Vol.2010-SLDM-145 No.2
2010/5/19

# GA を用いたディジタル回路設計の一手法

王　芳芳[†]　鮑　治国[††]　蘇　怡文[†††]　渡邊孝博[††††]
早稲田大学大学院 情報生産システム研究科

こディジタル回路の設計では回路の良さの尺度として、構成の複雑さに加えて信号遅延や消費電力などの複数の評価基準が必要となっており、このことが回路設計を一層複雑にしている。そこで、遺伝的アルゴリズム（GA：Genetic Algorithm）を用いて複数の評価基準を満たす回路を生成する設計手法が提案されている　（１）、　（２）。本論分では、遺伝子の表現と交差や選択の処理に新たな工夫を導入する、小規模の回路を用いて実験を行った結果、提案手法は従来のものに比べて、より少ないゲート数の回路を生成することができた。

# Optimized Design of Logic Circuit using Genetic Algorithms

Fangfang Wang[†]　Zhiguo Bao[††]　Yi-Wen Su[†††]
Takahiro Watanabe[††††]

In this paper, we propose a new digital circuit design by GA, which has sophisticated chromosome representation, crossover and mutation operators on the performance of GAs. We propose a tree-based chromosome representation, in which initialization depends on a guided random initialization. Based on considering the characteristic of representation, two kinds of crossover operators and three kinds of mutation operators are adopted. Experimental results show that our proposed method provides better results compared to other methods.

## 1. Introduction

Digital circuit design is focused on determining the layout and routing of multiple logic gates in order to achieve some desired logic functions while fulfilling some pre-defined design constraints

However, there are some special problem in digital circuit design, such as, location problem: It is very difficult to decide (1) the number of logic gates used in a circuit, and (2) the network structure of logic gates, allocation problem: It is very difficult to decide which logic gate type should be assigned on each location.

In this study, we are focusing on a genetic algorithm design for digital circuits design.

For solving the digital circuits design problems, we should decide: (1) the network structure of logic gates; and (2) the allocation of logic gate types. For this reason, we have to consider the logic gates' location problem and the logic gate type allocation problem by using genetic algorithms (GAs).

The rest of this paper is organized as follows. Section 2 describes network representation for circuit design. Section 3 introduces the details of our method by GA. Section 4 is experiments and discussion. The last section is conclusions.

## 2. Network Representation for Circuit Design

In this study, we give an idea by using network structure for presenting circuit layout. For example, the truth table and a correspondingly configured matrix for a full adder are shown in Table 1 and Fig. 1. In the table, $I_1$, $I_2$ and $I_3$ are the inputs and the carry-in to the adder and the two outputs are indicated by $\hat{O}_1$ (sum) and $\hat{O}_2$ (carry-out).



ⓒ2010 Information Processing Society of Japan

Vol.2010-SLDM-145 No.2
2010/5/19

Table 1 Truth table for a full adder circuit

| $I_1$ | $I_2$ | $I_3$ | $O_1$ | $O_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

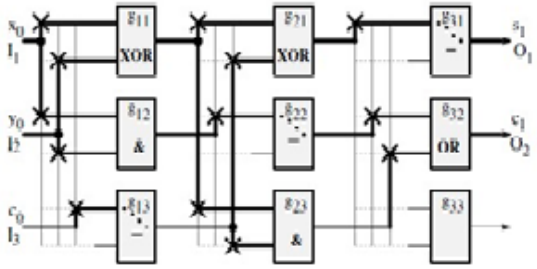



Fig. 1 Matrix for a full adder circuit

We can give a network representation as shown in Fig. 2 for circuit layout in Fig. 1.

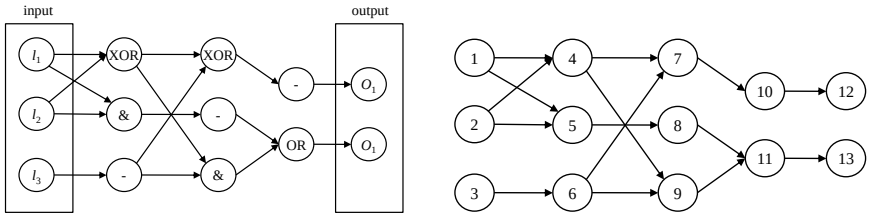



Fig. 2 Network representation for circuit layout

Based on the network representation, the circuit layout can be easily presented by following Edge List, Adjacency List or Adjacency Matrix.

## 3. GA Approach for Circuit Design

### 3.1 Genetic Representation

How to encode a solution of network design problem into a chromosome is a key issue of GA. When a new encoding method is given, it is necessary to examine whether we can build an effective genetic search by this encoding. Several principles have been proposed to evaluate an encoding: Space, Time, Feasibility, Uniqueness, Heritability, Locality.

In GAs literature, whereas the several kinds of encoding methods were used to obtain circuit layout problems, most of them cannot effectively encode or decode between chromosomes and corresponding network structure.

#### 3.1.1 Adjacency Matrix Representation

Recently, the most of researches use the matrix representation as the chromosome design for solving the circuit design problems. Coello *et al* (2). used a matrix to represent a circuit as shown in Fig 3. Each matrix element is a gate. There are five types of gates: AND, NOT, OR, XOR, and WIRE that receives its two inputs from any gate at the previous column as shown in Fig 3.

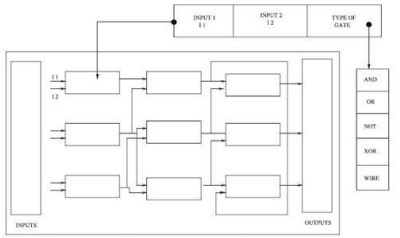



Fig. 3 Matrix used to represent a circuit

The adjacency matrix representation is the intuitive representation for the network optimization problems (such as circuit layout design). However, as they need a lot of space in memory, generally, we use not so much arcs for presenting the result, so there are many “0” in the matrix representation. After genetic operations (crossover and mutation), the original chromosome is difficult to be changed, because of the most insignificant information (value 0) in the matrix shown in Fig. 4. Other case, after genetic operations, the offspring should lose some edges to ensure the network structure continuity. The disadvantages are losing uniqueness and heritability.

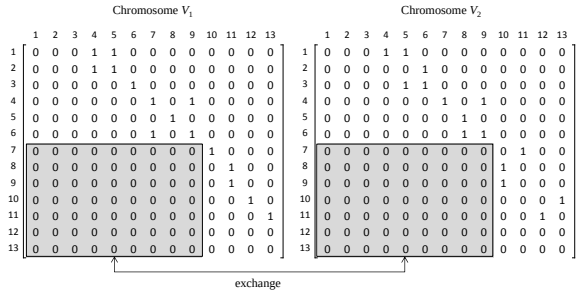



Fig. 4 Crossover example 1 for adjacency matrix representation



ⓒ2010 Information Processing Society of Japan

**3.1.2** Edge List Representation

Edge list representation is an intuitive representation of a network structure. A general edge list representation requires space proportional to n-1 and the time complexities is O(m), where n is the number of nodes and m is the number of edges. The mapping from chromosomes to solutions (decoding) may be 1-to-1 mapping. In a complete graph, m=n(n-1)/2 and the size of the search space is 2n(n-1)/2. As the disadvantages of edge list representation, there is a large difference between different network structures presented by edge list representation. If we use the traditional genetic operators (crossover and mutation), the offspring loses some edges to ensure the network structure continuity (Same with the adjacency matrix representation) as shown follows.

Chromosome V1 ={(1, 4), (1, 5), (2, 4), (2, 5), (3, 6), (4, 7), (4, 9), (5, 8), (6, 7), (6, 9), (7, 10), (8, 11), (9, 11), (10, 12), (11, 13)}.

Chromosome V2 ={(1, 4), (1, 5), (2, 6), (3, 5), (4, 7), (4, 8), (5, 8), (5, 9), (6, 8), (6, 9), (7, 11), (8, 10), (9, 10), (10, 13), (11, 12)}.

Chromosome V1' ={(1, 4), (1, 5), (2, 4), (2, 5), (4, 7), (4, 8), (5, 8), (5, 9), (6, 8), (6, 9), (7, 11), (8, 10), (9, 10), (10, 13), (11, 12)}.

After one-cut crossover, the chromosome V1' loses the edges from node 3. The disadvantage is: lost the heritability.

**3.1.3** Adjacency Lists

Generally, the adjacency lists are not useful for the chromosome design.

**3.1.4** Proposed Tree Representation

There are three difficult issues to be solved in the GA design:

Decision 1: the number of logic gates used for presenting a full adder circuit

Decision 2: the network structure of logic gates

Decision 3: which logic gate type should be assigned on each location

We propose a type of tree representations for the chromosome design to solve these difficult issues. The assumptions are given as follows

Assumption 1: We have the fixed logic gates for the experiments.

Assumption 2: For each logic gate, we have two inputs and one output. For uniform the gene structure in the chromosome, we define the NOT logic has two inputs (A, X) and one output $\overline{A}$, where X means any kind of inputs.

Assumption 3: For each inputs or output is 1 or 0.

Assumption 4: The cases in the truth table is 2n, where n is the number of inputs of circuit.

Assumption 5: In this study, we only consider the n-inputs 1-output problems, such as the first example showed in Table 2. The multi-output can be formulated as the special case with combining multiple one-output problem.

Table 2 Truth table for the circuit of the first example

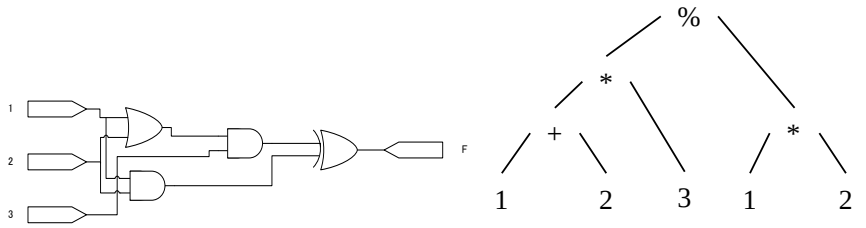
| 1 | 2 | 3 | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**3.1.5** Tree Representation

A logic gates layout can be represented as a slicing structure.

Slicing structures comprising given n inputs can be represented by slicing tree or polish expressions over the alphabetical set Z = {1, 2, …, n, +, *,…}.

An example of a slicing structure is shown as follows:




(a) A solution of logic gates layout for first example (b) Slicing tree

Fig. 6 Representation of slicing structure

In this tree representation, the advantages are (1) Saving the amounts of memory, and the smaller alternative solutions space. The matrix representation needs the n x n space, and edge list representation needs the m space, but the tree representation needs only n space, where n mean the number of nodes, m means the number of edges in the network. (2) By using the genetic operations (crossover or mutation), we can change the logic gates layout and right exchange the logic gate types in the chromosome. The tree representation has very properly good heritability. However, this tree representation has a disadvantage caused by: n-to-1 mapping. It is mean that we obtain the same result by different chromosomes. For example, Chromosomes v1=[1 2 + 3 * 1 2 * %] and v2=[1 2 * 1 2 + 3 * %] produce the same result shown in Fig. 6 (a). So we have to consider this disadvantage when we design the genetic operations for process convergence. We summarize the advantages (and disadvantages) of our


ⓒ2010 Information Processing Society of Japan

proposed tree representation and other genetic representations in Table 3.

Table 3 Summarizes the performance of Genetic Representations

| Representation | Space | Feasibility | Uniqueness | Heritability |
|---|---|---|---|---|
| Adjacency Matrix Representation | $n \times n$ | Worst | 1-to-1 mapping | Worst |
| Edge List Representation | $m$ | Poor | 1-to-1 mapping | Poor |
| Tree Representation | $n$ | Good | $n$-to-1 mapping | Good |

### 3.2 Initialization

As discussed above, one of difficulty is that different digital circuits require different numbers of logic gates. So the length of tree representation-based chromosome should be different. In this study, for the initialization of chromosomes, we try to use the most logic gates into the chromosome as possible as we can.

**procedure**: Guided Random Initialization
**input**: a set of inputs $Z$={1, 2, …, $n$}, number of logic gates $m$, a set of logic gate types $T$={*, +, -, ^, &, %, #}
**output**: a chromosome
**begin**
  **initialize** a set of logic gates and inputs $L \leftarrow Z$;
  **initialize** a set of inputs $Z' \leftarrow Z$;
  **initialize** the number of logic gates in set $L$: $m' \leftarrow 0$;
  **random select** 2 inputs from $L$: $(i, j) \leftarrow$ random$\{(a, b)|L\}$;
  **delete** the inputs form $Z' \leftarrow Z' - \{i, j\}$
  **random select** a logic gate type: $t \leftarrow$ random$\{ l|T\}$;
  **generate** a logic gate $g=\{(i, j), t\}$
  **update** set of logic gates $L \leftarrow L+\{g\}$;
  **update** the number of logic gates $m' \leftarrow m'+1$;
  **while** (m'=m & $Z'=\emptyset$) **do**
    **random select** 2 inputs from $L$: $(i, j) \leftarrow$ random$\{(a, b)|L\}$;
    **delete** the inputs form $Z' \leftarrow Z' - \{i, j\}$
    **random select** a logic gate type: $t \leftarrow$ random$\{ l|T\}$;
    **generate** a logic gate $g=\{(i, j), t\}$
    **update** set of logic gates $L \leftarrow L+\{g\}$;
    **update** the number of logic gates $m' \leftarrow m'+1$;
  **end**
  **generate** the chromosome by $L$;
**end**

### 3.3 Fitness Evaluation and strategy

Generally, the fitness is using the objective function of the problem. However, the objective of digital circuits design problem is the boolean value 0 or 1. In most cases, the objective value of the chromosome is 0. So it is very difficult to evaluate the chromosomes if all of the fitness values are 0. In fact, the digital circuits design problem is very similar to some constraint problems when we use GA to solve it. Several techniques have been proposed to handle constraints with GAs based on some strategies such as rejecting strategy, repairing strategy, modifying genetic operator strategy and penalty strategy.

Most of these strategies have the advantage that they never generate infeasible solutions, but have the disadvantage that they consider no points outside the feasible regions. Therefore, in this study, we adopt the penalty strategy for fitness evaluation. Penalty techniques transform a constrained problem into an unconstrained problem by penalizing infeasible solutions, in which a penalty term is added to the objective function for any violation of the constraints. The key idea of the penalty technique is based on the conventional optimization as follows.

How to determine the penalty term so as to strike a proper balance between the information preservation and the selection pressure for the infeasible solutions and to avoid both under-penalty and over-penalty

Keep some infeasible solution in population so as to force genetic search toward to optimal solution from both side of feasible and infeasible region.

We take the addition form expressed as follows:

$$\text{eval}(x) = f(x) + p(x)$$

where x represents a chromosome, f(x) is the objective function of problem (0 or 1), and p(x) the penalty term.

$$p(x) = \frac{1}{\alpha m + \beta b}$$

where $\alpha$ and $\beta$ are parameters for adjusting the penalty values, m is the number of logic gates used in the chromosome, and b is the total number of rows which violate the logical operation in the truth table.

### 3.4 Genetic Operators

The genetic operators mimic the process of heredity of genes to create new offspring at each generation. The operators are used to alter the genetic composition of individuals during representation. In the following well-known genetic operators, we combine two kinds of crossover and three kinds of mutations. The first motive is to alter the circuit structure, and the second motive is to alter the logic gate types.


ⓒ2010 Information Processing Society of Japan

**3.4.1** Crossovers

One-cut Crossover: the cut point has to be selected at the position of the end of logic gate symbol (Fig. 7).

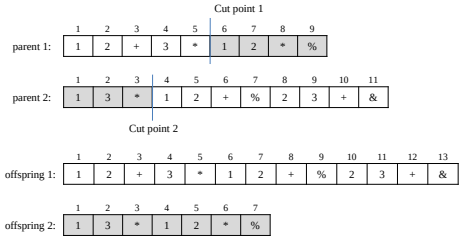



Fig. 7 One-cut crossover

Position-based Crossover (PX): It is essentially a kind of uniform crossover for permutation representation together with a repairing procedure. It also can be viewed as a kind of variation of Order Crossover (OX) in which the nodes are selected inconsecutively (Fig. 8).

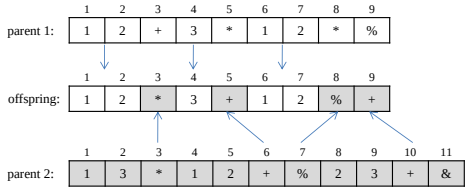



Fig. 8 PX

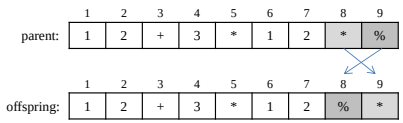



Fig. 9 Swapping mutation

**3.4.2** Mutations

There are typical three mutations: Swapping Mutation (Fig. 9), Inverting Mutation (Fig. 10) and Altering Mutation (Fig. 11)

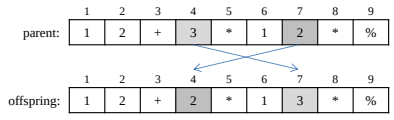



Fig. 10 An example of inverting mutation

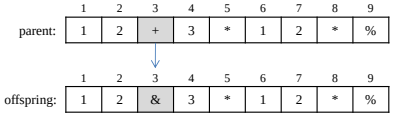



Fig. 11 An example of altering mutation

**3.4.3** Selection

The selection operator is intended to improve the average quality of the population by giving the high-quality chromosomes, i.e., a better chance to get copied into the next generation. The selection thereby focuses the exploration on promising regions in the solution space. In this paper, the roulette wheel selection, i.e., a type of fitness-proportional selection is adopted.

## 4. Experiments

In this section, our proposed Tree-based GA is compared with n-cardinality GA (NGA) by Coello et al., GA with Cell Crossover (ccGA) by Bao and Watanabe (2009), and two kinds of human designers by (1) Karnaugh Maps plus Boolean algebra identities to simplify the circuit, and (2) using the Quine–McCluskey Procedure. There are two test problems used. All the simulations were performed with Java on Intel(R) Core(TM)2 CPU 6700 2.66GHz 2.00 RAM.

### 4.1 Experiment 1

The first example is a three-even parity problem, whose truth table with three inputs and one output is shown in Table 4. Results are listed in Table 5.

Table 4 Truth table for the circuit of the first example

| 1 | 2 | 3 | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Table 5 Truth table for the circuit of the second example

| Algorithm | Solution | # of logic gates |
|---|---|---|
| HD 1 | F=3*(1%2)+2*(1%3) | 5 gates |
| HD 2 | F=1-*2*3+1*(2%3) | 6 gates |
| ccGA | F=3*(1+2)%(1*2) | 4 gates |
| Proposed Tree-based GA | F=(1+2)*3%(1*2) | 4 gates |

HD 1: human designers by Karnaugh maps
HD 2: human designers by Quine–McCluskey

### 4.2 Experiment 2

The second example has four inputs and one output, as shown in Table 6. Results are compared in Table 7.



ⓒ2010 Information Processing Society of Japan

Table 6 Comparison of the best solutions for the circuit of the first example

| 1 | 2 | 3 | 4 | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Table 7 Comparison of the best solutions for the circuit of the second example

| Algorithm | Solution | # of logic gates |
|---|---|---|
| HD 1 | F=((1-*3)%(4-*2-))+((3-*4)*(1%W-)) | 11 gates |
| Sasao | F=3-%4-*2-%3*4-*1-%3-*4-*2 | 12 gates |
| ccGA | F=(2*4*3-%((2+4)%1%(3+4+1)))- | 10 gates |
| Proposed Tree-based GA | F=((2+3*4)%((3+4)*(3%1)))- | 7 gates |

Comparisons of the approach to others that n-cardinality GA (NGA), GA with Cell Crossover (ccGA), and 2 kinds of human designers by Karnaugh maps plus boolean algebra identities to simplify the circuit, and the Quine–McCluskey procedure, provides better results on 2 test experiments.

## 5. Conclusions

In this study, we investigated a chromosome representation, crossover and mutation operators on the performance of GAs for digital circuits design. Based on the performance analysis of these chromosome representation methods in GAs, we proposed a tree-based representation, in which initialization depends on an guided random initialization. Considering the characteristics of the tree-based chromosome representation, we adapted two kinds of crossover operators and three kinds of mutation operators. Comparing our approach with others which are n-cardinality GA (NGA), GA with Cell Crossover (ccGA), and two kinds of human designers by Karnaugh maps plus boolean algebra identities, and the Quine–McCluskey procedure, the proposed one showed better results on two test experiments. As a result our porposed approach can generate better circuit than that by other approaches.This study was realized to investigate the effects of the different chromosome representation method; the interaction of the representation with the crossover operators and mutation operators affect its performance.

In the future, we will investigate the proposed tree-based GA to solve the multi-output digital circuits design, and evaluate the performance on the large-scale, real-life instances; and also improve the tree-based GA for the multi-objective digital circuits design.

## 6. Acknowledge

This study was partially supported by knowledge Cluster Project of MEXT and Grant of Waseda Univ. Special Research Project.

## 参考文献

1) Coello Coello, C.A. A comprehensive survey of evolutionary based multi-objective optimization techniques. Knowledge and Information Systems. An International Journal 1(3), 269–308, 1999.
2) Coello Coello, C.A., Christiansen, A.D., & Hernández Aguirre, A. Use of evolutionary techniques to automate the design of combinational circuits. International Journal of Smart Engineering System Design 2(4), 299–314, 2000.
3) Zhiguo Bao and Takahiro Watanabe. A Novel Genetic Algorithm with Cell Crossover for Circuit Design Optimization.IEICE, pp. 2982-2985, 2009.



ⓒ2010 Information Processing Society of Japan