A Symbiotic Evolutionary Design of Error-Correcting Code with Minimal Power Consumption

Heesung Lee and Euntai Kim

In this paper, a new design for an error correcting code (ECC) is proposed. The design is aimed to build an ECC circuitry with minimal power consumption. The genetic algorithm equipped with the symbiotic mechanism is used to design a power-efficient ECC which provides single-error correction and double-error detection (SEC-DED). We formulate the selection of the parity check matrix into a collection of individual and specialized optimization problems and propose a symbiotic evolution method to search for an ECC with minimal power consumption. Finally, we conduct simulations to demonstrate the effectiveness of the proposed method.

Keywords: ECC, single-error correction and double-error detection (SEC-DED), symbiotic evolution, genetic algorithm, power reduction.

Manuscript received Mar. 29, 2008; revised July 1, 2008; accepted Sept. 24, 2008.

This work was supported by the Ministry of Commerce, Industry and Energy of Korea.

Heesung Lee (email: 4u2u@yonsei.ac.kr) and Euntai Kim (phone: +82-2-2123-2863, email: etkim@yonsei.ac.kr) are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul, Rep. of Korea.

I. Introduction

Error correcting codes (ECCs) are used to protect against soft errors and increase the reliability of computer memory [1]. They are widely used in all types of memory, including caches and embedded memory. Single-error correcting and double-error detecting (SEC-DED) codes are generally used for this purpose [2]. This paper focuses on reducing power consumption in memory ECC circuitry that provides SEC-DED.

With the growing importance of power saving in circuit design, many researchers are investigating techniques to reduce power consumption in all components of system design. Power consumption is an important design concern in ECCs since the ECC check circuit is activated during all reading and writing memory accesses, which happen frequently. Therefore, researchers have proposed various methods to reduce power consumption in ECCs [3]-[5]. Some studies have directly applied general power reduction methods to the design of ECCs, while other research has reported power reduction strategies specifically tailored to ECCs. The former are usually not very efficient because they ignore some special properties of ECCs that should be exploited to further reduce power consumption [6]. Among the power reduction strategies tailored to ECC, Ghosh and others introduced an interesting power reduction strategy based on learning theory [6]. They used the genetic algorithm (GA) and simulated annealing (SA) to obtain a power-efficient ECC. Their study, however, does not fully exploit the characteristics of the GA in that the important features of the parent chromosomes are not delivered to the offspring, and the performance is thereby degraded. To solve this, Lee and others proposed a genetic design method for an ECC check circuit with minimal power consumption [7]. Their approach is to select a parity check matrix using a special

genetic operator and to implement the corresponding check circuit in order to minimize its power consumption. However, their study does not observe that the whole parity check matrix can be decomposed into a collection of independent parity columns.

In this paper, we propose a new design method for ECCs with the goal of power minimization. The new method is based on symbiotic evolution, and not on the general (or simple) GA. Symbiotic evolution differs in structure from the general GA. An individual in a simple GA means a complete solution to a given problem, while an individual in a symbiotic evolution is only a part of the solution, and the complete solution is attained when an appropriate set of individuals are gathered. We employ symbiotic evolution in this paper to exploit the property that the whole parity check matrix can be decomposed into a collection of independent parity columns. Thus, we first formulate the selection of the parity check matrix into a collection of independent optimization problems and obtain the complete solution of the decomposed formulation using symbiotic evolution.

The rest of this paper is organized as follows. Section II briefly explains the preliminary fundamentals, including the ECC and symbiotic evolution. Section III proposes a new ECC design scheme based on symbiotic evolution. The aim of the scheme is to design an ECC check circuit with minimal power consumption. Section IV discusses the results of simulations and compares the proposed method with previous methods. Finally, section V offers concluding remarks regarding the study as a whole.

II. Basic Theory

1. Error Correcting Code

ECCs are represented by a parity-check matrix, **H**, and are used on both read and write memory accesses. Once **H** has been selected, the corresponding ECC circuitry can be synthesized in a straightforward way. Consider an (n, k) code, where n is the length of a codeword, k is the number of data bits, and (n, k) is the number of parity check bits. The **H**-matrix is defined as

$$\mathbf{H} = [\mathbf{A}^T \mid \mathbf{I}_{n-k}], \tag{1}$$

where **A** is a $k\times(n–k)$ parity check generator matrix, and $\mathbf{I}_{n-k}$ is an $(n–k)\times(n–k)$ identity matrix. In **H**, there are $(n–k)$ rows and n columns. Each row corresponds to a check bit, and each column corresponds to a bit in the codeword. To possess an SEC-DED property, the **H**-matrix must satisfy the condition that the minimum weight requirement is four, which implies that three or fewer columns of the **H**-matrix are linearly independent [8]. There are many possible choices for the ECC code that satisfy this condition and provide the SEC-DED. We select from among them an ECC with minimal power consumption. The power consumed in the ECC is highly dependent on which columns are used in **H**-matrices and how they are permutated because most of the power is dissipated in the ECC circuitry when the outputs of the gates are switched.

In designing the SEC-DED, there are two configurations: Hamming code [9] and Hsiao code [10]. In Hamming code, we determine only the permutations of the columns of the parity check matrix to minimize power consumption. In Hsiao code, we determine not only the columns of the parity check matrix, but also the odd weight columns to include in the matrix.

2. Symbiotic Evolution

In this subsection, we review some basic concepts of symbiotic evolution. GAs are numerical optimization and search algorithms inspired by the mechanics of natural selection, genetics, and evolution. They typically maintain a population of individuals that represents the set of solution candidates for the optimization problem to be solved [11]. Unlike normal GAs, in which an individual in a population is considered as a potential complete solution to a given problem, symbiotic evolution regards an individual in a population as only a partial solution (a part of a complete solution) to the problem. Complete solutions are formed by concatenating several individuals.

The expression *symbiotic evolution* first appears in [12]. That work proposed the reinforcement learning method called *symbiotic, adaptive neuro-evolution* (SANE), which evolves neurons of the neural network through symbiotic evolution. The fitness of an individual is obtained by summing the fitness values of all possible combinations that the individual joins as a partial solution and dividing the sum by the total number of combinations that individual joins. Because the partial solutions specialize towards one aspect of the problem, partial solutions can be characterized as specializations [13]. A single partial solution cannot take over a population, since there must be other specializations present to obtain high fitness values [13]. If a specialization becomes too prevalent and dominates an entire population, its members will not combine with other specializations; thus, they will not receive the benefit of other specializations and will be assigned low fitness values. For these reasons, the specialization property of symbiotic evolution ensures diversity, and symbiotic evolution can find solutions in diverse and unconverged populations. This is unlike the standard evolutionary approach, which easily and

frequently converges at a local optimum. Symbiotic evolution also appears to be a faster and more efficient search scheme than a traditional GA [14].

III. Design of Error-Correcting Code through Symbiotic Evolution

In this paper, we propose a new design scheme for an ECC based on symbiotic evolution. The specialization property of symbiotic evolution closely matches the individual and independent properties of the columns in the parity check matrix, and symbiotic evolution can be considered a good solution for ECC design. In this paper, we exploit the match to obtain an efficient ECC.

1. Encoding

We first consider symbiotic evolution encoding for Hamming code. For Hamming code, each complete solution corresponds to a particular permutation of the columns of the parity check matrix. Figure 1 shows the structure of the chromosome for Hamming code.

A chromosome represents an association between a memory bit position and the ECC position. The first gene in the chromosome denotes a memory bit position and the position of the column in the **H**-matrix. The second gene denotes an input position for the ECC. The chromosome is only a part of the complete solution (the complete Hamming code) and combines with other chromosomes to form a complete solution as shown in Fig. 2. The chromosomes shown in Fig. 2 are randomly selected and combined so that the input positions of the ECC are all distinct. For example, consider a possible permutation string for k=64: "3, 1, 10, 4, 5,⋯, 15." The string indicates that the first bit of memory is mapped into the third bit in the ECC circuit, and the second bit of memory is mapped to the first input, and so on.

Next, we consider symbiotic encoding for Hsiao code. Since Hsiao code consists of all the low-order, odd-weight columns and some of the higher-order, odd-weight columns, the chromosome of Hsiao code should include the permutation of columns in addition to a selection of higher-order, odd-weight columns. In the case of 64-bit architecture, for example, the

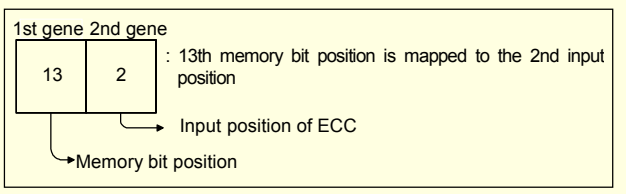


Fig. 1. Chromosome for Hamming code.

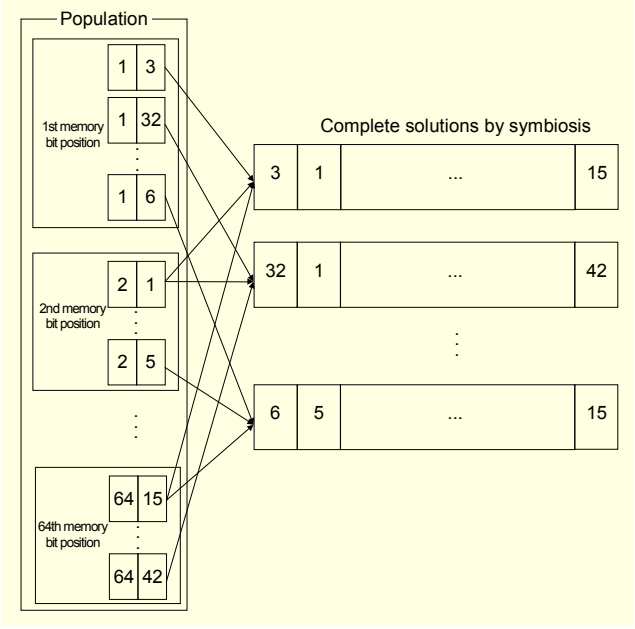


Fig. 2. Complete solution formed by combining several chromosomes.

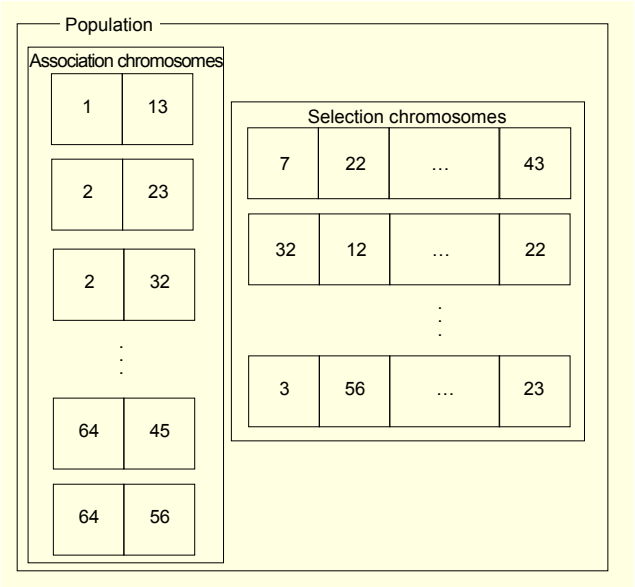


Fig. 3. Chromosome for Hsiao code.

Hsiao code includes all of the low-order, odd-weight columns (8 weight-1 columns and 56 weight-3 columns) and 8 weight-5 columns which are selected from 56 weight-5 columns. Therefore, a population has two kinds of chromosomes: selection chromosomes and association chromosomes as shown in Fig. 3.

The selection chromosomes denote the list of 8 weight-5 columns selected from 56. The selected columns are permutated with low, odd-weight columns to form a complete ECC. The associate chromosome is the same as in Hamming

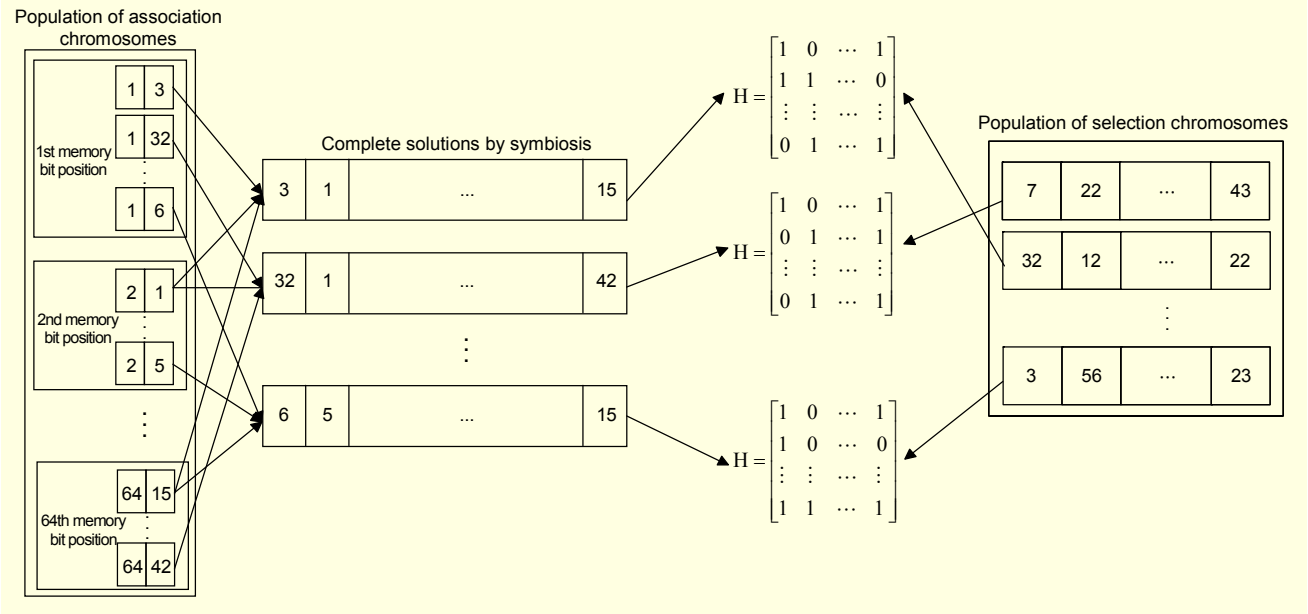


Fig. 4. Complete solution for the Hsiao code.

code. It associates a bit position in memory and an ECC position. Along with the other association chromosomes, it denotes the permutation of the columns of **H**-matrix. Figure 4 shows how the association chromosomes and selection chromosomes combine to form a complete solution. The complete solution includes the list of selected weight-5 columns and the permutation of odd-weight columns.

2. Fitness Function and Genetic Operators

A. Fitness Function

The goal of the proposed ECC design method is to find the ECC that consumes the least power. Instead of directly measuring the power consumed in the ECC, we estimate the power consumption by counting the number of transitions in the ECC check circuit, and use that estimate as a *fitness value of a complete solution*, as in [6], [7]. This approach is reasonable, since most of power dissipates when the outputs of the gates are switched [15].

We now consider how to assign the fitness values to each individual (association chromosome or selection chromosome). In generation, an individual may join more than one complete solution, and different complete solutions will have different fitness values. The fitness of each individual chromosome in symbiotic evolution is calculated by summing the fitness values of all appearances of the individual in that generation (whether it is association or selection) and dividing the sum by the total number of appearances. More specifically, let us consider an association chromosome P which joins n complete solutions:

$$X_1 = (XXPX \cdots XXXX),$$
$$X_2 = (XXXX \cdots XPXX),$$
$$\vdots$$
$$X_n = (XXXP \cdots XXXX).$$

Then, the fitness value of P is computed by

$$f(P) = \frac{1}{n}\sum_{i=1}^{n} T(X_i), \tag{2}$$

where $f(P)$ is the fitness value of P, and $T(X_i)$ is the number of transitions with complete solution, X_i. Similarly, we can compute the fitness value of the selection chromosome. Therefore, we form complete parity check matrices by combining individuals and check the dissipated power and use it as a fitness value for symbiotic evolution. Table 1 gives the detailed steps in assigning a fitness value.

B. Genetic Operators

After each individual has been assigned a fitness value, we apply the genetic operators to create new solutions in the new generation. There are two types of genetic operators: crossover and mutation. The purpose of crossover is to exchange information between different potential solutions.

We use a simple modified arithmetic crossover for Hamming code. The arithmetic crossover produces two complimentary linear combinations of parents. For example, assume that we have two parents, $P_1 = (P_1^1 P_1^2)$ and $P_2 = (P_2^1 P_2^2)$, where P_j^1 for $j = 1, 2$ is a memory bit position and P_j^2 for $j = 1, 2$ is an input position of the ECC. We generate a random number

Table 1. Basic steps in assigning a fitness value to individuals during symbiotic evolution.

Step 1	Set the initial fitness records of individuals to zero.
Step 2	Randomly choose individuals to form a complete solution.
Step 3	Construct an **H**-matrix that corresponds to a complete solution and synthesize an ECC circuit.
Step 4	Evaluate the ECC in terms of power consumption.
Step 5	Accumulate the fitness value of the ECC from its constituent individuals and record the number of appearances of individuals in the complete solutions.
Step 6	Repeat steps 2 to 5 a sufficient number of times.
Step 7	Divide the accumulated fitness value of each individual by its number of appearances.

r from a uniform distribution from 0 to 1 and create two new chromosomes, $O_1 = (O_1^1 O_1^2)$ and $O_2 = (O_2^1 O_2^2)$, according to the following equations:

$$\begin{aligned} O_1^i &= \lfloor rP_1^i + (1-r)P_2^i \rfloor, \\ O_2^i &= \lfloor (1-r)P_1^i + rP_2^i \rfloor, \qquad \text{for } i=1,2, \end{aligned} \tag{3}$$

where $\lfloor \cdot \rfloor$ is the greatest integer function that will round any number down to the nearest integer.

In Hsiao code, we have two chromosomes. For association chromosomes, we use the same crossover as in Hamming code. We now consider the selection chromosomes. Let S_1 and S_2 be two parent selection chromosomes of the Hsiao code, and represent them as

$$\begin{aligned} S_1 &= (S_1^1 S_1^2 \cdots S_1^8), \\ S_2 &= (S_2^1 S_2^2 \cdots S_2^8), \end{aligned} \tag{4}$$

where S_j^i for $i = 1,2,\cdots,8$ and $j = 1,2$ is a weight-5 column selected from all weight-5 columns. The proposed crossover first generates r from a uniform distribution from 0 to 1. Then, two offspring, K_1 and K_2, are computed by

$$\begin{aligned} K_1 &= (K_1^1 K_1^2 \cdots K_1^8), \\ K_2 &= (K_2^1 K_2^2 \cdots K_2^8), \end{aligned} \tag{5}$$

where

$$\begin{aligned} K_1^i &= \lfloor rS_1^i + (1-r)S_2^i \rfloor, \\ K_2^i &= \lfloor (1-r)S_1^i + rS_2^i \rfloor, \qquad \text{for } i=1,\cdots,8. \end{aligned} \tag{6}$$

If $K_1^n = K_1^m$ or $K_2^n = K_2^m$ for $n \neq m$, we regenerate the random number r and create two new chromosomes until the genes of the selection part are all distinct.

Mutation introduces genetic material that may have been missing from the initial population or lost during crossover operations [10]. In other words, mutation is used to alter a single chromosome and produce a new solution. In our study, we use uniform mutation [16]. This operator randomly selects one component, $i = 1,2,\cdots,q$, of the chromosome $P = (P^1 \cdots P^k \cdots P^q)$ and produces $P' = (P^1 \cdots P^{k'} \cdots P^q)$. Here, $P^{k'}$ is a uniform random number $U(1,\beta)$:

$$P^{i'} = \begin{cases} U(1,\beta), & i = k, \\ P^i, & i \neq k, \end{cases} \tag{7}$$

where β is right boundary of the range. For the association chromosome, we use the parameter β=64. For the selection chromosome, we use β=56. As in the crossover operation, we repeat the mutation of the chromosomes until all genes of the selection chromosome are distinct.

IV. Experiments

1. Synthetic Data

To demonstrate the performance of the symbiotic evolutionary ECC design proposed in this paper, we use three sets of 64-bit memory data taken from [7]. Figure 5 shows the characteristics of the three data sets.

In data set 1, one and zero are equally likely for all bits. In data set 3, zero is more likely than one for the bits close to the MSB, and the two numbers are equally likely for the bits close to the LSB. The characteristics of data set 2 lie between the characteristics of data set 1 and data set 3. In this experiment, we use the genetic parameters given in Table 2. We also use elitism, such that each generation automatically recommends its best solution to the next generation.

Since in ECC the major power dissipation comes from the switching of an XOR gate from one stable state to another, the switching activity is the dominant factor in power dissipation. To estimate the power consumption of each parity check matrix, we synthesize the corresponding circuit as a multiple-output logic minimization with 2-input XOR gates, and we check the switching activity. The switching activities in gates are given by the number of transitions in the outputs of the 2-input XOR gates. We make ten independent runs and compute the performances of the proposed algorithm and previous methods [6], [7]. Tables 3 and 4 compare the proposed method to previous methods in terms of power consumption for Hamming and Hsiao codes, respectively.

In Table 3, the random method denotes the case in which Hamming codes are designed randomly. In Table 4, the random method denotes the case in which odd weight columns are selected randomly. They are also permutated randomly, and the simulation is repeated ten times. From the tables, we note

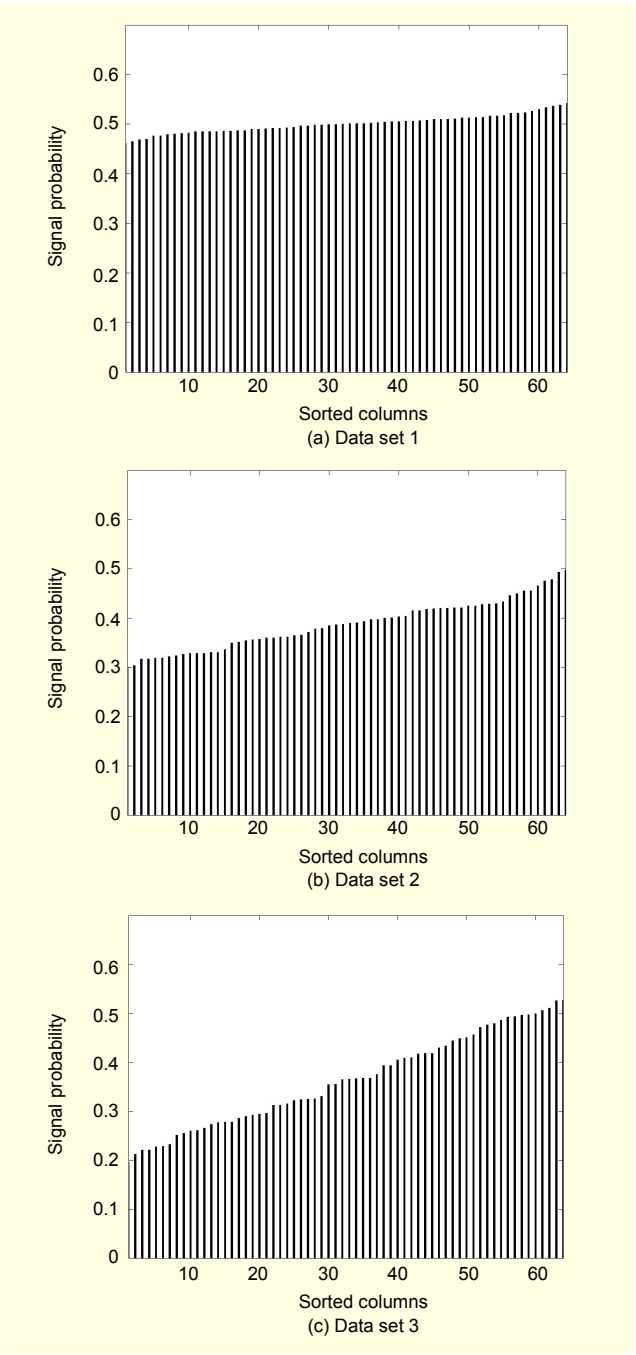


Fig. 5. Bitwise profiles of the memory data.

that the proposed method performs better than existing methods for both the Hamming and Hsiao codes. One explanation for the improved performance of the proposed method is that the specialization property of symbiotic evolution gives a good match to the individual and independent property of the columns of the parity check matrix. Therefore, symbiotic evolution encourages genetic diversity, which prevents the population from converging to a suboptimal solution and finds the best solutions in diverse and unconverged populations. This is in contrast to standard genetic strategies, which often stay at a local optimum. Figure 6 compares the performance of both Hamming and Hsiao codes.

To highlight the comparison between the algorithms, we normalize the power consumption of the four methods with respect to the random method and evaluate their relative performance. Compared to previous methods, the proposed

Table 2. Evolution parameters.

Parameter	Value
Crossover rate	0.6
Mutation rate	0.05
Population size	3,000
Generation	100

Table 3. Performance comparison for Hamming code.

		Random method	Ghosh et al. [6]	Lee et al. [7]	Proposed method
Data 1	Worst	128,382	127,463	127,306	115,254
	Best		127,031	127,010	114,074
	Average		127,223.2 (135.7)	127,167.1 (99.4)	114,740.8 (461.1)
Data 2	Worst	128,026	126,357	126,271	114,589
	Best		125,979	125,917	113,120
	Average		126,163.5 (150.2)	126,089.1 (115.8)	114,016.7 (432.4)
Data 3	Worst	125,557	123,575	123,503	111,664
	Best		123,150	122,917	110,735
	Average		123,411.1 (197.1)	123,243.6 (194.5)	111,303.1 (312.8)

( ): standard deviation

Table 4. Performance comparison for Hsiao code.

		Random method	Ghosh et al. [6]	Lee et al. [7]	Proposed method
Data 1	Worst	108,520	107,513	107,313	101,595
	Best	108,031	107,131	107,089	100,582
	Average	108,240.5 (190.0)	107,315.9 (111.6)	107,192.2 (86.5)	101,219 (301.9)
Data 2	Worst	104,730	103,579	103,283	97,650
	Best	103,896	103,067	102,975	97,017
	Average	104,433.9 (290.5)	103,290.8 (153.2)	103,128.0 (144.9)	97,359.5 (186.2)
Data 3	Worst	101,552	99,780	99,666	94,409
	Best	100,611	99,325	99,276	93,822
	Average	101,099.5 (287.3)	99,643.4 (136.6)	99,495.1 (141.3)	94,018.9 (312.8)

( ): standard deviation

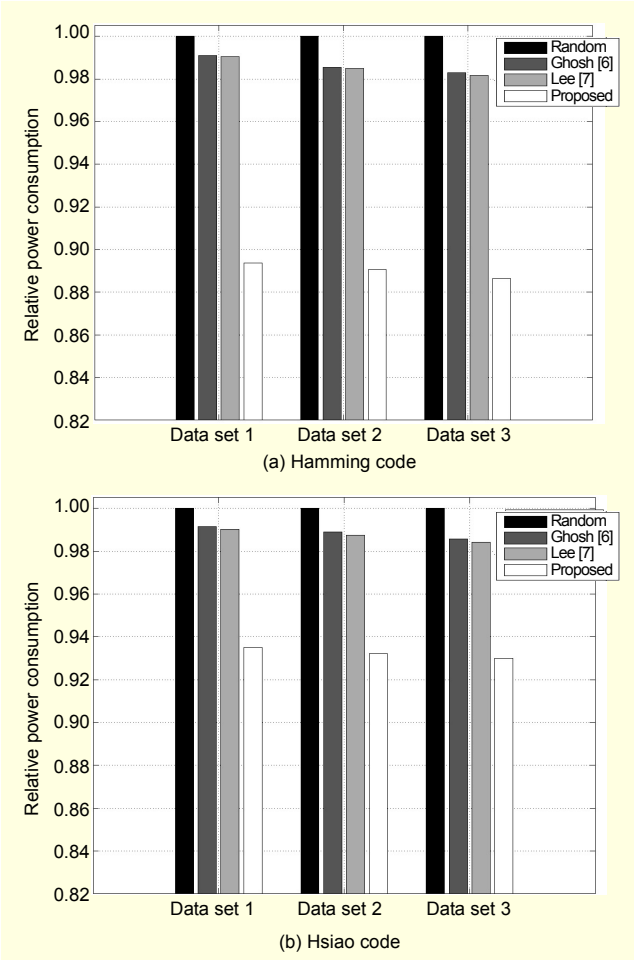


Fig. 6. Performance comparison of the proposed method with previous methods.

method shows excellent results. This is especially true for data set 3, and this property makes the proposed method very promising. In real applications, one and zero are not equally likely, and the actual distribution would be closer to data set 3 than to the other two sets. Hence, we expect the proposed method to perform well for real memory data.

2. Real Data

The proposed method was applied to a multimedia real data set to make a generalized statement concerning the validity of the proposed method. We collected the data from the UMIST face images [17]. Each image has 8-bit grey levels and the size of 92×112 pixels. Figure 7 shows example images from the database.

We downsampled the images to 8×8 pixels and used the rows of the downsampled images as real data. We used the same evolution parameters as in the previous simulation, which are given in Table 2. We made ten independent runs to obtain

Fig. 7. Sample images from the UMIST face database.

Table 5. Power consumption with real data.

		Random method	Ghosh et al. [6]	Lee et al. [7]	Proposed method
Hamming	Worst	1,062,038	966,219	963,180	954,287
	Best		932,461	925,801	925,306
	Average		950,509.2 (9,035.2)	945,848.3 (12,616.4)	942,163.8 (10,097.1)
Hsiao	Worst	880,698	831,257	829,172	822,654
	Best	877,402	821,570	819,727	812,007
	Average	878,908.7 (1,257.9)	826,720.9 (3,524.3)	821,926.6 (2,786.1)	819,845.6 (3,057.9)

( ): standard deviation

reliable results. In Table 5, the proposed method is compared with previous methods [6], [7] in terms of power consumption for Hamming and Hsiao codes. It can be seen that the proposed method shows better performance than the other methods for real data as for synthetic data.

V. Conclusion

In this paper, a new design method for ECCs has been proposed. The method employed the genetic algorithm with a symbiotic mechanism to minimize power consumption of ECCs. We formulated the selection of the optimal parity check matrix into a collection of independent and specialized optimization problems, and we presented the symbiotic evolution for the design of ECCs. The partial solutions in symbiotic evolution were diverse enough to obtain excellent power-saving performance. Our experimental results demonstrated the efficiency of the proposed method.

References

[1] C.L. Chen and M.Y. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review," *IBM J. Res. Develop.*, vol. 28, July 1984, pp. 124-134.

[2] H. Lee, J. Sung, and E. Kim, "Reducing Power in Error Correcting Code Using Genetic Algorithm," *Proc. Int. Conf.*

Computer Information and Systems Science and Engineering, 2007, pp. 179-182.

[3] K. Favalli and C. Metra, "Design of Low-Power CMOS Two-Rail Checkers," *Journal of Microelectronics Systems Integration*, vol. 5, no. 2, 1997, pp. 101-110.

[4] K. Mohanram and N.A. Touba, "Input Ordering in Concurrent Checkers to Reduce Power Consumption," *Proc. of IEEE Symposium on Defecated Fault Tolerance*, 2002, pp. 87-95.

[5] D. Rossi et al., "Power Consumption of Fault Tolerant Codes: The Active Elements," *Proc. of Intentional On-Line Testing Symposium*, 2003, pp. 61-67.

[6] S. Ghosh, S. Basu, and N. Touba, "Reducing Power Consumption in Memory ECC Checkers," *International Test Conference*, 2004, pp. 1322-1331.

[7] H. Lee and E. Kim, "A New Genetic Design for Error Correcting Code for Power Minimization," *Journal of Circuits, Systems, and Computers,* vol. 17, no. 5, Oct. 2008 (to appear).

[8] E. Fujiwara and D. Pradhan, "Error-Control Coding in Computers," *Computer*, vol. 23, 1990, pp. 63-72.

[9] M. Isaka and M. Fossorier, "High-Rate Serially Concatenated Coding with Extended Hamming Codes," *IEEE Communication Letters*, Feb. 2005, pp. 160-162.

[10] M.Y. Hsiao, "A Class of Optimal Minimum Odd-Weight-Column SECDED Codes," *IBM J. Res. Develop.*, vol. 14, July 1970, pp. 395-401.

[11] H. Lee, E. Kim, and M. Park, "A Genetic Feature Weighting Scheme for Pattern Recognition," *Integrated Computer-Aided Engineering*, vol. 14, 2007, pp. 161-171.

[12] D.E. Moriarty and R. Miikkulainen, "Efficient Reinforcement Learning through Symbiotic Evolution," *Mach. Learn.*, vol. 22, 1996, pp. 11-32.

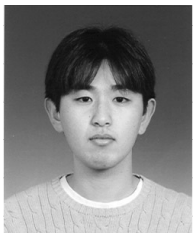
[13] C. Juang, J. Lin, and C. Lin, "Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design," *IEEE Trans. Syst., Man, Cybern. Part B*, vol. 30, no. 2, 2000, pp. 290-302.

[14] H. Juo and H. Chang, "A New Symbiotic Evolution-Based Fuzzy-Neural Approach to Fault Diagnosis of Marine Propulsion Systems," *Artificial Intelligence*, vol. 17, 2004, pp. 919-930.

[15] H. Kim, C. Kim, and S. Kang, "A New Scan Partition Scheme for Low-Power Embedded Systems," *ETRI Journal*, vol. 30, no. 3, 2008, pp. 412-420.

[16] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1999.

[17] D.B. Graham and N.M. Allinson, "Characterizing Virtual Eigensignatures for General Purpose Face Recognition," *Face Recognition: From Theory to Applications*, H. Wechsler et al., eds., 1998, vol. 163, NATO ASI Series F, Computer and Systems Sciences, pp. 446-456.



Heesung Lee received the BS and MS degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2003 and 2005, respectively. He is currently a PhD candidate with the School of Electrical and Electronic Engineering at Yonsei University. His current research interests include computational intelligence, pattern recognition, biometrics, and neural networks.

Euntai Kim received the BS (with top honors), MS, and PhD degrees in electronic engineering from Yonsei University, Seoul, Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2002, he was a full-time lecturer with the Department of Control and Instrumentation Engineering at Hankyong National University, Gyeonggi-do, Korea. Since 2002, he has been with the School of Electrical and Electronic Engineering at Yonsei University, where he is currently an associate professor. He was a visiting scholar with the University of Alberta, Edmonton, Canada, in 2003, and is now a visiting researcher with the Berkeley Initiative in Soft Computing (BISC), UC Berkeley, USA. His current research interests include computational intelligence and machine learning and their application to intelligent service robots, unmanned vehicles, home networks, biometrics, and evolvable hardware.