices have been mainly used for prototyping and reducing the manufacturing cost, the EHW-chip demonstrates that hardware reconfigurability is indispensable to satisfy the severe performance requirement. Thus, this paper shows the new direction of reconfigurable computing.

Table 4: Compression ratio of Exhaustive search and proposed method in divided printer images.

	1	2	3	4	5	6	7	8	Average
Exhaustive search	15.66	6.34	4.47	4.50	4.86	6.02	7.6	25.00	6.61
Proposed method	16.08	6.566	4.57	4.55	4.74	5.81	6.77	23.48	6.52

Acknowledgements

This work is supported by MITI Real World Computing Project (RWCP). We thank Dr. Otsu and Dr. Ohmaki in Electrotechnical Laboratory, and Dr. Shimada in RWCP for their support.

Reference

Bennett III, F. H., Koza, J. R., Andre, D. and Keane, M. A. 1996. Evolution of a 60 Decibel Op Amp Using Genetic Programming, *Evolvable Systems: From Biology to Hardware*, 455-469, Springer.

Dasgupta, D. and McGregor, D. R. 1992. Nonstationary Function Optimization using the Structured Genetic Algorithm. *Parallel Problem Solving from Nature, 2*: 145-154. Elsevier.

Davis, L. ed. 1991. *Handbook of Genetic Algorithms*: Van Nostrand Reinhold.

Forchhammer, S. 1993. Adaptive Context for JBIG Compression of Bi-Level Halftone Images, Proc. of the 1993 Data Compression Conference: 431 IEEE Computer Society Press.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*: Addison-Wesley.

Higuchi, T., Niwa, T., Tanaka, T., Iba, H., de Garis, H. and Furuya, T. 1992. Evolvable Hardware with Genetic Learning, *Proc. of Simulated Adaptive Behavior*. The MIT Press.

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

Grefenstette, J. J. 1992. Genetic Algorithms for changing environment. *Parallel Problem Solving from Nature, 2*: 137-144, Elsevier.

International Telegraph and Telephone Consultative Committee (CCITT). 1993. *Progressive Bi-level Image Compression*, Recommendation T.82.

Murakawa, M., et al.. 1996. On-line Adaptation of Neural Networks with Evolvable Hardware, *Proc. of the 7th International Conference on Genetic Algorithms*, pp.792-799, Morgan Kaufmann.

Pennebaker, W. B., Mitchell, J. L., Langdon Jr., G. G. and Arps, R. B. 1988. An overview of the basic principles of the Q-coder. *IBM Journal of Research & Development* 32 (6): 717-726.

Salami, M. et al.. 1998. On-Line Compression of High Precision Printer Images by Evolvable Hardware. *Proc. of the 1998 Data Compression Conference*. IEEE Computer Society Press.

Yao, X. and Higuchi, T. 1998. Promises and Challenges of Evolvable Hardware. *IEEE Transactions on Systems, Man, and Cybernetics*, Part C, 28 (4).

Sayood, L. 1996. *Introduction to Data Compression*. 475: Morgan Kaufmann.

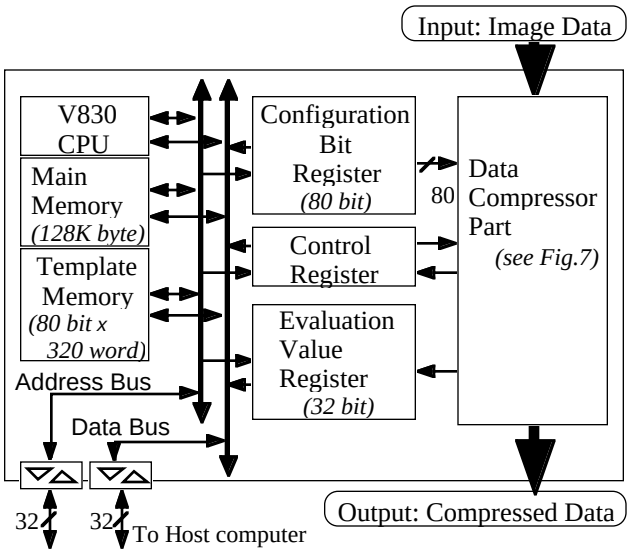

Fig.6: Architecture of data compression chip with EHW.

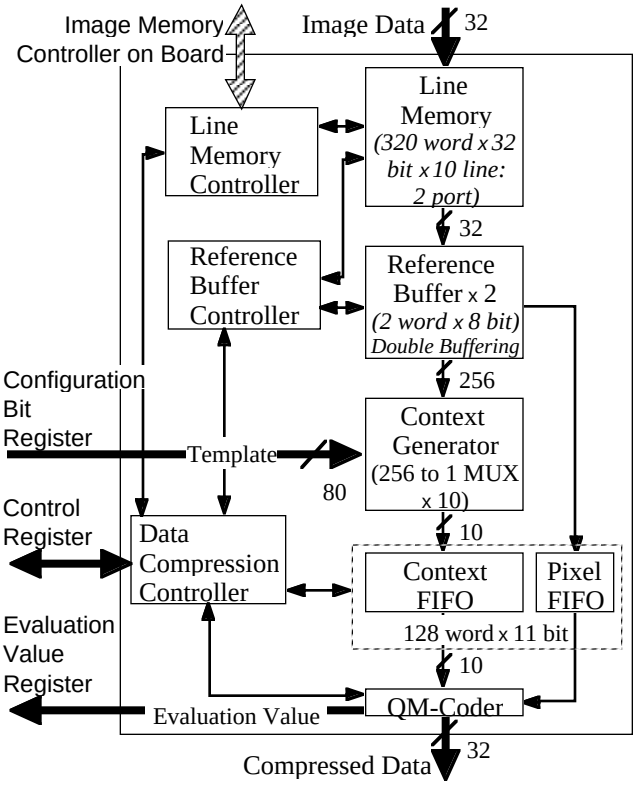

Fig.7: Architecture of data compressor part.