Vol. 42 No. 4 IPSJ Journal Apr. 2001

Regular Paper

Evolutionary Synthesis of Bit-serial Arithmetic Circuits

Toshiki Terasaki,† Takafumi Aoki† and Tatsuo Higuchi†

The authors have proposed a new graph-based evolutionary optimization technique, called "*Evolutionary Graph Generation* (EGG)", for synthesizing circuit structures. This paper presents an application of EGG to the design of bit-serial data-parallel arithmetic circuits which frequently appear in real-time DSP architectures. The potential of the proposed approach is examined through the synthesis of bit-serial data-parallel adders with multiple operand inputs. A new version of EGG system employs a symbolic verification technique for fast functional evaluation of circuit structures, and can evolve the optimal 8-operand bit-serial adder within a single evolutionary run of 1.5 hours.

1. Introduction

Arithmetic circuits are of major importance in today's computing and signal processing systems. Most of the arithmetic circuits are designed by experienced designers who have specific knowledge of the basic arithmetic algorithms. Even the state-of-the-art logic synthesis tools can provide only limited capability to create structural details of arithmetic circuits.

Addressing this problem, we have proposed an approach to designing arithmetic circuits using a new evolutionary optimization technique called Evolutionary Graph Generation (EGG) [1]. The key idea of the proposed EGG system is to employ general graph structures as individuals and introduce new evolutionary operations to manipulate graph structures directly without encoding them into other indirect representations, such as bit strings (used in GA [2]) and trees (used in GP [3]). The potential of EGG has already been investigated through the design of combinational arithmetic circuits, such as parallel multipliers [1,4]. A natural (but essential) question may arise here: Is it possible to apply the concept of EGG to a wider class of arithmetic circuits whose specifications include not only combinational operations but also sequential operations? This paper is the first attempt to address this question.

In order to apply the EGG system to sequential design specifications of practical size, we must solve a major problem of evolutionary approach related to its computation time. The run time of EGG is mainly dominated by the functional evaluation of evolved structures in every generation. Basically, the complete functional verification of an arithmetic circuit with n input bits for t time steps requires $O(2^{nt})$ logical simulation cycles. This paper presents a new possibility of reducing this computational complexity by introducing a symbolic verification technique. We propose a method of checking the function of the given arithmetic circuit quickly by solving a set of mathematical equations. This approach significantly reduces the time of functional verification for sequential arithmetic circuits. In this paper, we focus on the problem of creating multi-operand bit-serial adders as an example. The new version of EGG system can generate the optimal 8-operand bit-serial adder within a single evolutionary run of 1.5 hours. The proposed approach can also be applied to various sequential design specifications including multiply-adders as demonstrated at the last part of this paper.

The main contributions of this paper are:

(i) to demonstrate that the EGG system can be applied to the synthesis of bit-serial (sequential) arithmetic circuits, and

(ii) to introduce a fast functional verification technique using symbolic computation for bit-serial arithmetic circuits.

2. Basic Concept of EGG and Its Implementation

The Evolutionary Graph Generation (EGG) technique can be regarded as a unique variation of evolutionary computation techniques [5]. In general, evolutionary methods mimic the process of natural evolution, the driving process for emergence of complex structures well-adapted to the given environment. The better an individual performs under the conditions the greater is the chance for the individual to live

† Graduate School of Information Sciences, Tohoku University

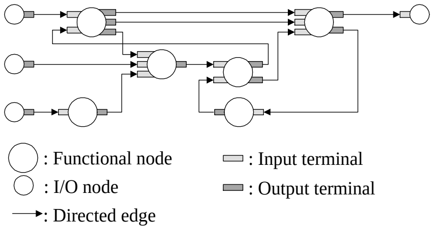


Fig. 1 Example of a circuit graph.

for a longer while and generate offspring. As a result, the individuals are transformed to the suitable forms on the designer's defined constraint. In the EGG system, a graph representing a specific circuit structure is modeled as an individual, and a population of individual graphs is evolved by evolutionary operations. The EGG system is designed to manipulate the graph structures directly without encoding them into other indirect representations, such as bit strings and trees, used in GA and GP.

The EGG system employs *circuit graphs* (**Fig. 1**) to represent circuit structures. A circuit graph G is defined by

$$G = (N(G), D(G)), \quad (1)$$

where $N(G)$ is the set of nodes and $D(G)$ is the set of directed edges. Nodes are of two classes: functional nodes and input/output nodes. Every node has its own name, the function type and input/output terminals. We assume that every directed edge must connect one output terminal (of a node) and one input terminal (of another node), and that each terminal has one edge connection at most. A circuit graph is said to be *complete* if every terminal has an edge connection. In order to guarantee valid circuit structures, all the circuit graphs used in the EGG system are complete circuit graphs.

Figure 2 shows the overall procedure of the EGG system. At first, the system generates embryonic circuit graphs randomly as follows: (i) Select a set of functional nodes S randomly; (ii) Calculate the difference $I - O$, where I denotes the total number of input terminals of the selected nodes, and O denotes the total number of output terminals; (iii) Add some nodes to S so as to satisfy $I - O = 0$; (iv) Connect the input terminals and the output terminals randomly to obtain a complete graph consisting of the nodes S. (This process is also employed for *mutation* operation illustrated later.) After the evolutionary run, every circuit graph in the population is evaluated by symbolic computation technique, which will be described in Section 3. Then, the circuit graphs having higher scores are selected to perform variation operations, and the system generates offsprings for the next generation. The EGG system has two variation operations, *crossover* and *mutation*, to generate offsprings from the parents. The *crossover* operation recombines two parent graphs into two new graphs by exchanging their compatible subgraphs as illustrated in **Fig. 3** (a). The *mutation* operation, on the other hand, partially reconstructs the given circuit graph by replacing its subgraph with a randomly generated subgraph which is compatible with the original subgraph as illustrated in Fig. 3 (b). Both operations transform the structure of circuit graphs, but they preserve the completeness property, that is, if the parents are complete circuit graphs, the generated circuit graphs are also complete. Note that the evolutionary operators shown in Fig. 3 could generate all the possible complete graphs in principle, since the *mutation* operation involves the process of creating arbitrary (complete) circuit graphs randomly. However, system's efficiency in reaching the solution in the search space depends on the actual implementation of the total system flow (Fig. 2). This efficiency must be confirmed through experiments.

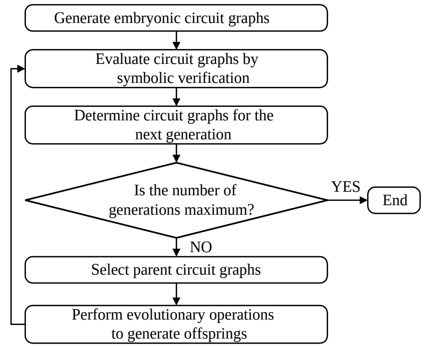


Fig. 2 EGG system flow.

The conventional EGG system, specialized for generating combinational arithmetic circuits, could not be directly applied to other design problems. Addressing this problem, we developed a new version of EGG system on the basis of an object-oriented programming approach. In this system, the *framework classes*, which contain fundamental components for evolutionary graph generation, and the *application class*, which contains application-dependent components, are separated, and hence the sys-

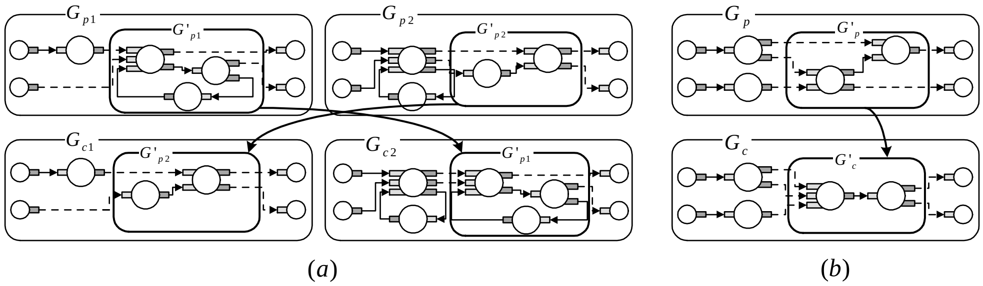


Fig. 3 Examples of evolutionary operations: (a) crossover, (b) mutation.

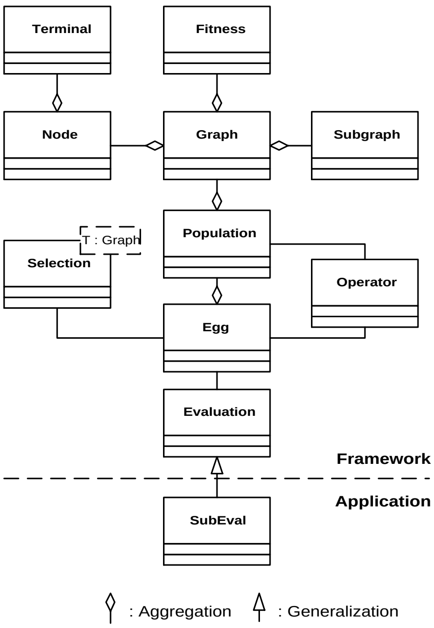


Fig. 4 Class diagram of EGG system.

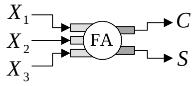
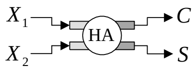
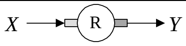
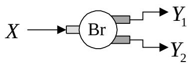
tem can be systematically modified for different design problems. We implemented the EGG system based on the class relationship diagram shown in **Fig. 4**. The EGG system consists of framework (or invariable) classes and an application (or variable) class. The `Egg` class controls the overall work-flow and has an aggregation relationship with the `Population` class, which contains the basic individual model defined by the `Graph` class. The `Graph` also aggregates the `Node`, `Subgraph` and `Fitness` classes, where the `Node` contains the `Terminal` class. The `Operator` class holds miscellaneous operations for handling circuit graphs. The `Evaluation` class, which gives "*fitness*" value to every individual, provides the interface to various applications. By inheriting framework classes, the EGG system can be modified for a wide variety of design problems. The `SubEval` class inherits attributes from the `Evaluation` class, and defines the application-dependent objects. We applied the EGG system to bit-serial arithmetic design by describing the function of inherited `SubEval` class without considering other classes, which is one of the advantages of the object-oriented approach.

3. Synthesis of Multi-Operand Bit-serial Adders

We demonstrate the capability of the new EGG system through an experiment of generating multi-operand bit-serial adders. Note that the proposed method can be applied to other design specifications easily by changing the target function. **Table 1** shows four functional nodes used in this experiment. We have selected a set of functional nodes that makes possible the construction of various bit-serial data-parallel adders, constant-coefficient multipliers, constant-coefficient multiply-adders, which are frequently appeared in signal processing applications. The circuit graphs generated by the EGG system are evaluated by a combination of two different fitness functions, *functionality* and *performance*. The functionality measure F evaluates the validity of logical function compared with the target function. The performance measure P, on the other hand, is assumed to be the product of circuit delay D and the number of inter-module interconnections A.

First, we describe the functionality measure F in detail. In our original work [1)], every circuit graph is translated into the corresponding Verilog-HDL code, which is simulated to evaluate its logical behavior. Basically, the complete functional verification of a sequential arithmetic circuit with n input bits for t time steps requires $O(2^{nt})$ logical simulation cycles. This time is also multiplied by the population size and the number of generations. This is a major draw-

Table 1 Functional nodes used in the experiment.

Name	Symbol / Mathematical representation	Delay
Full adder	 $2C+S=X_1+X_2+X_3$	2τ
Half adder	 $2C+S=X_1+X_2$	τ
1-bit register	 $Y=2X$	
Branch	 $Y_1=X, Y_2=X$	0

back of EGG in its application to practical design problems. The new version of EGG solves this problem by using a symbolic verification technique. We propose a method of checking the function of arithmetic circuits quickly by solving a set of mathematical equations. This method reduces the time for functional verification to $O(m^2)$, where m denotes the number of nodes within the circuit. In practice, the typical time for verifying the function of an evolved circuit is given as 0.0561 seconds ($n=8$, $m=28$), 0.0572 seconds ($n=8$, $m=29$), 0.0602 seconds ($n=8$, $m=30$), 0.0647 seconds ($n=8$, $m=31$), and 0.0685 seconds ($n=8$, $m=32$), while the Verilog-HDL simulation takes 50.2 seconds ($n=8$, $m=32$). Thus, the verification technique itself can be employed for larger circuits. However, the EGG system restricts the number of nodes up to 30 in order to keep the time of total evolution process within a reasonable range.

In the following, we describe the verification technique in detail. This technique can be applied only to arithmetic circuits consisting of components whose functions are represented by addition and multiplication operations. Further investigations will be required to develop a technique applicable to a larger class of arithmetic circuits. We assume the use of LSB-first bit-serial arithmetic based on unsigned binary number system, where the first bit has the weight 2^0, the second has 2^1, the third has 2^2, and so on. Hence, all the bit-serial signals carry non-negative integers. Consider the symbolic verification of a bit-serial arithmetic circuit shown in **Fig. 5**. Using the mathematical

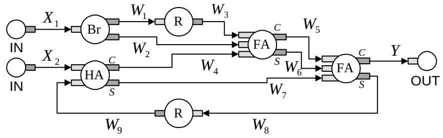


Fig. 5 Example of a 2-input bit-serial arithmetic circuit.

representation of node functions shown in Table 1, we can describe the circuit function as a set of simultaneous equations:

$$\begin{aligned} W_1 &= X_1, \\ W_2 &= X_1, \\ W_3 &= 2W_1, \\ 2W_5 + W_6 &= X_2 + W_3 + W_4, \\ 2W_4 + W_7 &= X_2 + W_9, \\ W_9 &= 2W_8, \\ 2Y + W_8 &= W_5 + W_6 + W_7, \end{aligned}$$

where the variables appeared in the above equations are non-negative integer variables represented by corresponding bit-serial signals shown in Fig. 5. The input/output relationship of the circuit can be derived by solving these equations. Using Gauss elimination, we have

$$2Y = 3X_1 + X_2 - W_4 - W_5 + W_8.$$

As shown in this example, the function of an n-input 1-output bit-serial arithmetic circuit consisting of the nodes shown in Table 1 can be represented in general as

$$\hat{K}_0 Y = \sum_{i=1}^{n} \hat{K}_i X_i + f(X_1, \cdots, X_n), \quad (2)$$

where X_i ($i=1,\cdots,n$) represent bit-serial inputs, Y represents the bit-serial output, $\hat{K}_i$ ($i=0,\cdots,n$) are non-negative integer coefficients, and $f(X_1,\cdots,X_n)$ is a nonlinear function of input operands. The term f involves intermediate variables W_j ($j=1,2,\cdots$) which can not be eliminated through Gauss elimination.

In this paper, we assume that the target function is given by

$$K_0 Y = \sum_{i=1}^{n} K_i X_i, \quad (3)$$

where K_i ($i=0,\cdots,n$) are non-negative integer coefficients. Note here that we must set the target coefficients as $K_0 = K_1 = \cdots = K_n = 1$, when we synthesize an n-operand bit-serial adder. The functionality measure F for the evolved graph is calculated by evaluating the similarity between the coefficients $\hat{K}_i$ (in Eq. (2)) and the target coefficients K_i for $i=0,1,\cdots,n$. To do this, we first expand the coefficients into binary strings as

$$\hat{K}_i = \hat{k}_{i,0}2^0 + \hat{k}_{i,1}2^1 + \cdots + \hat{k}_{i,\|\hat{K}_i\|-1}2^{\|\hat{K}_i\|-1},$$
$$K_i = k_{i,0}2^0 + k_{i,1}2^1 + \cdots + k_{i,\|K_i\|-1}2^{\|K_i\|-1},$$

where $\|K\| = \lceil \log_2(K+1) \rceil$. The similarity between these coefficients is evaluated by computing their cross-correlation. The correlation $M_{\hat{K}_i,K_i}(s)$ of the two coefficient strings with the shift amount s is defined by

$$M_{\hat{K}_i,K_i}(s) = \begin{cases} \dfrac{1}{\|\hat{K}_i\|} \displaystyle\sum_{l=0}^{\|\hat{K}_i\|-1} \delta\left(\hat{k}_{i,l} - k_{i,l-s}\right) & \text{if } \|\hat{K}_i\| \geq \|K_i\|, \\ \dfrac{1}{\|K_i\|} \displaystyle\sum_{l=0}^{\|K_i\|-1} \delta\left(\hat{k}_{i,l-s} - k_{i,l}\right) & \text{if } \|\hat{K}_i\| < \|K_i\|, \end{cases} \quad (4)$$

where $\delta(x)$ is defined as $\delta(x) = 1$ if $x = 0$ and $\delta(x) = 0$ if $x \neq 0$. In the above calculation, we assume the values of the undefined digit position to be 0 for both coefficient strings. Using this correlation function, the similarity F' between Eqs. (2) and (3) is defined as

$$F' = \frac{1}{n+1} \sum_{i=0}^{n} \left[\max_{0 \leq s \leq d} \left\{ 100 M_{\hat{K}_i,K_i}(s) - C_1 s \right\} \right],$$

where $d = \left| \|\hat{K}_i\| - \|K_i\| \right|$ and $C_1 = 10$ in this experiment. The term $C_1 s$ represents the adverse effect due to the shift amount s. Using this similarity, we define the functionality measure F as

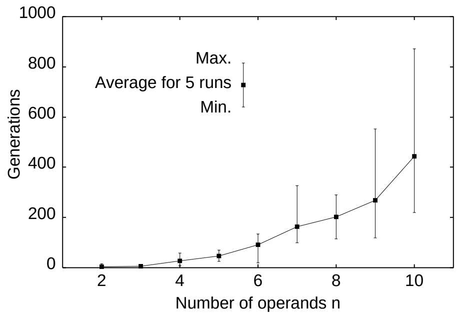
$$F = F' - C_2 p - C_3 q, \quad (5)$$

where p is the number of delay-free loops in the evolved circuit, q is the number of intermediate variables involved in term $f(X_1, \cdots, X_n)$ (that can not be eliminated through symbolic computation), and $C_2 = C_3 = 5$ in this experiment.

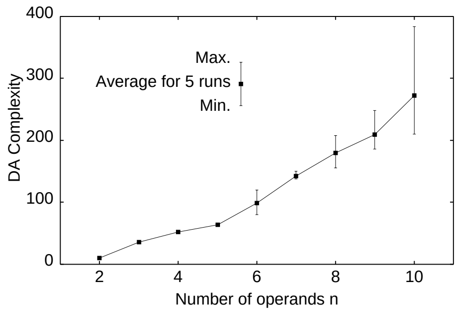
On the other hand, the performance measure P is defined as

$$P = \frac{C_4}{DA}, \quad (6)$$

where A is the total number of inter-module interconnections and D is the maximum register-to-register delay measured by using a 2-input XOR gate as a unit delay element. We use $F+P$ as a total fitness function, where the ratio $P_{\max}/F_{\max}$ is adjusted about 5/100 by tuning the constant C_4.



(a)



(b)

Fig. 6 Result of five evolutionary runs: (a) the number of generations required to obtain the first individual having 100% functionality, (b) the best DA product obtained in the 3000th generation.

4. Experimental Results

The target function considered here is the n-operand bit-serial adder given by Eq. (3) with $K_0 = K_1 = \cdots = K_n = 1$. In this experiment, we assume the condition that the population size is 100, the maximum number of generations is 3000, the maximum number of nodes is 30, the crossover rate is 0.7, and the mutation rate is 0.1. **Figure 6** shows the result of a set of evolutionary runs, in which the EGG system generates n-operand adders for $2 \leq n \leq 10$. We perform five distinct evolutionary runs for every n. The graph (a) plots the average of the number of generations required to obtain the first individual having 100% functionality. The graph (b), on the other hand, is the average of the best DA product obtained in the 3000th generation. The error bars indicate the variation range during five distinct runs. The EGG system can evolve the optimal 8-operand bit-serial adder in 3000 generations, which correspond to the computation of 4.7 hours on Sun Ultra 60 workstation (CPU: 360 MHz, Memory 1.15 GB). **Figure 7** shows the best individuals obtained in five runs for the number of operands

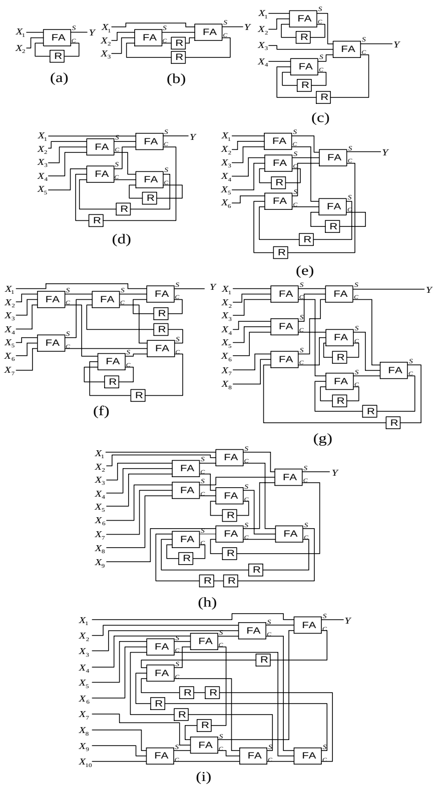


Fig. 7 Best individuals obtained in the 3000th generation, where the number of operands are (a) $n = 2$, (b) $n = 3$, (c) $n = 4$, (d) $n = 5$, (e) $n = 6$, (f) $n = 7$, (g) $n = 8$, (h) $n = 9$, and (i) $n = 10$.

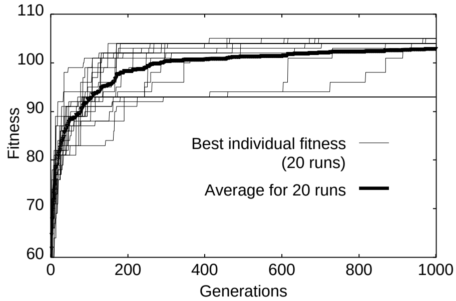


Fig. 8 Transition of the best individual fitness in the population.

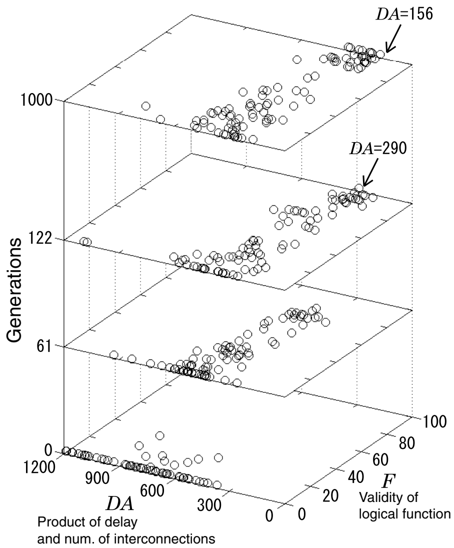


Fig. 9 Example of the evolution process of an 8-operand adder.

ranging from 2 to 10. In every evolutionary run, the individual having 100% functionality was obtained within 3000 generations. We can confirm that all the circuits consist of adder trees with the minimum height (thus the latency is minimized). For the circuits with up to 8 operands ($n \leq 8$), the amount of hardware resources is also minimized. This implies that the EGG system can create near-optimal circuit structures with limited knowledge of arithmetic algorithms.

For more detailed discussion, let us examine the evolution process of an 8-operand bit-serial adder as an example. **Figure 8** shows the transition of the best individual fitness for 20 runs. We can see the staircase improvements of the best individual fitness for every trial. **Figure 9** shows example snapshots of a single evolutionary run. The vertical axis indicates the number of generations, and the horizontal axes indicate the functionality measure F and the DA complexity. Given the initial random population, the evolution is mainly driven towards better functionality. Each individual shows a tendency to keep a specific level of DA product corresponding to the target function. The first individual achieving 100% functionality appears in the 122nd generation. This individual has the DA product of 290. In the 3000th generation, we obtain the best adder configuration shown in Fig. 7 (g), where the DA complexity is reduced to 156. It can be proved that this structure consists of the minimum number of counter stages. Thus, we can confirm the capability of the EGG system to create sequential arithmetic circuits through evolution without using special knowledge of arithmetic algorithms.

Figure 10 shows the comparison of total computation time between the EGG system using symbolic verification and that using Verilog-HDL simulation, where each bar corresponds to the time for a single evolutionary run. We

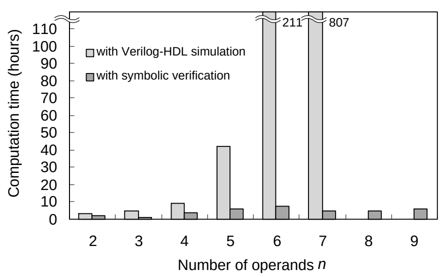


Fig. 10 Computation time of the EGG system.

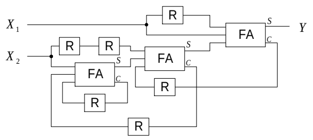


Fig. 11 Evolved multiply-adder structure under the target function: $Y = 3X_1 + 5X_2$.

can observe significant reduction in evolution process by introducing the symbolic computation technique. The speed-up factor reaches 161 times for the case of 7-operand bit-serial adder synthesis.

The proposed approach can be applied not only to the synthesis of multi-operand adders but also to other design problems by changing the target function. For example, if we use the target function (3) with different parameters $n = 2, K_0 = 1, K_1 = 3, K_2 = 5$, we can synthesize the multiply-adder given by $Y = 3X_1 + 5X_2$. **Figure 11** shows the best solution obtained in the 3000th generation. Although further investigations will be required, it may be possible to construct the EGG-based arithmetic synthesis system that can handle general design problems.

Another important issue to be addressed for practical application of EGG system is its computation time. We have recently introduced inexpensive COTS (Commercial Off-The-Shelf) cluster computing technique to reduce the time for experiments of EGG-based circuit synthesis. Each node of the cluster (Linux PC with 700 MHz Pentium III and 1 GB memory) takes only 1.5 hours to generate the optimal 8-operand bit-serial adder in 3000 generations, which is about 3 times faster than the computation done by Ultra 60 workstation (360 MHz UltraSPARC-II with 1.15 GB memory). At present, by clustering 5 PC nodes, we can achieve 5 times increase of evolution throughput ideally, and hence total 15 times speed-up compared with the standard workstation is expected for a large set of evolutionary trials. Thus, this kind of inexpensive COTS parallel processing technique provides a potential possibility of building an EGG-based CAD system that can be applied to various practical circuit design problems.

5. Conclusion and Future Prospects

In this paper, we have presented an application of the EGG system to the design of bit-serial multi-operand adders. A new functional verification technique based on symbolic computation has been proposed to evaluate functions of evolved arithmetic circuits quickly. An experimental design of multi-operand bit-serial adders demonstrates the potential capability of the new EGG system to generate sequential arithmetic circuits without using special knowledge of arithmetic algorithms. The listed below are research subjects to be considered in future:

(i) We need to compare the proposed method of synthesizing bit-serial arithmetic circuits with the conventional rule-based design approaches.

(ii) We must investigate a systematic way of applying the original EGG system to various circuit synthesis problems. In order to extend possible application areas, more generic formal verification techniques for sequential circuits must be introduced.

(iii) In order to achieve further reduction in computation time, we need to explore a technique for utilizing inexpensive COTS parallel processing technology optimized for evolutionary graph generation.

References

1) Aoki, T., Homma, N. and Higuchi, T.: Evolutionary Design of Arithmetic Circuits, *IEICE Trans.Fundamentals*, Vol.E82-A, No.5, pp.798–806 (1999).

2) Miller, F.J., Thomson, P. and Fogarty, T.: Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study, Quagliarella, D., Pèriaux, J., Poloni, C. and Winter, G. (Eds.), *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pp.105–131, John Wiley & Sons (1997).

3) Koza, R.J., III, Bennett, H.F., Andre, D., Keane, A.M. and Dunlap, F.: Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming, *IEEE Trans. Evolu-*

tionary Computation, Vol.1, No.2, pp.109–128 (1997).

4) Homma, N., Aoki, T. and Higuchi, T.: Evolutionary graph generation system with symbolic verification for arithmetic circuit design, *Electronics Letters*, Vol.36, No.11, pp.937–939 (2000).

5) Back, T., Hammel, U. and Schwefel, P.H.: Evolutionary Computation: Comments on the History and Current State, *IEEE Trans. Evolutionary Computation*, Vol.1, No.1, pp.3–13 (1997).

(Received September 22, 2000)
(Accepted February 1, 2001)

Toshiki Terasaki received the B.E. degree in Information Engineering from Tohoku University, Sendai, Japan, in 1999. He is currently working toward the M.E. degree. His research interest includes automatic hardware synthesis using evolutionary computation.

Takafumi Aoki received the B.E., M.E., and D.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1988, 1990, and 1992, respectively. He is currently an Associate Professor of the Graduate School of Information Sciences at Tohoku University. For 1997–1999, he also joined the PRESTO project, Japan Science and Technology Corporation (JST). His research interests include theoretical aspects of computation, VLSI computing structures for signal and image processing, multiple-valued logic, and biomolecular computing. Dr. Aoki received the Outstanding Paper Award at the 1990 and 2000 IEEE International Symposiums on Multiple-Valued Logic, the Outstanding Transactions Paper Award from the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan in 1989 and 1997, the IEE Ambrose Fleming Premium Award in 1994, the IEICE Inose Award in 1997, the IEE Mountbatten Premium Award in 1999, and the Best Paper Award at the 1999 IEEE International Symposium on Intelligent Signal Processing and Communication Systems.

Tatsuo Higuchi received the B.E., M.E., and D.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1962, 1964, and 1969, respectively. He is currently a Professor, and was Dean from 1994 to 1998 in the Graduate School of Information Sciences at Tohoku University. From 1980 to 1993, he was a Professor in the Department of Electronic Engineering at Tohoku University. His general research interests include the design of 1-D and multi-D digital filters, linear time-varying system theory, fractals and chaos in digital signal processing, VLSI computing structures for signal and image processing, multiple-valued ICs, multiwave opto-electronic ICs, and biomolecular computing. Dr. Higuchi received the Outstanding Paper Awards at the 1985, 1986, 1988, 1990, and 2000 IEEE International Symposiums on Multiple-Valued Logic, the Outstanding Transactions Paper Award from the Society of Instrument and Control Engineers (SICE) of Japan in 1984, the Technically Excellent Award from SICE in 1986, and the Outstanding Book Award from SICE in 1996, the Outstanding Transactions Paper Award from the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan in 1990 and 1997, the Inose Award from IEICE in 1997, the Technically Excellent Award from the Robotics Society of Japan in 1990, the IEE Ambrose Fleming Premium Award in 1994, the Outstanding Book Award from the Japanese Society for Engineering Education in 1997, the Award for Persons of scientific and technological merits (Commendation by the minister of state for Science and Technology), the IEE Mountbatten Premium Award in 1999 and the Best Paper Award at the 1999 IEEE International Symposium on Intelligent Signal Processing and Communication Systems. He also received the IEEE Third Millennium Medal in 2000. He received the fellow grade from IEEE, IEICE, and SICE.