

This article was downloaded by:[University of Birmingham]
On: 9 December 2007
Access Details: [subscription number 777121144]
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954
Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



Connection Science

Publication details, including instructions for authors and subscription information:
<http://www.informaworld.com/smpp/title~content=t713411269>

A novel and practicable on-chip adaptive lossless image compression scheme using intrinsic evolvable hardware

Jingsong He^{ab}; Xin Yao^{acd}; Yunbi Chen^{ab}

^a Nature Inspired Computation and Applications Laboratory, China

^b Department of Electronic Science and Technology, University of Science and Technology of China, China

^c Department of Computer Science and Technology, University of Science and Technology of China, China

^d School of Computer Science, University of Birmingham, UK

Online Publication Date: 01 December 2007

To cite this Article: He, Jingsong, Yao, Xin and Chen, Yunbi (2007) 'A novel and practicable on-chip adaptive lossless image compression scheme using intrinsic

evolvable hardware', Connection Science, 19:4, 281 - 295

To link to this article: DOI: 10.1080/09540090701725508

URL: <http://dx.doi.org/10.1080/09540090701725508>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.informaworld.com/terms-and-conditions-of-access.pdf>

This article maybe used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

A novel and practicable on-chip adaptive lossless image compression scheme using intrinsic evolvable hardware

JINGSONG HE*†‡, XIN YAO†§¶ and YUNBI CHEN†‡

†Nature Inspired Computation and Applications Laboratory, China

‡Department of Electronic Science and Technology, University of Science and Technology of China, China

§Department of Computer Science and Technology, University of Science and Technology of China, China

¶School of Computer Science, University of Birmingham, UK

Adaptive lossless image compression is one of the most important applications in the field of evolvable hardware (EHW). However, related studies in the past focused on implementations with extrinsic EHW, which uses a host computer to run software simulation and compiling, and then download the final circuit to the silicon chip. This is not suitable for tasks of on-chip adaptation. This paper presents a novel technique to reformulate the problem as a task of evolving a set of switches. As a result, the whole scheme can be implemented easily using intrinsic EHW. In order to enhance the scalability of the whole scheme, a strategy based on data-decomposition and pyramidal fitness evaluation strategy is developed for evolving larger scale images. Software simulation shows that the proposed method can largely reduce the computation time, and can scale up the image size up to 70 times with relatively slow increase in computation time. Hardware simulation shows that the method can be applied in practice.

Keywords: Evolvable hardware; Lossless image compression; Evolutionary computation; Scalability

1. Introduction

As a newly emerged field, evolvable hardware (EHW) may provide new mechanisms for automatic circuit design and optimization, as well as for novel designed circuits to adapt the environments by themselves. According to their definitions and purposes, the executive modes of EHW can be divided into two types, *i.e.*, the intrinsic EHW and the extrinsic EHW (Yao and Higuchi 1999). In general, an extrinsic EHW was thought to be more suitable for automatic circuit design and optimization (Hemmi *et al.* 1997), while an intrinsic EHW was thought to be more suitable for online adaption and deep exploration of novel circuits (Thompson and Wasshuber 2000). Early studies on modes of EHW focused on such aspects as the role they can play in the combination of evolutionary computation and hardware design

*Corresponding author. Email: hjss@ustc.edu.cn.

(Hemmi *et al.* 1997, Thompson *et al.* 1999), the ability they have (Stoica *et al.* 2001), and the problems they will face in the future (Montana *et al.* 1998, Yao and Higuchi 1999, Stoica *et al.* 2001). Recent studies tend to pay more attention to seeking valuable applications of EHW, and to discover new problems and their corresponding solutions. It seems the best way of research in the field of EHW is to combine intelligent computation with real-world applications.

The adaptive lossless image compression is a typical application of EHW in the past years. Higuchi *et al.* (1997) first presented an accomplishment of such a task of compressing images on a special functional FPGA (F²PGA) with their special variable length genetic algorithm (VGA). This has been regarded as merely the non-toy problem in the early researches in the field (Fukunaga *et al.* 1998). Thereafter, they have performed continuous researches on the problem of evolving more large scale images (Sakanashi *et al.* 2001, 2004). In a different study, Fukunaga *et al.* (1998) proposed a new prototype system, in which genetic programming (GP) was used as the evolving engine and the Lisp S-expression was used as coding mechanism, so as to solve the problem of such implementation on conventional FPGA. Through improving the GP-based method He *et al.* (2005) scaled up the processable amounts of data from 64 KB to 2 MB, and reduced the computing time from 2 h (Fukunaga *et al.* 1998) down to 5 min (in simulation mode) at the same time. Unfortunately, no work is concerned with the prime problem of how to execute the evolutionary procedure on-chip.

Both the systems mentioned above belong to the type of extrinsic EHW: the VGA (Higuchi *et al.* 1997) for optimizing templates is executed on a host computer (Sakanashi *et al.* 2001), and so is the GP engine used in Fukunaga *et al.* (1998). Nevertheless, a common problem for these two modes is the large time-cost of compiling. For instance, the compiling time of converting a chromosome in Lisp S-expression to an assessable circuit is usually about 0.5 h (Fukunaga *et al.* 1998). In view of this reason, they are not suitable for real-time image compression, as there are limitations in their functionality.

This paper proposes a novel and simple evolutionary technique for predictive lossless image compression. In our approach, a genetic algorithm with small size of populations is introduced to reduce circuit resources, and the parameters of the predictive function are discrete. Therefore, the chromosome can correspond to a set of circuit switches, and thus can be evolved on chip directly. Experimental results show that the proposed method can not only process more large images, but also reduce the computing time efficiently.

2. Related work and problem analysis

The compressibility of an image lies on the correlation between neighboring pixels. For an image of size $m \times n$, we denote the horizontal site by i , the vertical site by j , and the pixel value at (i, j) by $x_{i,j}$. For convenience, we also describe the order of pixels with raster scan as in figure 1. For example, the current pixel and its neighbors are labeled by as $x, x_1, x_2, x_3, x_4, \dots$, to represent $x_{i,j}, x_{i,j-1}, x_{i-1,j-1}, x_{i-1,j}, x_{i-1,j+1}, \dots$, respectively. The predictive value of the current pixel is denoted by x' , and the predictive error by $\delta_{i,j}$ ($\delta_{i,j} = x - x'$). Thus, the error matrix between the original image and the predicted image could be obtained and denoted by Δ . The smaller the entropy of Δ , the less the averaged length of Huffman code. From figure 1, it is clear that the closer the pixel is to the current one, the stronger is the correlation between them. Fukunaga *et al.* (1998) demonstrated that using the four-neighbor predictive mode, *i.e.*, using x_1, x_2, x_3 , and x_4 to predict x described in figure 1, is able to make the results good enough.

Note that the symbols of chromosomes used by Lisp s-expression in Fukunaga *et al.* (1998) and by VGA in Higuchi *et al.* (1997) have similar meaning: each symbol corresponds to

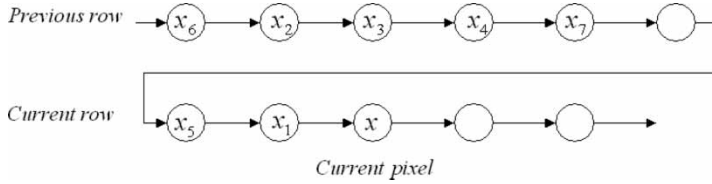


Figure 1. The current pixel x and the arrangement of its neighboring pixels.

a circuit block. Here we use a chromosome in Lisp S-expression as an example to explain what problems the extrinsic EHW encounters. In Fukunaga *et al.* (1998), the variable set $\{x_1, x_2, x_3, x_4\}$ and the arithmetic operation set $\{+, -, \times, \div, \min, \max\}$ have been taken as the basic elements for GP to approach the original image. The chromosome, *i.e.*, the coded predictive function, can be illustrated by the Lisp S-expression as in figure 2, where each element in the variable set corresponds to a circuit block. The procedure is to evolve into a satisfactory predictive function (still a coded expression at this time) on a host, then decode the chromosome, and finally compile it on to FPGA.

Problems of compiling from the chromosome expression with VHSIC Hardware Description Language (VHDL) evolved by GP to the executive circuit were investigated (Montana *et al.* 1998), and the scalability of the evolutionary model was emphasized. For a processable system, the scalability implies that while it is good to start with small problems with small amounts of data, the switch to large problems with large amounts of data should not be overly difficult. Improving evolutionary algorithms can indeed scale up processable amount of data (He *et al.* 2005), but can not reduce the compiling time down to FPGA. For the adaptive lossless image compression, since it demands a high real-time performance in real-world applications, the following problems are obviously important and urgent.

First, in the extrinsic model, the chromosome is usually a symbolic string as the Lisp S-expression used in figure 2. A long time (usually 0.5 h) is needed to compile the final chromosome and download it to FPGA. Thus, how to implement the task of predictive lossless image compression by taking advantage of intrinsic EHW is a topic of both theoretical interest and practical significance.

Second, it is hard to control the size of the chromosome with extrinsic EHW mode, either by the dendriform expression used in GP or by the variable length chromosome used in VGA. The main difficulty is due to the process of compiling the chromosome to chip, because the usable resource on a chip is limited while the circuit needed for a real-world problem is usually unknown. Therefore, how to convert the unknown numbers of circuit resource of a real problem to an expression with finite circuit resource seems to be the most important issue.

$$\{ - \{ + \{ - x_4 \ 0.49115 \} \ - 0.6769 \} \\ \{ / \{ + \{ - \{ + \{ + x_2 \\ \{ - x_4 \\ \{ - x_1 \ x_1 \} \} \} x_4 \} \} x_1 \} x_4 \} \\ \{ + \{ + \{ / \ - 0.06982 \ - 0.18761 \} \\ \{ + x_4 \ - 0.46071 \} \} \\ \{ / x_4 \\ \{ + \ - 0.73187 \ x_4 \} \} \} \}$$

Figure 2. An example of the chromosome of GP (Fukunaga *et al.* 1998) using the Lisp S-expression, where each symbol corresponds to a circuit block.

Third, the parallel computation on a chip is a characteristic of intrinsic EHW which endows the evolvable chip with the ability to adapt to changes in circumstances and tasks. In general, it is favorable to maintain a relative large number of populations to keep the diversity of individuals in evolution. For intrinsic EHW, however, too large a population size is sometimes detrimental to reducing the number of circuits. This can be exemplified by the work of Fukunaga (Fukunaga *et al.* 1998), where the population size is such that 2000 individuals are involved. If the 2000 individuals are evolved and evaluated simultaneously, the size of chip should be enlarged 2000 times than that being done serially. Obviously, minimizing the size of populations in evolutionary computation is a new and valuable problem in real-world applications.

3. Lossless image compression using intrinsic EHW

The characteristic of extrinsic EHW is the use of symbolic expression in the chromosome, where each symbol corresponds to a real circuit completed by conventional design. Though this mode has the advantage that many commercial compilers and well-designed circuit resources can be employed directly, it has the disadvantage in real-time control and adaption that its executable mode may be detrimental to online tasks. This is because an extra device is needed for extrinsic EHW to accommodate a compiler software, and much time is consumed in compilation and downloading. To overcome these drawbacks of extrinsic EHW, we study methods and algorithms which are suitable for implementation with intrinsic EHW.

3.1 The binary coding mechanism

According to the description and analysis in section 2, if the control parameters of the predictive function could be described as a set of switches, the evolving of states of the switches will be easily achieved on a chip, and thus the intrinsic EHW mode can be used. Since the four-neighbor prediction mode has been used for adaptive lossless image compression successfully (Fukunaga *et al.* 1998, He *et al.* 2005), it will still be used here. To obtain a chromosome of fixed length with binary coding, we use exponential function to approach the predictive function. By using interpolating function with the four-neighbor predictive mode, the predictive function is written in the form of

$$x'_k = \sum_{i=1}^4 x_{k,i} e^{-\alpha_i d(x_k, x_{k,i})}, \quad (1)$$

where x'_k is the k th predicted value, $x_{k,i}$ is the pixel value of the i th neighbor of x_k , α is a parameter of the interpolating function, and $d(x_k, x_{k,i})$ stands for the distance between the current point and its i th neighbor. Therefore, the task of prediction is equivalent to minimize the following objective function

$$f(x) = \sum_{k=1}^N \|x_k - x'_k\|^2 = \sum_{k=1}^N \left[x_k - \sum_{i=1}^4 x_{k,i} e^{-\alpha_i d(x_k, x_{k,i})} \right]^2 \quad (2)$$

where N is the number of pixels for prediction. For convenience, it is assumed that the interpolating function only affects its neighboring pixels, *i.e.*, $d(x_k, x_{k,i}) = 1$. Under this condition, equation (2) is reduced to

$$f(x) = \sum_{k=1}^N \left[x_k - \sum_{i=1}^4 x_{k,i} e^{-\alpha_i} \right]^2. \quad (3)$$

Thus the lossless image compression becomes a problem of parameter optimization with four variables $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. If the problems of hardware implementation analyzed in section 2 are not concerned, there are many classical and advanced evolutionary algorithms which can be used to optimize these parameters, *e.g.* the genetic algorithms, the simulated annealing, the evolutionary programming, the evolutionary strategies, the differential evolution (DE) (Storn and Price 1995), and the particle swarm optimization (Kennedy and Eberhart 1995), etc. However, the task of optimization problem in the field of EHW (especially in intrinsic EHW mode) is quite different from those implemented by software.

To make the problem of equation (3) easily implemented by intrinsic EHW, we let the parameters in equation (3) be coded with a set of binary string. The values of the exponential functions thus can be calculated on chip by querying an embedded look-up table (LUT). For example, for the values of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ we can use '101, 101, 011, 100, 101, 100, 010, 101' to represent '0.8125, -1.7500, 0.7500, -1.3125', respectively. Then the candidate solution of the predictive function in equation (1) can be expressed by a linear function $x' = 2.25353478721321x_1 + 0.173773943450445x_2 + 2.11700001661267x_3 + 0.269146348729184x_4$. In this case, the size of LUT is equal to $2^{L/4}$, with L being the length of the binary string. Therefore, such a method of coding leads to the task of predictive lossless image compression being implemented easily on a chip. The following issues are evident:

- (1) The implementation of the task of optimizing parameters in equation (1) is very easy on a chip with small circuit resources, and no compilers are needed. This implies that such an implementation mode of EHW belongs to an intrinsic mode, rather than the extrinsic mode as in Higuchi *et al.* (1997) and Fukunaga *et al.* (1998).
- (2) The expression of binary string offers better accuracy to express the real number of parameters than are used as the parameters of a linear predictive function. Since the length of bit string is fixed, the exponential function can be easily implemented by hardware through querying a table. For querying the table, the number of values listed is equal to $2^{L/4}$.

3.2 The binary evolutionary programming algorithm

For the sake of being easily archived on silicon with smaller circuit resources, a binary coded expression and a reduced evolutionary algorithm are needed. Two important notions are helpful in achieving a binary evolutionary programming algorithm. One is the technique of asexual reproducing for local selection presented in StGA (Tu and Lu 2004), and the other is the idea that long jumps can help to generate an offspring at the neighborhood of the global minimum, as discussed in IFEP (Yao *et al.* 1999). To reduce circuit resources as much as possible by minimizing the population size, these two notions are fabricated here to archive the binary evolutionary programming algorithm.

Technically, the asexual reproducing can repeat one by one with a series of compounding mutations as shown in figure 3, where h_0 is the parent, $h_1 - h_5$ are offsprings, p_{m1} and p_{m2} are probability for mutations. Empirically, p_{m1} and p_{m2} can usually take $1/3$ and $1/9$, respectively. As a result, the averaged differences (search step size with Hamming distance) between the parent and each of the offspring can be obtained as $1/3, 1/9, 4/9, 1/27$, and $16/81$ of the length of the bit string, respectively. It is apparent that the characteristic of the local selection mechanism in StGA is absorbed, but the random generation method is substituted into a mixed search with five mean step size of mutation. The characteristic of using two kinds of mean step size (Gaussian mutation and Cauchy mutation) for the mutation presented in Improved Fast Evolutionary Programming (IFEP) is also introduced, but jumps have been made more times and more deeply.

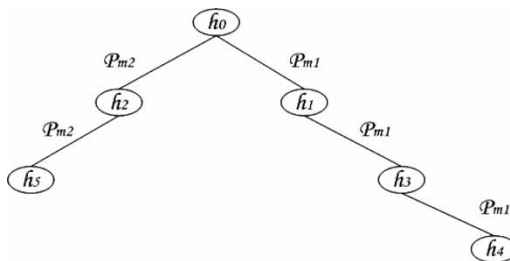


Figure 3. The illustration of the asexual dendriform reproducing, where h_0 is the parent, h_1 , h_2 , h_3 , h_4 , and h_5 are offsprings, p_{m1} and p_{m2} are the probability of mutations.

Since the local dendriform reproducing (shown in figure 3) involves the mixing of large and small mutations, it can be thought to be a hybrid mechanism between local and global searches. For scalability reasons, here we just take it as an independent evolution procedure, and denote it as BinEP. The basic procedure of BinEP is summarized as follows.

Step 1 Initiate a binary string of L -bits randomly, denote it as h_0 and let $C = 1$. Each bit of h_0 is taken as a circuit switch for controlling the real number of parameter in the predictive function. Evaluate the fitness of h_0 .

Step 2 Generate five offsprings asexually as follows:

mutate h_0 with the probability $p_{m1} \rightarrow h_1$;

mutate h_0 with the probability $p_{m2} \rightarrow h_2$;

mutate h_1 with the probability $p_{m1} \rightarrow h_3$;

mutate h_3 with the probability $p_{m1} \rightarrow h_4$;

mutate h_2 with the probability $p_{m2} \rightarrow h_5$.

Here p_{m1} and p_{m2} ($1 > p_{m1} > p_{m2}$) are the probability of mutation happening on each binary bit with the values changing from 0 to 1 or from 1 to 0.

Step 3 Evaluate the fitness of each offspring by equation (3).

Step 4 Select the best one from $\{h_1, h_2, h_3, h_4, h_5\}$ and denote it as h'_0 . If h'_0 is better than or equal to h_0 , it is chosen in replacement of h'_0 substitute h_0 to be the parent of next generation.

Step 5 Repeat the procedure from step 2 with $C = C + 1$, until the halting criterion is satisfied.

From the above five steps we can see that step 3 is related to the scalability problem defined by Montana *et al.* (1998). That is, the larger the image size, the larger the amount of processing data, and thus the longer the computation time for fitness evaluation. We use here a kind of pyramidal fitness evaluation method by introducing the sampling and interpolating mechanisms into step 3. Both mechanisms are basic concepts in the field of signal processing, and can be used to solve the problem of scalability simply and directly. To reduce the computation time on large amount data, the method is as follows.

Step 1 Separate the predictive area of a $m \times n$ sized image into a set of $l \times k$ templets, where $l < m$ and $k < n$.

Step 2 Take the set composed of the mean value of each $l \times k$ templet as the input data, and then execute BinEP.

Step 3 If the fitness has not been improved after some generations, reduce l and k randomly and then go to step 2. The procedure stops if the fitness has not been improved after some generations, till $l = 1$ and $k = 1$.

We denote the combination of the BinEP with the above technique as the extended BinEP (EBinEP). Obviously, in EBinEP changing of input data can result in the transition of the

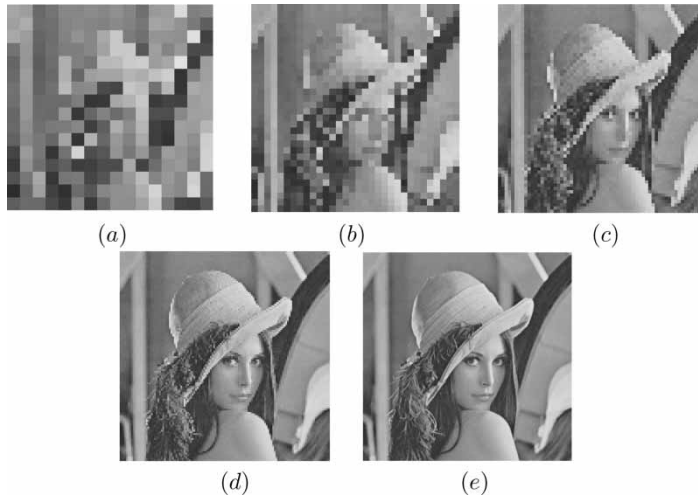


Figure 4. Images of Lena from coarse granularity to fine granularity, where (a), (b), (c), (d) are results for different sampling scales from the original image (e).

environment (*i.e.*, input data of fitness evaluation) from coarse granularity to fine granularity. This situation can be elucidated with figure 4, where different sampling scales correspond to different evaluation environments. An optimization can start from a coarse image, and gradually approach to the original image according to evolutionary progress. Since the coarse expression of search space can make the objective problem more easily, EBinEP supplies a greedy and gradual approach through disposing the scalability problem at the same time.

4. The accomplished structure of intrinsic EHW

With the proposed binary evolutionary programming algorithm, we can implement the task of adaptive lossless image compression on a FPGA chip by using hardware description language. The hardware design is accomplished on our development platform with Altera Cyclone EP1C12Q240 FPGA chips, and the clock is of 50 MHz. The EHW engine architecture is shown in figure 5, and can be divided into optimization module and compression module at function level. The optimization module and the compression module are independent of each other, and during the optimization of an image, the previous optimized image can be compressed. The optimization module is the most important part in the whole system, which has the ability to

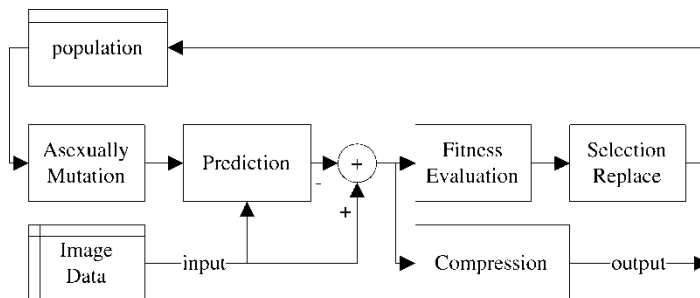


Figure 5. The EHW engine architecture.

gain the optimal predictive parameters through learning the features of different images within the prescribed time. The optimization module consists of the population registers module, the mutation module, the fitness evaluation module and the selection module. Pipeline registers are inserted among them so that all the modules can process the data in parallel to speed up the optimization. In the following we will concentrate on introducing the systolic array for the design of fitness evaluation module, which essentially speeds up the fitness evaluation and satisfies the constraints of system resources and power.

4.1 Fitness evaluation analysis

To describe the systolic array design, we represent the parameter α_i of the k th neighboring pixel in equation (3) by $\alpha_i = a_i(b_i + 1)$. Thus, the formulas related to fitness evaluation are

$$e_{ij} = |x'_{ij} - x_{ij}| = |s_{ij} + t_{ij}|, \tag{4}$$

$$s_{ij} = x_{i-1,j-1} \cdot w_1 + x_{i-1,j} \cdot w_2 + x_{i-1,j+1} \cdot w_3, \tag{5}$$

$$t_{ij} = x_{i,j-1} \cdot w_4 - x_{ij}, \tag{6}$$

$$w_k = e^{-a_k \cdot (b_k + 1)}, \tag{7}$$

$$err = \sum e_{ij}, err' = \frac{\sum e_{ij}^2}{(m-2)(n-1)}, \tag{8}$$

where $i = 2, 3, \dots, n; j = 2, 3, \dots, m - 1; k = 1, 2, 3, 4$. The fitness value is measured by err' . The greater is err' , the smaller is the fitness value. Only when the fitness order is taken into account is err used to replace err' in order to simplify the fitness evaluation.

4.2 Systolic array design at function level

Equation (4) can be regarded as the absolute value of the sum of equations (5) and equation (6) whose inputs are sequences of pixels line by line. The dependent relationship in calculation of the pixel errors is shown in figure 6. As shown in the dependence graph, there are three basic edges corresponding to equation (5), where the data import upward along the direction (0, 1); (1, 0) denotes the right shift weight values and the results move along the direction (1, 1), and there are also three edges corresponding to equation (6), but the directions of the input data, the weight values and the results are (0, -1), (1, 0), and (1, 1). In order to implement the

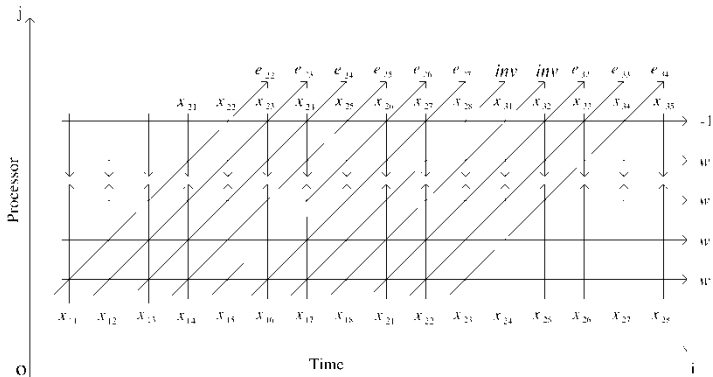


Figure 6. Dependence graph.

application in hardware, this two-dimensional dependence graph should be converted to the time-space form. Only the structure with the broadcast input, the shifted results and the static weights is considered here. The projection vector \mathbf{d} , the processor vector \mathbf{p}^T and the schedule vector \mathbf{s}^T are defined as

$$\mathbf{d} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{p}^T = (0 \quad 1), \quad \mathbf{s}^T = (1 \quad 0). \tag{9}$$

Accordingly, the weights, the input data and the results are corresponding to the ports in the systolic array as shown in table 1.

With the results shown in table 1, the systolic array can be designed as in figure 7, where each systolic cell consists of a multiplier and an adder.

4.3 Systolic array design at bit level

The systolic cell (see figure 8a) consisting of a multiplier and an adder can be realized by various structures. When the scalability is dealt with, the systolic structure at bit level is adopted so that pipeline registers can be easily inserted among the bit operation units. This remarkably improves the system throughput and meets different speed requirements. As an example, for the case that the multiplier input is of three-bits width and the adder input is of six-bits width, the computing process is shown in figure 8b.

The hardware structure is demonstrated in figure 9, where HA stands for the half-adder and FA for the full-adder, and the critical path comprises two half-adders and four full-adders. If the inputs of the multiplier and the adder are of n -bits and $2n$ -bits widths, respectively, the critical path comprises two half-adders and $2n-2$ full-adders.

Table 1. Edge mapping in systolic array.

e^T	$\mathbf{p}^T \cdot \mathbf{e}$	$\mathbf{s}^T \cdot \mathbf{e}$
$w_k (1 \ 0)$	0	1
$i/p (0 \ 1)$ or $i/p (0 \ -1)$	1 or -1	0 or 0
Result (1 0)	1	1

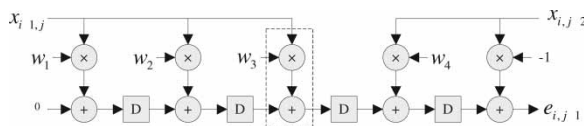


Figure 7. Systolic array at function level.

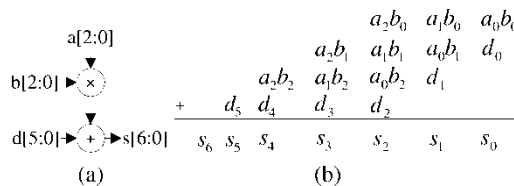


Figure 8. Systolic cell and computing process.

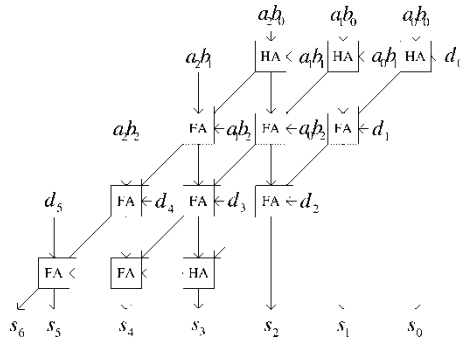


Figure 9. Systolic array at bit level.

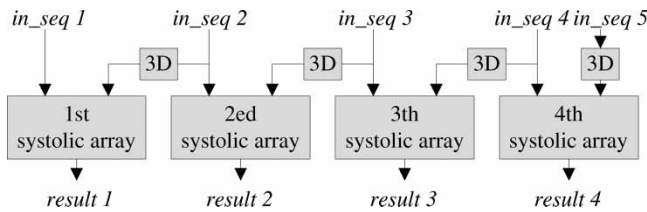


Figure 10. Systolic arrays in parallel.

4.4 Scalability design in parallel

For a high-rate coding, it would be more efficient to employ several systolic arrays. As shown in figure 10, each systolic array has two input ports for the sequence A and B, and one output port for the results. The sequence B of the systolic array is just the sequence A of the next systolic array after three clock cycles delayed, the neighboring two arrays can share a common input sequence. For example, for a structure with four systolic arrays, the pixels have to be accessed once, except those in the $4k + 1$ th ($k = 1, 2, \dots$) lines which have to be accessed twice. In contrast, for the structure with a single systolic array, all the pixels but those in the first and last lines should be accessed twice. Since the memory access is much time-consuming, the data reuse technique is employed to reduce the access times. This not only increases the system speed but also reduces its power consumption.

5. Experiments with software and hardware simulations

The proposed EBinEP is evaluated by comparing it with the method of Huffman coding, as well as the lossless JPEG. Images used here are Lena (as shown in figure 4(e), 256×256 pixels) and a number of scientific images from NASA (available online at: <http://photojournal.jpl.nasa.gov/gallery/universe>). We use these images because they are open, thus the work in this paper can be validated by others. Two examples of scientific images are shown in figure 11. In the experiments, the initial parameters of EBinEP are taken as $l = 10$, $k = 10$, $p_{m1} = 1/3$, $p_{m2} = 1/9$, and the length of chromosome is 24 bits. The size of compressed image is calculated as: $sizeof(\text{compressed error}) + sizeof(\text{compressed border}) + sizeof(\text{binary string})$.

The experiments are divided into three parts according to different points of view. The first is to compare the EBinEP with JPEG2000, the GP-based method, the DE-based method, and

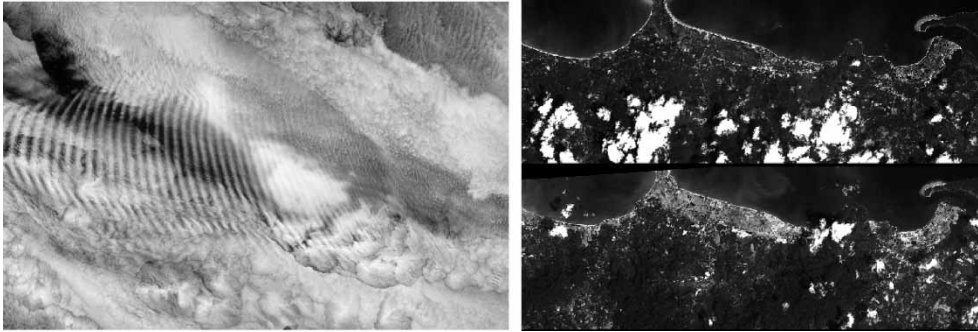


Figure 11. Two examples of science images PIA04349 (left) and PIA07227 (right).

the BinEP on Lena image. This can basically exhibit the efficiency of the proposed method. The second is to evaluate the scalability of EBinEP on a number of large scale images in software simulation, where all results have been averaged over 50 runs of software simulation on AMD 1.2 G CPU. Thirdly, we investigate how the performance of the proposed intrinsic EHW is on FPGA.

5.1 Basic evaluation on small size image with software simulation

In this basic experimental evaluation, we take the conventional method of lossless JPEG2000 and the GP-based method (a typical method with extrinsic EHW) as benchmarks. Our object is to validate the performance of the proposed method. Although the request of EHW on evolutionary algorithms is quite different from that of pure software simulation, we also give the result of DE here for comparison. It should be indicated that although DE is a simple yet effective global optimization algorithm with superior performance in several real-world applications, it is not suitable for evolving on FPGA at least now. Experimental results on Lena are summarized in table 2, where the result for GP comes from (Fukunaga *et al.* 1998, He *et al.* 2005), and the result for JPEG2000 is of lossless compression, and computational times listed in table 2 are averaged results of 50 runs.

It is clear that JPEG2000 belongs to conventional method, therefore it needs relatively less computation time in compressing an image. While evolutionary methods are based on populations and iterative fitness evaluation, they can find relatively better solutions with algorithms going on. Results listed in table 2 and presented in Fukunaga *et al.* (1998), Sakanashi *et al.* (2001) and He *et al.* (2005), reflect this interesting result done by evolutionary techniques. That is, evolutionary algorithms can perform better than conventional methods in compression efficiency; however they consume much more computation time.

Table 2. Comparison between EBinEP and GP-based method on Lena image (256×256 , grey, 66536 bytes), where EBinEP is with 3×3 templet, and BinEP means EBinEP with 1×1 templet (*i.e.*, EBinEP with no technique of gradual approach).

	JPEG2000	GP	DE	BinEP	EBinEP
Compressed size(bytes)	44,090	43,154	40,247	40,247	40,247
Computing time(s)	<1	169.0 ± 0.1	18.23 ± 0.1	11.23 ± 0.1	3.080 ± 0.1

Note: Here the methods for coding error matrix are both the Huffman code technique. For BinEP and EBinEP, the number of generations is set to 300. The switch point of EBinEP from the templet 3×3 to the templet 1×1 is at 200 generations. For DE, the number of generations is 100, and population size is 20, and its scaling factor is 0.5, and crossover ratio is 0.9. The parameters of DE used here is the result of our carefully turning by hand.

With the results listed in table 2, we can clearly see that BinEP outperforms the GP-based method in both compression efficiency and computation time. Comparing with the well-turned DE, BinEP needs relatively less computation time. EBinEP has the same performance of compression efficiency as BinEP, but consumes less computation time. The success of the proposed method is owing to the efficiency of using interpolating function for prediction, the simple expression of chromosome, and the effective search algorithm. This means that BinEP is suitable for the task of adaptive lossless image compression, and the EBinEP is competent for enhancing the performance of BinEP.

5.2 Evaluation on large scale images with software simulation

In the experiment, the initial templet used in EBinEP is 10×10 . Two lossless coding techniques are used to code the evolved error matrix: one is the Huffman code which was used in the past approaches (Fukunaga *et al.* 1998) and (Fukunaga and Stechart 1998), the other is the lossless JPEG2000 which is investigated newly in this paper. The results are listed in table 3, where the results of using Huffman code and the lossless JPEG2000 to directly compress the original image are also summarized. From table 3 we can see, EBinEP with Huffman code and JPEG2000 lossless mechanism outperform both Huffman code and JPEG2000 lossless mechanism, respectively.

Taking the result of BinEP on 256×256 size image Lena as the reference, we summarize the efficiency of EBinEP on larger scale images approximately as shown in figure 12. This relation between the increasing speed of image size and the increasing speed of time consumption shows that the EBinEP can scale up the size of image up to 70 times with a relatively small increase of computation time. From this point of view, the compressive results and the computation time of EBinEP on large scale images are acceptable. The difficulty of EBinEP with very large images, such as PIA07335, is at the later period of evolution when the templet is shrunk to one pixel. At this time, the computation time can not be reduced, because the amount of the evolved data is the total of an image at that time.

5.3 Performance of hardware simulation on-chip

The performance evaluated by the software simulation denotes that the proposed EBinEP algorithm can reach a satisfying result with the ability of scaling up processable data more

Table 3. Comparison between EBinEP, Huffman code, and the lossless JPEG2000 on large scale images.

Data name (images)	Huffman (bytes)	JPEG2000 (bytes)	EBinEPv1 (bytes)	EBinEPv2 (bytes)	Evolving time (s)
PIA07335	5,578,020	2,515,657	2,311,920	2,389,501	266.57 \pm 3
PIA07217	6,286,500	2,031,297	1,918,890	1,983,626	214.89 \pm 3
PIA05578	5,522,256	1,540,111	1,698,480	1,444,500	161.81 \pm 4
PIA07225	4,919,459	2,108,906	2,151,157	2,016,007	121.58 \pm 2
PIA07096	4,247,303	1,938,940	1,637,002	1,886,143	132.92 \pm 3
PIA07343	2,270,780	1,935,780	1,762,368	1,814,022	75.50 \pm 2
PIA07227	1,955,670	1,622,853	1,552,944	1,550,086	52.23 \pm 3
PIA04349	1,115,492	789,619	720,341	735,189	32.32 \pm 3
PIA06322	731,328	529,634	481,381	492,653	37.03 \pm 2
PIA05202	816,127	576,951	534,536	541,220	25.14 \pm 2

Note: EBinEPv1 means that Huffman code is used for coding the error matrix, and EBinEPv2 means the lossless JPEG2000 is used for coding the error matrix. The number of generations is set to 300, and the switch point from the templet 10×10 to the templet 1×1 is at 200 generations. The size of images from up to down are: 3000×2400 , 3000×2400 , 2400×2400 , 2841×1846 , 3000×1688 , 2104×1726 , 1320×1840 , 1374×889 , 1239×805 , and 1065×771 , respectively.

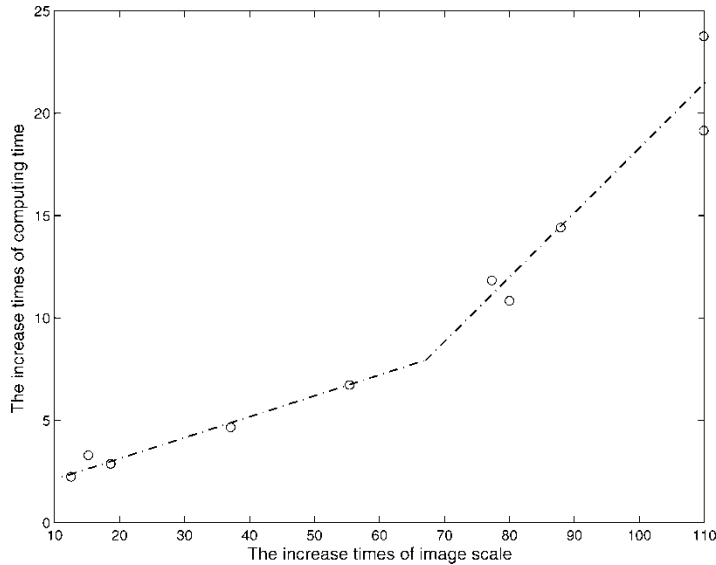


Figure 12. The approximately estimation of the efficiency of EBinEP on larger scale images, where the size of image for contrast is 256×256 . The vertical axis is the increase times of computing time, and the horizontal axis is the increase times of image scale.

Table 4. Performance of hardware simulation.

Data name (Images)	Software simulation (Seconds)	Hardware simulation (s)	Speed up ratio ($t_{\text{soft}}/t_{\text{hard}}$)
PIA07335	148.35	297.54×10^{-3}	499
PIA07217	148.70	297.54×10^{-3}	500
PIA05578	118.55	238.03×10^{-3}	498
PIA07225	107.68	216.73×10^{-3}	497
PIA07096	103.81	209.27×10^{-3}	496
PIA07343	74.02	150.07×10^{-3}	493
PIA07227	49.76	100.37×10^{-3}	496
PIA04349	25.31	50.48×10^{-3}	501
PIA06322	20.60	41.22×10^{-3}	500
PIA05202	17.07	33.93×10^{-3}	503

The software simulation is on AMD 1.2G CPU. The clock frequency of the FPGA chip for hardware simulation is 50 MHz. The number of generations is set to 300, and the switch point from the templet 10×10 to the templet 1×1 is at 200 generations. The size of images from up to down are: 3000×2400 , 3000×2400 , 2400×2400 , 2841×1846 , 3000×1688 , 2104×1726 , 1320×1840 , 1374×889 , 1239×805 , and 1065×771 , respectively.

larger. For the issue of adaptive lossless image compression in EHW, the most important result of a method is how the on-chip performance is in dealing with the real-time tasks. Experimental results of the proposed EHW model are summarized in table 4, where hardware simulations are all on Altera Cyclone EP1C12Q240 FPGA chip.

It is apparent that: (1) the implemented intrinsic EHW can make evolutionary algorithms run much faster. The speeding up ratio is about 500 times advance in total; (2) the proposed EHW model is suitable for evolving large images online (compared with the size of Lena). From results listed in table 4 we can see that the computation time for compressing an image with 1024×786 size or less will be less than 33.93×10^{-3} s. It is an interesting result that the computation time is less than the request time for transfer video images with the speed 25-frame in 1 s (*i.e.*, transfer one frame in 40×10^{-3} s).

It should be indicated that although the hardware implementation can speed up the evolutionary algorithm and fitness evaluation, the idea of how to reformulate the task of adaptive lossless image compression as a task of on-chip evolving topic is crucial. This distinguishes the work in this paper from the previous work of Fukunaga *et al.* (1998) and Sakanashi *et al.* (2004).

6. Conclusions

This paper proposes a novel and practicable intrinsic EHW model, other than the extrinsic EHW model used by Higuchi *et al.* (1997) and Fukunaga *et al.* (1998), for predictive lossless image compression. The proposed method can be implemented and executed on a chip directly, while the GP-based method (Fukunaga *et al.* 1998) and the new result of Sakanashi *et al.* (2004) with dispersed reference compression still need a host machine to run an evolutionary algorithm. It is the first time we have implemented the task of the adaptive lossless image compression using intrinsic EHW. The primary results show that the proposed method has the ability to make a satisfying performance for real-time applications.

The basic idea of the predictive lossless image compression used in this paper is the same as Higuchi *et al.* (1997) and Fukunaga *et al.* (1998): that is, to do lossless image compression through evolution. However, the advantage of our work is that the proposed method is suitable for dealing with real-world tasks on-chip, as we use intrinsic EHW mode. This result owes much to our work on the expression of the object problem, which is the first step of artificial intelligence in solving application problems. The success of EBinEP denotes that it is important to reformulate the object problem to a suitable form of EHW mode in real-world applications.

For the problem of solving the large scale images, the proposed EBinEP can gradually change the environment of individuals from coarse granularity to fine granularity. It is helpful to EHW to deal with large amount of processing data, therefore it can be beneficial for EHW to solve the scalability problem on the issue of lossless image compression. Software simulation shows that the proposed method can largely reduce the computation time, and can scale up the processed image size up to 70 times with relatively slower increase speed in computation time. Hardware simulation on FPGA chip shows that the method can be applied in practice.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China through Grant No. 60573170 and Grant No. 60428202.

References

- A. Fukunaga, K. Hayworth and A. Stoica, "Evolvable hardware for spacecraft autonomy", *Aerosp. Conf. IEEE*, 3, pp. 135–143, 1998.
- A. Fukunaga and A. Stechert, "Evolving nonlinear predictive models for lossless image compression with genetic programming", in *Proceedings of the Third Annual Genetic Programming Conference*, 1998. pp. 95–102.
- J. He, X. Wang, M. Zhang, J. Wang and Q. Fang, "New research on scalability of lossless image compression by GP engine", in *The 2005 NASA/DoD Conference on Evolvable Hardware*, 2005, pp. 160–164.
- H. Hemmi, T. Hikage and K. Shimahara, "AdAM: a hardware evolutionary system", in *IEEE International Conference on Evolutionary Computation*, 1997, pp. 193–196.
- T. Higuchi, M. Murakawa, M. Iwata, I. Kajitani, W. Liu and M. Salami, "Evolvable hardware at function level", in *IEEE International Conference on Evolutionary Computation*, 1997, pp. 187–192.
- J. Kennedy and R.C. Eberhart, "Particle swarm optimization", in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Chichester, UK: IEEE press, Vol. 4, 1995, pp. 1942–1948.

- D. Montana, R. Popp, S. Iyer and G. Vidaver, "EvolvaWare: genetic programming for optimal design of hardware-based algorithms", in *Proceedings of the Third Annual Conference*, 1998, pp. 869–874.
- H. Sakanashi, M. Iwata and T. Higuchi, "A Lossless compression method for halftone images using evolvable hardware", in *Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science 2210*, Berlin: Springer Verlag, 2001, pp. 314–326.
- H. Sakanashi, M. Iwata and T. Higuchi, "Evolvable hardware for lossless compression of very high resolution bi-level images", in *IEE Proceedings – Computers and Digital Techniques*, Vol. 151, 2004, pp. 277–286.
- A. Stoica, R. Zebulum, D. Keymeulen *et al.*, "Reconfigurable VLSI architectures for evolvable hardware: from experimental field programmable transistor arrays to evolution-oriented Chips. Very large scale integration (VLSI) systems", *IEEE Trans.* 9(1), pp. 227–232, February 2001.
- A. Stoica, R. Zebulum and D. Keymeulen, "Progress and challenges in building evolvable devices", in *Evolvable Hardware, 2001. IEEE Proceedings. The Third NASA/DoD Workshop on*, 2001, pp. 33–35
- R. Storn and K. Price, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR-95-012, ICSI (1995).
- A. Thompson and C. Wasshuber, "Evolutionary design of single electron systems", in *Evolvable Hardware, 2000. IEEE Proceedings. The Second NASA/DoD Workshop on*, 2000, pp. 109–116.
- A. Thompson, P. Layzell and R.S. Zebulum, "Explorations in design space: unconventional electronics design through artificial evolution", *IEEE Trans. Evolu. Comput.*, 3, pp. 167–196, September 1999.
- Z. Tu and Y. Lu, "A robust stochastic genetic algorithm (StGA) for global numerical optimization", *IEEE Trans. Evol. Comput.*, 8, pp. 456–470, October 2004.
- X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster", *IEEE Trans. Evol. Comput.*, 2, pp. 82–102, 1999.
- X. Yao and T.Higuchi, "Promises and challenges of evolvable hardware", *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, 29, pp. 87–97, February 1999.