Design and Development of Automated Digital Circuit Structure Base on Evolutionary Algorithm Method

K.H. Chong[1] , S.P. Koh[1], S.K. Tiong[1] , K.H. Yeap[2]
[1] *Department of Electronic Communication, College of Engineering*
Universiti Tenaga Nasional, Km 7, Jalan Kajang-Puchong, 43009 Kajang, Selangor, Malaysia
ChongKH@uniten.edu.my, JohnnyKoh@uniten.edu.my, Siehkiong@uniten.edu.my

[2] *Department of Electronics Engineering, Faculty of Engineering and Green Technology*
Universiti Tunku Abdul Rahman, Jalan Universiti, Bandar Barat, 31900 Kampar, Perak, Malaysia
yeapkh@utar.edu.my

Abstract: Evolutionary Algorithms (EAs) covers all the applications involving the use of Evolutionary Computation in electronic system's design. It is largely applied to complex optimization problems. EAs introduces a new idea for automatic design of electronic systems; instead of imagine model, abstractions, and conventional techniques, it uses search algorithm to design a circuit.

In this paper, a method for automatic optimization of digital circuit design method has been introduced. This method is based on randomized search techniques mimicking natural genetic evolution. The proposed method is an iterative procedure that consists of a constant-size population of individuals, each one encoding a possible solution in a given problem space.

The structure of the circuit is encoded into one-dimensional genotype as represented by a finite string of bits. A number of bit string used to represent the wires connection between the level and 7 types of possible logic gates; XOR, XNOR, NAND, NOR, AND, OR, NOT 1 and NOT 2. The structure of gates are arranged in a m * n matrix form which m is the number of input variables.

Keywords-Digital structure design, Evolutionary Algorithm, Optimization, Genetic Algorithm

I. INTRODUCTION

One of the factors to reduce the cost of the digital circuit is to minimize the number of gate used per system. As a result, the higher the integration level, the better and the cheaper is the final product produced. Many general applications such as binary adder and subtractor have been fabricated into a single integrated circuit (IC). The level of integration becomes complicated when the number of bit per IC getting increases. Subsequently, the number of gates used per single substrate will be increased as well as the power consumption.

The conventional digital circuit design is a very complex task which requires much knowledge in domain-specific rules. The design procedures involves process such as choosing the suitable gate types to match the logical specification, minimizing and optimizing the boolean representation with respect to the user defined constraints [7].

Besides conventional method, Evolutionary Algorithm (EA) has been also widely applied to complex optimization problems. One of the applications can be presented the nation of structure design for the digital circuit [12].

Genetic Algorithm (GA) is one of the search techniques of EA. GA search technique is found by Holland [2] and extensively applied into optimization tasks by Goldberg [1]. GA is a stochastic search method that mimics the metaphor of natural biological evolution. GA operates on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in the natural adaptation.

The application of GA in combination logic design is firstly proposed by Louis [5]. In his work, he combines GA with knowledge-based systems and uses masked crossover operator to solve the combination logic circuit. His method can solve the functional output for the combination logic but not emphasize on the optimization of the gate usage. Liew used combination technique of case-based reasoning and genetic algorithm in combination logic circuit design [10]. This approach borrows ideas from case-based reasoning (CBR), the technique of reasoning and learning from prior experience. Domain knowledge acquired from an old problem is stored as cases in a case base. When combining GAs and CBR, appropriate cases are injected into an initial population and run the genetic algorithm.

The injected cases guide a GA's search by providing domain information learned from a previous search and consequently improves the current search. Her method is tested on the parity checker.

Miller used GA in designing the functional arithmetic circuits. The design involves one-bit, two-bit adders with carry, and two and three-bit multipliers and details of the 100% correct evolution of three and four-bit adders. The largest and complicated digital circuits have been designed by purely evolutionary means. The proposed algorithm is able to re-discover conventionally optimum designs for the one-bit and two-bit adders, but more significantly is able to improve on the conventional designs for the two-bit multiplier. The technique is based on evolving the functionality and connectivity of a rectangular array of logic cells and is modelled on the resources available on the Xilinx 6216 FPGA device [3].

Nilagupta proposed an approach to enhance the designing process by minimize the number of transistors used in designing a combinational logic circuit by using GA [8]. The chromosome representation is base on Louis [5] method which is a bi-dimensional matrix. Each cell in the matrix is logic gates; includes NULL, NOT, NAND, NOR and XOR. The fitness function works in two stages. The validity of the circuit outputs or the functional output is taken into account in the initial stage. GA uses to optimize the number of transistor count after the functional solution appeared. This research is mainly focused on the optimization of transistor used in the digital circuit but not the number of gate.

In this paper, we proposed a method to design the structure of the combination logic circuit using GA. This approach is able to optimize the usage of logic gate compare to the conventional method. The main objective of this research is to design a digital circuit which can produce the desired minterm which specified by user with the most minimum usage of logic gates. The types of gate used can be a combination of Wires, XOR, XNOR, NAND, NOR, AND, OR and NOT. The structure of the digital circuit is encoded into one-dimensional genotype as represented by a finite string of bits. The design process then be done by GA operator such as selection, crossover and mutation. The best solution found is decoded back to the digital circuit structure.

II. METHODOLOGY

GA is an adaptive heuristic search algorithm premised on the evolutionary ideal of natural selection and genetic. The basic concept of GA is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Darwin of survival of the fittest.

The proposed approach is represented by the solution inside the block in Figure 1. The input signals are represented by *I* and the output signals are represented by *O*. The detail solutions inside the block are encoded into a string of chromosome bits and subjected to the process of GA.

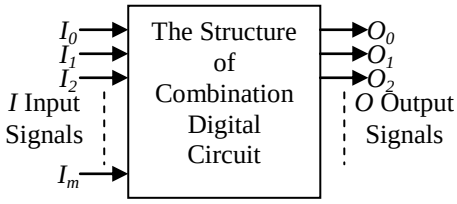

Figure 1. Block Representation of Combination Digital Circuit Structure

The architecture of the proposed system is shown in Figure 2. An encoding system is used to represent the structure of combination logic circuit. The two-dimension phenotype is encoded into one-dimension genotype as represented by a finite string of decimal bits. A group of bits string used to represent 7 types of possible logic gates and two wire connection which are listed in Table 1. Every logic gate has two inputs and one output.

Table 1. Logic Gates Representation

Bit Representation	Gate Representation
0	Wire 1
1	Wire 2
2	XOR
3	XNOR
4	NAND
5	NOR
6	AND
7	OR
8	NOT 1
9	NOT 2

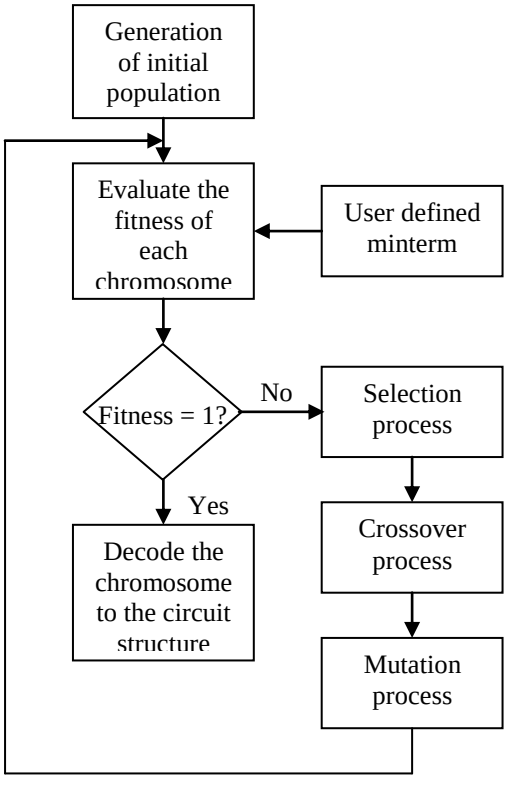

Figure 2. Block Diagram of the Proposed System

The structures of gates are arranged in $G_{x,y}$ matrix form in which x represents the gate in row and y represents the gate in the column. The m inputs are connected to x row of the logic gates. The possible input combination is based on 2^m. The output constraint signal is obtained at $y+n$ column of the logic gates; n is the number of column.

$$\begin{bmatrix} G_{x,y} & G_{x,y+1} & G_{x,y+2} & \cdots & G_{x,y+n} \\ G_{x+1,y} & G_{x+1,y+1} & \cdots & & G_{x+1,y+n} \\ G_{x+2,y} & \cdots & & & \\ \vdots & & & & \\ G_{x+m,y} & G_{x+m,y+1} & \cdots & & G_{x+m,y+n} \end{bmatrix} \quad (1)$$

We define that $G_{x,y+1}$ gate always obtain one signal from the output of $G_{x,y,}$ gate and another signal from $G_{x+1,y}$ gate, while $G_{x+m,y+1}$ obtain the signals from the output of $G_{x+m,y}$ and $G_{x,y}$. The system process is started with the random generation of initial population. The length of the chromosomes depends on the size of the entire structure. The number of bit in the chromosomes represents type of gate. The collection of all individuals of the cell represents a solution.

The algorithm for the proposed system shows in Figure 2 is as follows:

```
begin
  create a random initial population N;
  evaluate the fitness of each chromosome in the
  population;
  while (there is no design with fitness = 1.0) do
    conduct selection, crossover and mutation
    replace the old generation by the new generation of
    designs
    re-evaluate fitness of the designs in the population
  end while;
  decode the chromosome with fitness = 1.0
end
```

The design has fitness equal to 1 if it output value satisfies the constraint specification. Otherwise the structure design is deviated from the performance specification. The fitness of the process is the correctness of the obtained logic circuit in matching the truth table of the required function. In this work, we proposed the fitness functional called Constraint Fitness (CF). CF is based on the comparison of circuit output with the constraint output. The formula for CF is:

$$x = \sum_{i=1,2,3,\ldots.\ 2^m}^{m} \left| c_o - f_o \right| \quad (2)$$

$$F_C = \frac{1}{1+x} \quad (3)$$

F_C achieves 1 when x is zero. c_o is the constraint output set by the user and f_o is the functional output generated from the circuit. When the functional output is exactly equal to the constraint output, this will cause (2) equals to 0. Consequently, F_C will be equalled to 1 due to x equal to zero.

It is possible to get more that one constraint fitness in a population. In order to determine the minimum gate use of the design, we accumulated all the population with the constraint fitness equal to 1 for further evaluation.

In this work, we are targeting the optimization of a number of gates used in the structure design. After obtaining CF, the second objective is to get the fitness of gate optimization used in the circuit design. The population for this Gate Optimization Fitness (GOF) evaluation is based on the chromosome with CF of 1. Each string of chromosome is evaluated for the gate optimization fitness. The algorithm for this fitness is:

$$F_{GO} = \frac{\sum G_{F_C=1}}{\sum G_{x+m,y+n}} \quad (4)$$

$$0 < F_{GO} < 1 \quad (5)$$

The numerator part of (4) is the summation of the gates which can produce the constraint fitness equal to 1. The denominator part is the summation of the gate per structure. The fitness of GOF should be in between 0 and 1. The global optimal is achieved when the smaller fitness of GOF is met.

III. EXPERIMENTAL RESULTS

Experiments have been conducted using the system to investigate its effectiveness in constraint-directed logic synthesis. The results produced by the proposed approach were compared with those generated by Karnaugh Minimizer [9] and Boolean algebra [7]. GA approach can generate the design of the logic circuit with using XOR or XNOR directly without further optimize. Both of these conventional methods need human effort to optimize the function.

The experiments use population size N = 1000, percentage of crossover reproduction C= 70, percentage of mutation reproduction M = 50.

The author used the proposed method to design a 3-bit circuit with the minterm F(a, b, c) = Σ(0, 3, 5, 6). Part of the result is illustrated in Table 2.

Table 2. The Generated Error and CF for Each Chromosome

Chromosome						CF
1	4	1	0	2	3	0.142857
3	4	2	4	4	0	0.166667
4	1	2	4	4	3	0.166667
4	3	4	4	2	0	0.166667
1	2	4	2	0	0	0.200000
2	2	0	1	0	3	0.200000
1	4	4	4	1	2	0.200000
4	2	0	0	4	2	0.200000
4	1	1	3	1	1	0.200000
1	0	2	2	1	4	0.200000
3	3	4	1	0	1	0.200000
1	3	3	0	4	0	0.200000
0	4	4	0	1	3	0.200000
0	0	1	4	4	4	0.200000
0	0	0	3	1	0	0.200000
2	0	4	3	1	0	0.200000
2	3	3	3	0	4	0.200000
2	4	1	2	0	2	0.200000
4	4	4	2	2	1	0.333333
1	0	3	0	2	1	1.000000

The system is evaluated CF for each of the chromosome string. CF is equalled to 0.142857 at the early stage. The chromosome string is evolved through GA process to improve CF. The table values show that CF value increases and eventually achieved value of 1. The plot of CF versus number of generation is illustrated in Figure 3.

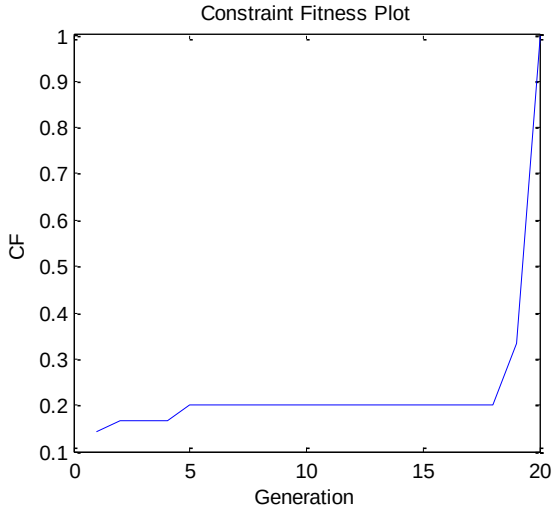

Figure 3. The plot of CF versus number of generation

Moreover, there more than one combination of gates that can produce the desired output bit. The chromosome bits are shown in Table 3 together with it corresponding output function and GOF value.

Table 3. The Required No of Gate and GOF for the Digital Circuit with F(a, b, c) = Σ(0, 3, 5, 6)

Chromosome						Output Function	No Gate	GOF
2	3	0	3	3	3	$(((a+b)\oplus(b+c)')'\oplus((b\oplus c)'\oplus c)')'$	5	0.8333
1	4	3	2	2	2	$(b\oplus(bc)')\oplus((bc)'\oplus(a+c)')$	5	0.8333
1	0	2	3	3	3	$((b\oplus b)'\oplus(b(a\oplus c))')'$	4	0.6666
1	0	2	4	1	2	$b'\oplus(a\oplus c)$	3	
1	0	3	0	2	1	$(b\oplus(a\oplus c)')'$	2	0.3333

The last chromosome string produces lower GOF compares with others. The chromosome bits are 1 0 3 0 2 1. After decoded the chromosome, the resulted circuit which can generate the corresponding minterm is shown in Figure 4.

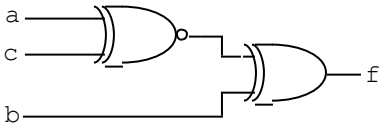

Figure 4. The Schematic Diagram of the Digital Circuit with F(a, b, c) = Σ(0, 3, 5, 6)

The output function of the circuit in Figure 4 is $f = (b \oplus (a \oplus c)')'$. The circuit needs only 1 XOR and 1 XNOR to generate the desired output bits.

Besides, three 3-bit, two 4-bit and a 5-bit digital circuits were designed using the proposed method as well. The minterm of the circuit is as follows:

D1: F(a,b,c) = Σ(3, 5, 6)
D2: F(a,b,c) = Σ(3, 5, 6, 7)
D3: F(a,b,c) = Σ(0, 3, 4, 5, 6)
D4: F(a,b,c,d) = Σ(2, 3, 5, 6, 8, 9, 12, 15)
D5: F(a,b,c,d) = Σ(7, 10, 11, 13, 14, 15)
D6: F(a,b,c,d,e) = Σ(1, 2, 4, 7, 8, 11, 13, 14, 16, 19, 21, 22, 25, 26, 28, 31)

The results for the circuit designed by the proposed method, Karnaugh Minimizer and Boolean algebra are shown in Table 5, Table 6 and Table 7 respectively. The tables are compared based on the output function and gates count for the circuit designed by each of the methods. The gates count for all designs are based on 2 inputs gate listed in Table 1. These results show that the proposed system is able to produce effective solution of constraint-directed logic optimization.

Table 5. The Output Function and Gates Count for the Circuit Designed by the Proposed Method

D	Output Function	Gates count
1	$(((c \oplus (a \oplus b)) + (a + c)')')'$	4
2	$ab + (a + b)c$	4
3	$(a + b)' + (b \oplus (a \oplus c)')$	4
4	$(((bd) \oplus a) \oplus c)$	3
5	$(c + (bd))(a + (c \cdot (bd)))$	5
6	$(a \oplus b)' \oplus ((c \oplus d) \oplus e')'$	5

Table 6. The Output Function and Gates Count for the Circuit Designed by Karnaugh Minimizer

D	Output Function	Gates count
1	$ab'c + a'bc + abc'$	8
2	$ab + ac + bc$	5
3	$ac' + ab' + b'c' + a'bc$	8
4	$a'cd' + a'b'c + ab'c' + ac'd' + a'bc'd + abcd$	19
5	$ac + bcd + abd$	7
6	$a'b'c'd'e + a'b'c'de' + a'b'cd'e' + a'b'cde + a'bc'd'e' + a'bc'de + a'bcd'e + a'bcde' + ab'c'd'e' + ab'c'de + ab'cd'e + ab'cde' + abc'd'e + abc'de' + abcd'e' + abcde$	79

Table 7. The Output Function and Gates Count for the Circuit Designed by Boolean Algebra

D	Output Function	Gates count
1	$c(a \oplus b) + abc'$	5
2	$c(a \oplus b) + ab$	4
3	$b'c' + a'bc + a(b \oplus c)$	7
4	$b'(a \oplus c) + b(a \oplus (c \oplus d))$	6
5	$ab(a \oplus b) + ac(b \oplus d)' + ab(c \oplus d)$	10

The circuit structure for D1 design requires 4 gates; 1 OR, 1 NOR and 2 XOR if designed by the proposed method. The design needs more gate if designed by Karnaugh Minimizer, which is 8 gates; 2 OR and 6 AND. While Boolean algebra approach only needs 5 gates; 1 XOR, 1 OR and 3 AND. The schematic diagram for the designs is illustrated in Figure5, Figure 6 and Figure 7 respectively.

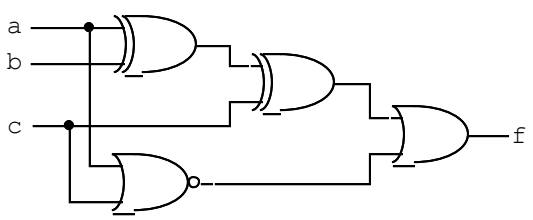

Figure 5. The schematic diagram for D1 which is designed by the proposed method.

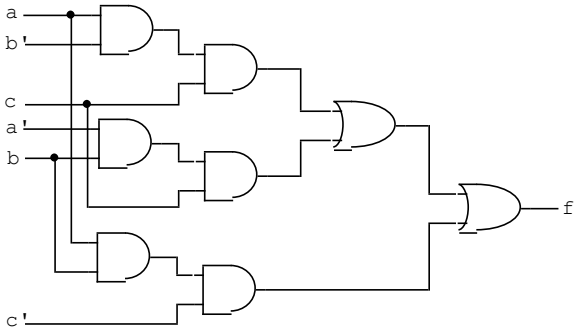

Figure 6. The schematic diagram for D1 which is designed by Karnaugh Minimizer.

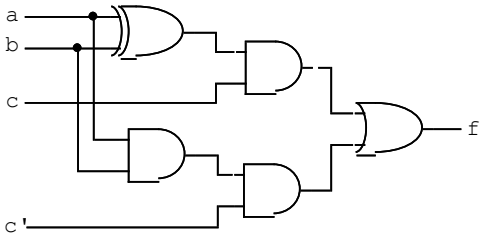

Figure 7. The schematic diagram for D1 which is designed by Boolean algebra.

For D2, the proposed method needs only 4 gates to generate the desired minterm; 2 AND and 2 OR. The same output specification was designed by Karnaugh Minimizer as well. The total number of gates required is 5; 3 AND and 2 OR. Boolean algebra method also needs 4 gates; which is 1 XOR, 1 OR and 2 AND. Boolean algebra and the proposed method used the same number of gate in this design. The schematic diagram for the designs is shown in Figure 8, Figure 9 and Figure 10 respectively.

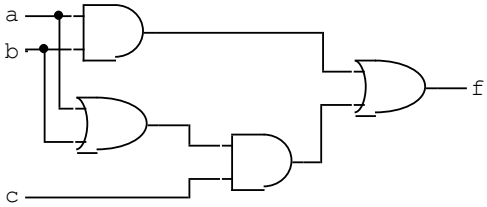

Figure 8. The schematic diagram for D2 which is designed by the proposed method.

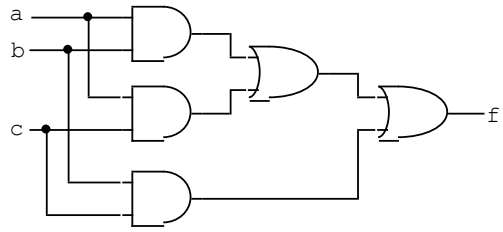

Figure 9. The schematic diagram for D2 which is designed by Karnaugh Minimizer.

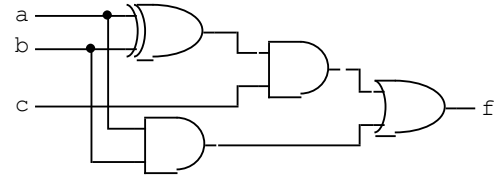

Figure 10. The schematic diagram for D2 which is designed by Boolean algebra.

D3 is a 3-bit digital circuit, the design requires 4 difference type of gates if designed by the proposed method; 1 OR, 1 NOR, 1 XOR and 1 XNOR. The design needs more gate by Karnaugh Minimizer, which are 8 gates; 3 OR and 5 AND. While Boolean algebra approach requires one gate lesser than Karnaugh Minimizer; 1 XOR, 2 OR and 4 AND. The schematic diagram for the designs is illustrated in Figure 11, Figure 12 and Figure 13 respectively.

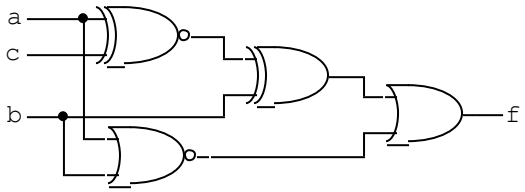

Figure 11. The schematic diagram for D3 which is designed by the proposed method.

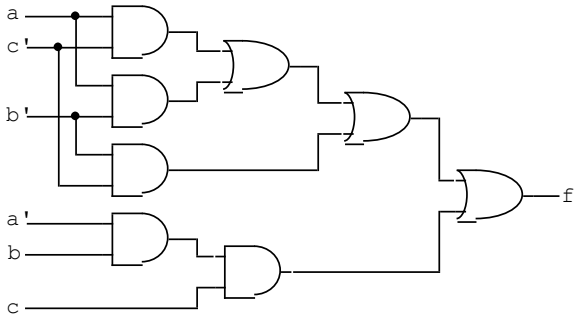

Figure 12. The schematic diagram for D3 which is designed by Karnaugh Minimizer.

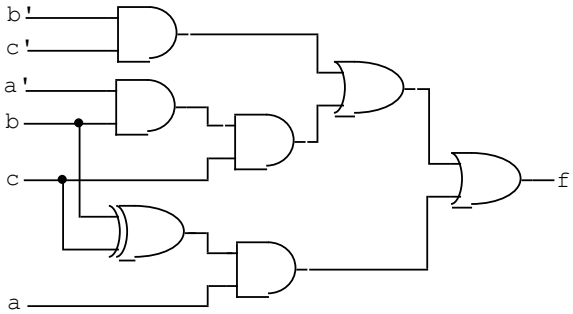

Figure 13. The schematic diagram for D3 which is designed by Boolean algebra.

For D4 design, the proposed method needs only 3 gates to generate the desired output; 2 XOR and 1 AND. The same output specification was designed by Karnaugh Minimizer. The total number of gates required is 19; 14 AND and 5 OR gates. While Boolean algebra approach needs 6 gates; 1 OR, 2 AND and 3 XOR. The schematic diagrams for the designs are illustrated in Figure 14, Figure 15 and Figure 16 respectively.

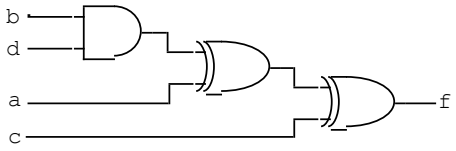

Figure 14. The schematic diagram for D4 which is designed by the proposed method.

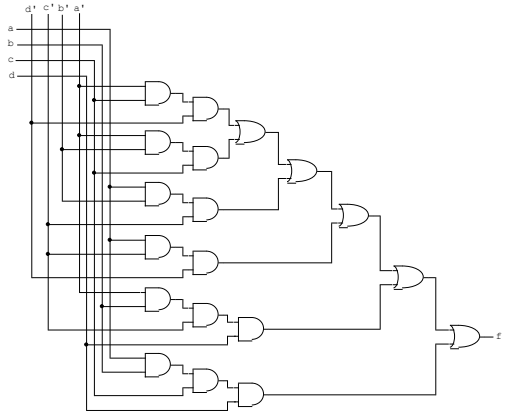

Figure 15. The schematic diagram for D4 which is designed by Karnaugh Minimizer.

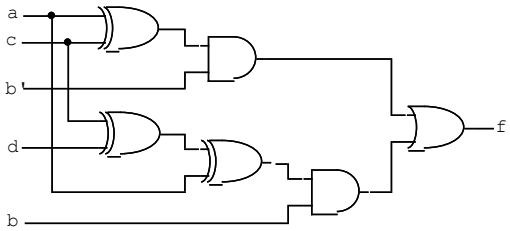

Figure 16. The schematic diagram for D4 which is designed by Boolean algebra.

For D5, in order to produce the expected output bit, the proposed method just need 5 gates; 2 AND and 3 OR. Moreover, Karnaugh Minimizer approach, the total number of gates required is 7; 2 OR and 5 AND. Boolean

algebra method needs 10 gates; which is 1 XNOR, 2XOR, 2 OR and 5 AND. The schematic diagram for the designs is shown in Figure17, Figure 18 and Figure 19 respectively.

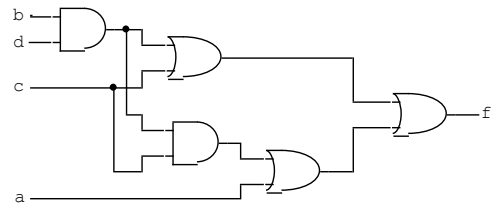

Figure 17. The schematic diagram for D5 which is designed by the proposed method.

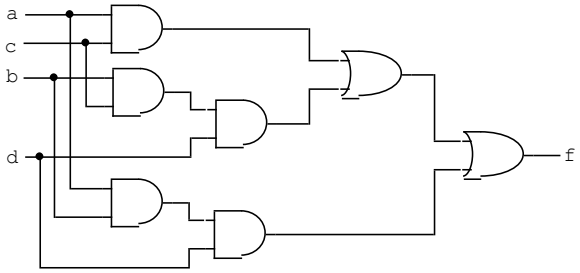

Figure 18. The schematic diagram for D5 which is designed by Karnaugh Minimizer.

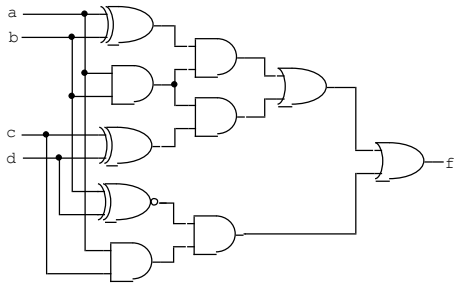

Figure 19. The schematic diagram for D5 which is designed by Boolean algebra.

D6 is a 5-bit digital circuit design. Obviously, the proposed method can give the optimized design compare to Karnaugh Minimizer. The circuit design only need 5 gates, which are 1 NOT, 2 XOR and 2 XNOR to generate the desired minterm, while Karnaugh Minimizer requires up to 79 gates; 64 AND and 15 OR. In the other hand, the required minterm of this design is too complicated to solve by Boolean algebra approach. Therefore, the authors only used the proposed method and Karnaugh Minimizer to generate the circuit design.

By referring to the experiment results, the proposed method is able to design a circuit in nonstandard output function and optimize the circuit by using XOR and XNOR gates. In fact, the optimization using XOR and XNOR can be done by Boolean algebra approach as well, but this procedure of minimization is awkward because it lacks specific rules to predict each succeeding step in the manipulative process, and this is the limitation of Boolean algebra.

Besides, the result also shown that the proposed method can give better design compare to Boolean algebra. Karnaugh Minimizer needs more gates because it can only give the circuit output function in sum of product (SOP) form.

In fact, the output function produced by Karnaugh Minimizer can be further optimized by applying Boolean algebra theorem. However, for complicated circuit design, Boolean algebra theorem may not be a practical option to be implemented. Besides, Karnaugh Minimizer unable to optimize the logic function using XOR and XNOR operation.

Furthermore, the number of gate count for Karnaugh Minimizer and Boolean algebra method is excluded NOT gate which is used to generate the compliment input to the circuit. The authors assume that the input variables are directly available in their complement form, so NOT gates are not included in the design. Therefore, the number of gates count for both of these designs should be more if take consideration of the compliment inputs.

IV. CONCLUSION

The experiment results show that the proposed method is able to optimize the gate count in the digital circuit design. The advantages of the proposed method are as follows:

i) The proposed method can generate the circuit output function in nonstandard form.
ii) The proposed method can optimize the circuit output function with XOR and XNOR function.
iii) The proposed method use decimal number instead of binary number for the chromosome representation. Therefore, the process duration can be reduced.
iv) The proposed method can archive the functional and optimal solution within a small population of chromosome as well as the number of generation.
v) The propose method offer flexibility of gate selection in the design. For example, the designer can select only 4 types out of 8 types of gate that is listed in Table 1 by limiting the bit representation in the chromosome.

In conclusion, the obtained result shows that the proposed method manages to produce the digital circuit structure with lesser gate count. EA method can produce the global optimal solution but the drawback of this method is the process duration become longer for more complicated structure.

REFERENCES

[1] David, E. Goldberg. Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley, 1989.
[2] Holland, J. H., Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, Cambridge, Massachusetts, 1992.
[3] J. F. Miller, P. Thomson, T. Fogarty, "Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study". Genetic Algorithms Recent Advancements and Industrial Applications, Quagliarella, D. et al., Eds., John Wiley & Son, New York, 1997.

[4] Ram Vemuri and Ranga Vemuri. Using Genetic Algorithm for Constraint-Directed Design of Digital Logic Circuits. Electronics Letters, 1994.

[5] Sushil J. Louis. Genetic Algorithms as a Computational Tool for Design. Ph.D. thesis, Department of Computer Science, Indiana University, 1993.

[6] Sadiq M. Sait, Mostafa Abd-El-Barr, Uthman Al-Saiari and Bambang A. B. Sarif. Fuzzified Simulation Evolution Algorithm for combination digital logic design targeting Multi-Objective Optimization. IEEE Congress on Evolutionary Computation, 2002.

[7] Mano, M. M.. Digital Design, Prentice Hall, 2002.

[8] Pradondet Nilagupta, Nuchtiphong Ou-thong, "Logic Function Minimization base on Transistor Count using Genetic Algorithm". ICEP2003 GeneticAlgorithm,Transistor minimization, Digital Circuit Design, Automatic Design, 01 2546, 2003.

[9] Shuriksoft Software, "Karnaugh Minimizer", commercial software, http://karnaugh.shuriksoft.com/

[10] Xiaohua Liu and Sushil J. Louis., 1996. *Combining genetic algorithms and case-based reasoning for structure design*. In M. E. Cohen and D. L. Hudson, editors, Proceedings of the ISCA Eleventh International Conference on Computers and their Applications, pages 103--106. ISCA.

[11] Zebulum, R. S., Pacheco, M. A. C., Vellasco, M. M. B.R. Evolutionary Electronics: Automatic Design of Electronic Circuit and Systems by Genetic Algorithms. CRC Press, 2002.