A Novel Genetic Algorithm with Different Structure Selection for Circuit Design Optimization

Zhiguo Bao and Takahiro Watanabe
Graduate School of Information Production and Systems
Waseda University
Kitakyushu-shi, Japan
Email: baozhiguo@moegi.waseda.jp

Abstract

Evolvable Hardware (EHW) is a new research field about the use of Genetic Algorithm (GA) to synthesize an optimal circuit. In traditional GA, the tournament selection for crossover and mutation is based on fitness of individuals. It can make convergence easily, but maybe lose some useful genes. In selection, besides fitness, we consider the different structure from individuals comparing to elite one. First, select some individuals with more different structures, then cross over and mutate these ones to generate new individuals. By this way, GA can increase diversification to searching spaces, so that it can find better solution. We propose optimal circuit design by using GA with different structure selection (GAdss) and with fitness function composed of circuit complexity, power and signal delay. Its effectiveness is shown by simulations. From the results, we can see that the best elite fitness, the average value of fitness of correct circuits and the number of correct circuits of GAdss are better than GA. The best case of optimal circuits generated by GAdss is 8.1% better in evaluating value than the circuit of GA.

Keywords

GA, EA, Circuit Optimization, EHW

1 Introduction

In artificial intelligence, Evolutionary Algorithm (EA) is a generic population-based heuristic optimization algorithm. It uses some mechanisms inspired by biological evolution: selection, crossover, mutation and reproduction. Genetic Algorithm (GA) [1][2] is one of the typical evolutionary algorithms.

In traditional GA, the tournament selection for crossover and mutation is based on fitness of individuals. It can make convergence easily, but maybe lose some useful genes. In tournament selection, besides fitness, we consider the different structure from individuals comparing to elite one, because the offspring with more different structure can enhance diversification to searching spaces. The value of different structure is defined by the sum of absolute value of difference of genes between two individuals. First, select some individuals with more different structures, then cross over and mutate these ones to generate new individuals. By this way, GA can increase diversification to searching spaces, so that it can find better solution.

While GA technologies are developed, Evolvable Hardware Systems (EHW) [3][4] is researched, inspired by the Evolution theory. EHW was first proposed in the early 1990s' for hardware design. It is classified into two categories: original design and adaptive systems. Original design uses EA to design a system that meets a predefined specification, and adaptive systems reconfigure an existing design to adapt to a variable operational environment. EHW can be used as an alternative to conventional hardware design methodology. The application of the EHW technique appears to be successful and promising, because it could automatically generate digital circuits by using EA. However, there still remain critical issues such as scalability, maintainability and generalization [5][6] to apply EHW for practical design problems. One of them is a circuit design optimization problem, where mixed design constraints are subjected.

In this paper, we propose a new approach for circuit design optimization by GA with Different Structure Selection (GAdss), where mixed constraints on circuit complexity, power and signal delay are considered. First, we introduce the evaluating value about correctness, complexity, power and signal delay to the fitness function in order to get an optimal circuit. The fitness function used in the experiments aims at accepting solutions with 100% correctness of the target circuit, and with maximal evaluating values about complexity, power and signal delay. Then GAdss can autonomously synthesize a circuit that is equivalent to a conventional design in functionality, but is simpler and has better performance. As a result, GAdss can find a better circuit, compared to GA. To verify an effectiveness of our approach, a simple 2-bit half adder circuit is experi-

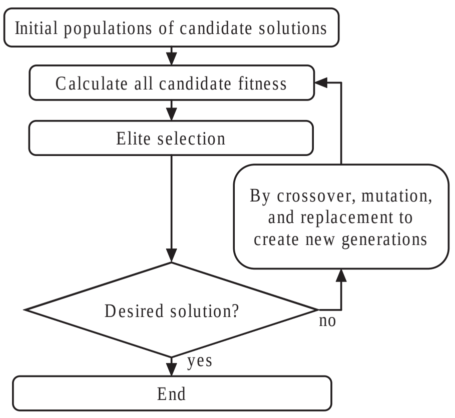
Initial populations of candidate solutions
Calculate all candidate fitness
Elite selection
By crossover, mutation, and replacement to create new generations
Desired solution?
no
yes
End

Figure 1: The evolutionary process of GA.

mentally synthesized.

In the next section, a brief overview of GAdss is described. Section 3 describes the use of GAdss as a new approach for the automatic design of an optimized circuit. Section 4 shows experiments on a 2-bit half adder circuit design as an example. Finally, the paper concludes with a summary of the results in section 5.

2 Genetic Algorithm with different structure selection

2.1 Genetic Algorithm

GA is a search technique used to find exact or approximate solutions to optimization and search problems. Figure 1 shows a graphical representation of the GA mechanisms. GA involves a search from a population of individuals.

In the initialization of a GA population, each individual is randomly generated. In the evaluation, GA evaluates each candidate according to a fitness function, which indicates how well a candidate satisfies the design specification. In each generation, the elite individuals are preserved and the rest of the individuals are replaced by the new ones generated by crossover and mutation. GA continues to evolve until it finds the best individual.

2.2 Genetic Algorithm with different structure selection

In traditional GA, elite selection and tournament selection are based on the fitness of individuals. This is good for GA to find the local best solution, but it maybe premature convergence. To avoid premature convergence, we consider the different structure of individuals compared to elite one. The value of different structure is defined by the sum of absolute value of difference of genes between two individuals. We select some individuals with different structure to do crossover and mutation, to generate some new individuals to the next populations. This method can extend diversification to search spaces, so can be fond a better solution.

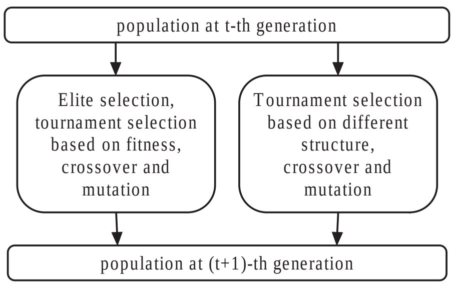
population at t-th generation
Elite selection, tournament selection based on fitness, crossover and mutation
Tournament selection based on different structure, crossover and mutation
population at (t+1)-th generation

Figure 2: The reproduction of GAdss.

Figure 2 shows a graphical representation of the reproduction of GAdss. In each generation, the elite individuals are preserved. In selecting some individuals, half of them are processed by tournament selection based on fitness, and another half are processed by tournament selection based on different structure. Then crossover and mutation are performed to create new ones for next generation.

$< Crossover >$

The crossover is operated between two parents, and two new individuals are generated. The procedure of the crossover is as follows.

(1) Select two individuals as parents using tournament selection. (2) Some bits in the parents are selected as the crossover bits with the probability of Pc. (3) Two parents exchange the corresponding selected bits with each other. (4) The two new individuals become the individuals of the next generation.

$< Mutation >$

Mutation is executed in one parent and a new individual is generated. The procedure of mutation is as follows.

(1) Select one individual as a parent using tournament selection. (2) Some bits are selected with the probability of Pm. The selected bits are changed randomly and the new individual is generated. (3) The new individual becomes the individual of the next generation.

Here, tournament selection runs a "tournament" among two individuals chosen at randomly from the population, and selects the winner which with the better fitness or has more different structure.

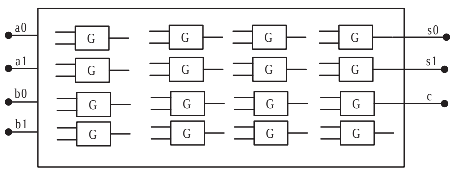

Figure 3: An initial 4*4 array with the input/output functionality for a 2-bit half adder.

3 Circuit Design Optimization using GAdss

3.1 Objective

The overall objective is to discover new and novel solutions by the application of GA in the circuit design process. The target circuit has to provide identical functional behavior equivalent to the specification but require less complexity, less power and less signal delay.

In this section, we will demonstrate the principle of GAdss in the design process using a 2-bit half adder as a sample logic circuit.

3.2 Genetic encoding

To process a genetic encoding easily, the logic circuit under consideration is assumed to be organized on a two dimensional array of cells, which was proposed in [3]. Each cell accepts two inputs and produces one output. The cells in the first column of the array are set with predefined inputs. For the purpose of this experiment, the combinatorial circuit takes four primary inputs. Therefore there are 16 input patterns of the circuit. Cells in the following columns receive outputs from cells in the previous columns. The chromosome is a string of integers where each three continuous genes embody a cell. Each triplet in the chromosome encodes the two inputs and the type of a cell respectively. In this experiment, the last cell is not used, so the chromosome length is calculated by the following formula:

$$3 * ((numberofcolumns) * (numberofrows) - 1). \quad (1)$$

In the experiment, the array is a fixed size of 4*4 cells (shown in Figure 3), thus the length of the chromosome is 45 ($3 * (4 * 4 - 1)$). The inputs of each cell in the first column of the array can take the value of any integer in the range of [0 to $(max\_number\_inputs - 1)$]. Cells in all other columns can take any integer value in the range of [0 to $((now\_column - 1) * (numberofrows) - 1)$]. As for the third gene in the triplet, cell type is defined as shown in Table 1, which was proposed in [7].

A typical chromosome then can be a sequence of triplets such as:

Table 1: Information of cells

CT[a]	LF[b]	GC[c]	EC[d]	power	EP[e]	GSD[f]	ESD[g]
0	NAND	4	6	3	7	4	6
1	NOR	4	6	3	7	4	6
2	XNOR	8	2	4	6	6	4
3	NOT(in1)	2	8	2	8	3	7
4	NOT(in2)	2	8	2	8	3	7
5	WIRE(in1)	0	10	6	4	8	2
6	WIRE(in2)	0	10	6	4	8	2
7	AND	6	4	5	5	7	3
8	OR	6	4	5	5	7	3
9	XOR	8	2	4	6	6	4
-	(not used)	0	20	0	20	0	20

[a]cell type
[b]logical function
[c]gate complexity
[d]evaluating gate complexity
[e]evaluating gate power
[f]gate signal delay
[g]evaluating gate signal delay

([0-X],[0-X],[0-9])...([0-X],[0-X],[0-9]).

Here, X is 3 in the first four cells (in the first column); X is $((now\_column - 1) * 4 - 1)$ in the other columns, now_column is the number of (2,3,4) column where the cell is placed.

3.3 Fitness Function

The fitness function in this experiment aims to accept solutions with 100% correctness of the target circuit, and with maximal evaluating values about complexity, power and signal delay. We use two functions F_1 and F_2. The former is a ratio of correct outputs to all test data, and the latter is an evaluating function of circuit complexity, power and signal delay. The following shows how the fitness of individuals is calculated, which was proposed in [7]:

$$F_1 = \frac{num\_rightout * 100}{num\_testdata}. \quad (2)$$

- $num\_rightout$: the number of correct outputs from circuit individuals.
- $num\_testdata$: the number of all test data.

$$\begin{aligned} F_2 &= (\sum_{i \in N} ecv_i) * \alpha_c + (\sum_{i \in N} epv_i) * \alpha_p \\ &+ (\sum_{j \in Cols} (\min_{k \in Rows} edv_{jk})) * \alpha_d. \end{aligned} \quad (3)$$

- ecv_i : evaluating complexity value of cell i.
- epv_i : evaluating power value of cell i.
- $edv_{j,k}$: evaluating signal delay value of cell in column j and row k in an array.
- a_c : the coefficient about complexity (set to 1 here).
- a_p : the coefficient about power (set to 1 here).
- a_d: the coefficient about signal delay (set to 1 here).
- N : the number of all the cells.
- $Cols$: all the columns of the array.
- $Rows$: all the rows of the array.

$$Fitness = \begin{cases} F_1, & if(F_1 < 100) \\ F_1 + F_2, & otherwise. \end{cases} \quad (4)$$

The first part F_1 of the fitness function compares the output response of the evolved circuit with the desired ones from truth table. If all matching, then the fitness value for the correctness is 100. The second fitness F_2 searches for the most optimum solution in terms of complexity, power and signal delay. This is done by designating gates with different evaluating values about complexity, power and signal delay (shown in Table 1).

In order to judge the difference of the complexity, power and signal delay between different gates, we assign values about complexity, power and signal delay to each gates. In the evolution of GA, the larger the fitness is, the better the circuit is. So we use evaluating values about complexity, power and signal delay in fitness function. In Table 1, "GC" is the complexity about CMOS circuit of one gate, "EC" equals to $(10 - GC)$. "power" is the value about power of one gate, "EP" equals to $(10 - power)$. "GSD" is the value about signal delay of one gate, and "ESD" equals to $(10 - GSD)$. When a cell is not used in a circuit, then its evaluating values are set to 20.

4 Experiments

This experiment aims to verify circuit optimization by GAdss. Table 2 shows the parameters of the evolution of GAdss. There is no fixed method to define the number of generations, population size, crossover probability and mutation probability. Therefore some preliminary experiments were performed in advance to decide parameters suitable for our experiment.

Table 2: Conditions for evolution

Number of Generation : 500
Population Size : 1210
Elite Size : 10
Crossover Size : 600
Mutation Size : 600
Crossover Probability (Pc) : 0.2, 0.5
Mutation Probability (Pm) : 0.023

Table 3: The results of different GA

item	GA(0.2)[a]	GAdss(0.2)[b]	GA(0.5)[c]	GAdss(0.5)[d]
best	447	465	471	501
quality	-	5.2%	-	8.1%
average	428.4	438	464.4	500.4
quantity	3	7	14	16
time(m)	19	19	25	25

[a]GA with $(Pc : 0.2)$
[b]GAdss with $(Pc : 0.2)$
[c]GA with $(Pc : 0.5)$
[d]GAdss with $(Pc : 0.5)$

The proposed method has been implemented in Eclipse SDK 3.1.1 with jre 1.6.0; and tested on a PC with Inter(R) Core(TM)2 CPU at 2.67GHz and 2.0GB RAM.

Table 3 shows the results of different GA. For each GA, we select the successful results over 60 independent trials. In Table 3, "best" means the best elite fitness value; "quality" the percent of better in evaluating value of best individual compared to one of GA; "average" the average fitness value of top three individuals; "quality" the number of correct individuals over 60 independent trials; "time" the running time of 60 trials.

From the results, we can see that the best elite fitness, the average fitness value of top three correct circuits, and the number of correct circuits of GAdss are better than GA. Compared to GA, GAdss consider the different structure from individuals comparing to elite one, then it can enhance diversification to searching spaces, so that it can find better solution.

In the experiments, the optimized circuit with fitness 501 was obtained by the GAdss (Pc:0.5). This chromosome is as follows:

(0,2,2)(0,2,7)(1,3,1)(1,3,2)(3,2,5)(0,0,5)(3,0,5)(1,3,1)
(1,6,9)(4,2,6)(3,6,1)(5,7,4)(0,6,3)(1,3,2)(2,7,1)

The graphical representation of this chromosome is shown in Figure 4. In this figure, we show the useful gates

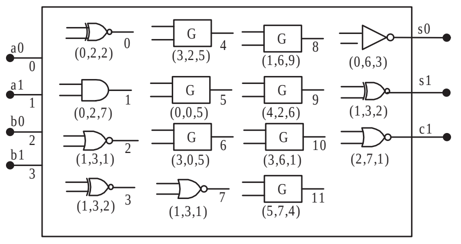

Figure 4: The graphical representation of chromosome (501).

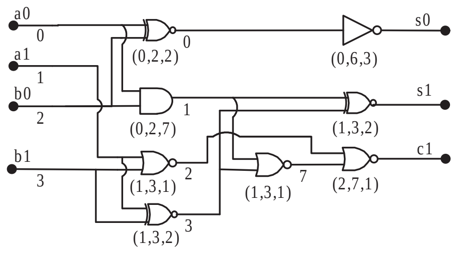

Figure 5: The optimized circuit after removing unnecessary gates (501).

with logic gates symbols. Figures 5 and 6 show the optimized circuits with fitness 501 and fitness 471 respectively, after removing unnecessary gates. The circuit in Figure 5 is obviously better than the one in Figure 6, because the former is composed of less gates, so that the lager fitness can produce a circuit with less complexity, less power and less signal delay.

5 Conclusion

This paper proposed GAdss and its application to autonomous design optimization for combinatorial circuits. By evolution, GAdss can find optimized circuits with less complexity, less power and less signal delay than GA .

We can also apply GAdss to autonomous design circuits for more complex functional requirements, and enhance more exact information about circuit to fitness function. In the future, we will develop the adaptive systems which reconfigure an existing design to adapt to a variable operational environment.

Acknowledgements

This research was supported by "Ambient SoC Global COE Program of Waseda University" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

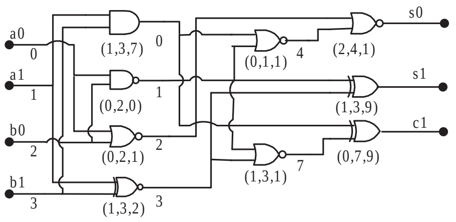

Figure 6: The optimized circuit after removing unnecessary gates (471).

References

[1] J. H. Holland, "Adaptation in Natural and Artificial systems," Ann Arbor, MI: The University of Michigan Press, 1975.

[2] Y. Chen, J. Hu and K. Hirasawa et al, "Solving Deceptive Problems Using A Genectic Algorithm with Reserve Selection," Proc. of IEEE Congress on Evolutionary Computation 2008 (CEC2008), pp.884-889, Hongkong, Jun., 2008.

[3] J. D. Lohn and G. S. Hornby, "Evolvable hardware: using evolutionary computation to design and optimize hardware systems," Computational Intelligence Magazine, IEEE, pp. 19-27, Feb., 2006.

[4] E. Benkhelifa, A. Pipe and G. Dragffy et al, "Towards evolving fault tolerant biologically inspired hardware using evolutionary algorithms," Proc. of IEEE Congress on Evolutionary Computation 2007 (CEC2007), pp. 1548-1554, Sept., 2007.

[5] X. Yao and T. Higuchi, "Promises and Challenges of Evolvable Hardware," IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 29, No. 1, pp. 87-97, Feb., 1999.

[6] V. K. Vassilev and J. E. Miller, "Scalability problems of digital circuit evolution evolvability and efficient designs," Proc. of the Second NASA/DoD Workshop on Evolvable Hardware, pp. 55-64, Palo Alto, CA, USA, Jul., 2000.

[7] Z. Bao and T. Watanabe, "A New Approach for Circuit Design Optimization using Genetic Algorithm," Proc. of International SoC Design Conference 2008 (ISOCC2008), Busan, Korea, Nov., 2008.