Design of Digital Circuit Structure Based on Evolutionary Algorithm Method

K. H. Chong*, I. B. Aris, S. M. Bashi** and S. P. Koh*****

Abstract – Evolutionary Algorithms (EAs) cover all the applications involving the use of Evolutionary Computation in electronic system design. It is largely applied to complex optimization problems. EAs introduce 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 the 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 a one-dimensional genotype as represented by a finite string of bits. A number of bit strings is 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 an m * n matrix form in which m is the number of input variables.

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

1. Introduction

One of the factors to reduce the cost of the digital circuit is to minimize the number of gates 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 the binary adder and subtractor have been fabricated into a single integrated circuit (IC). The level of integration becomes complicated when the number of bits per IC 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 involve such processes as choosing suitable gate types to match the logical specification and minimizing and optimizing the boolean representation with respect to the user defined constraints [7].

Besides the conventional method, the Evolutionary Algorithm (EA) has been widely applied to complex optimization problems. One of the applications can be presented as the nation of structure design for the digital circuit [11].

Genetic Algorithm (GA) is one of the search techniques of EA. The GA search technique was founded 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 by 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 procedure 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 was 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 does not place emphasis on the optimization of gate usage.

Liew utilized the 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

* Department of Physic & Science, Faculty of Engineering & Science, Setapak Campus Universiti Tunku Abdul Rahman. (chongkh@mail.utar.edu.my)

** Department of Electrical & Electronics Engineering, Faculty of Engineering Universiti Putra Malaysia. ([ishak,senan]@eng.upm.edu.my)

*** College of Engineering, Universiti Tenaga Malaysia (UNITEN).(johnnykoh@uniten.edu.my)

Received 2 March, 2007 ; Accepted 4 September, 2007

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 improving 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, two and three-bit multipliers, and details of the 100% correct evolution of three and four-bit adders. The largest and most 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, it 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 modeled on the resources available on the Xilinx 6216 FPGA device [3].

Nilagupta proposed an approach to enhance the designing process by minimizing the number of transistors used in designing a combinational logic circuit by using GA [8]. The chromosome representation is based on Louis' [5] method, which is a bi-dimensional matrix. Each cell in the matrix is a logic gate, and it 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 is used to optimize the number of transistor counts after the functional solution has appeared. This research is mainly focused on the optimization of the transistor used in the digital circuit but not on the number of gates.

In this paper, we propose a method to design the structure of the combination logic circuit using GA. This approach is able to optimize the usage of logic gate compared to the conventional method. The main objective of this research is to design a digital circuit that can produce the desired minterm specified by user with the minimum usage of logic gates. The types of gates 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 is then performed by GA operator such as selection, crossover, and mutation. The best solution found is decoded back to the digital circuit structure.

2. Methodology

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

The architecture of the proposed system is shown in Fig. 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 bit strings is 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.

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 and n is the number of columns.

We define that $G_{x,y+1}$ gate always obtains one signal from the output of $G_{x,y,}$ gate and another signal from the $G_{x+1,y}$ gate, while $G_{x+m,y+1}$ obtains 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 bits in the chromosomes represents type of gate. The collection of all individuals of the cell represents a solution.

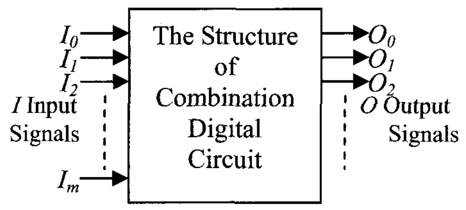


Fig. 1. Block Representation of Combination Digital Circuit Structure

Table 1. Logic Gate 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

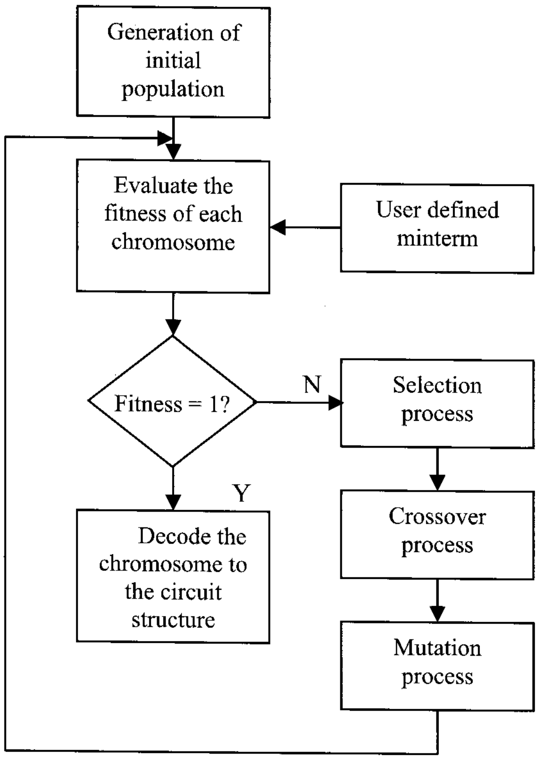

Generation of initial population
Evaluate the fitness of each chromosome
User defined minterm
Fitness = 1?
N
Y
Selection process
Crossover process
Mutation process
Decode the chromosome to the circuit structure

Fig. 2. Block Diagram of the Proposed System

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

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

```
begin
   create a random initial population $N$;
      evaluate the fitness of each chromosome in the population;
      (there is no design with fitness = 1.0)
   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
      decode the chromosome with fitness = 1.0
end
```

The design has fitness equal to 1 if its 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:

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

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

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) to equal 0. Consequently, F_C will be equalled to 1 due to x equal to zero.

It is possible to get more than 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.

3. Experiment 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 optimization. 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, and 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.

The system is evaluated using CF for each of the chromosome strings. CF is equal to 0.142857 at the early stage. The chromosome string is evolved through the GA process to improve the CF. The table values indicate that CF increases and eventually achieves the value of 1. The plot of CF versus number of generations is illustrated in Figure 3.

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

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

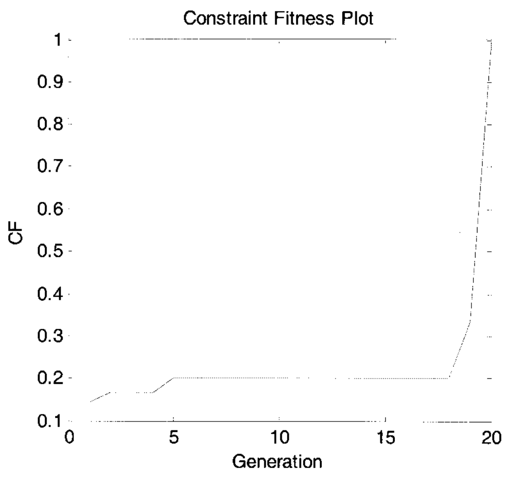


Fig. 3. The plot of CF versus number of generation

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.

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

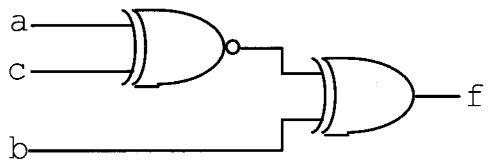


Fig. 4. The Schematic Diagram of the Digital Circuit with F(a, b, c) = Σ(0, 3, 5, 6)

Table 3. The Required No. of Gates 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

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

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

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 for the output function and gate count for the circuit designed by each of the methods. The gate count for all designs is based on 2 input gates 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 Gate Count for the Circuit Designed by the Proposed Method

D	Output Function	Gate 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 Gate Count for the Circuit Designed by Karnaugh Minimizer

D	Output Function	Gate 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 Gate Count for the Circuit Designed by Boolean Algebra

D	Output Function	Gate 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 gates if designed by Karnaugh Minimizer, which is 8 gates in total; 2 OR and 6 AND while Boolean algebra approach requires 5 gates in total; 1 XOR, 1 OR, and 3 AND. The schematic diagram for the designs is illustrated in Figure 5, Figure 6, and Figure 7, respectively.

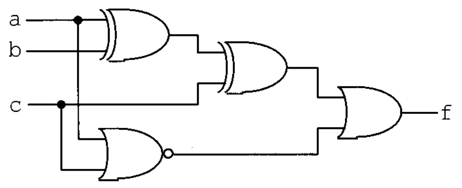


Fig. 5. The schematic diagram for D1 which is designed by the proposed method.

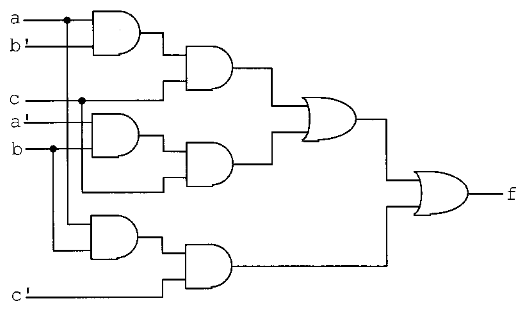


Fig. 6. The schematic diagram for D1 which is designed by Karnaugh Minimizer.

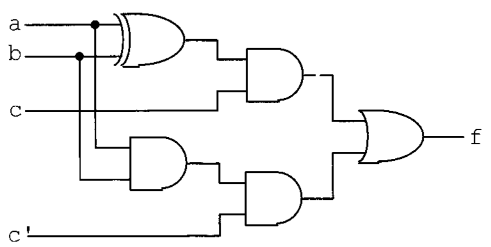


Fig. 7. The schematic diagram for D1 which is designed by Boolean algebra.

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; 1 XOR, 1 OR, and 2 AND. Boolean algebra and the proposed method used the same number of gates in this design. The schematic diagram for the designs is shown in Figure 8, Figure 9, and Figure 10, respectively.

D3 is a 3-bit digital circuit, the design of which requires 4 different types of gates if designed by the proposed method; 1 OR, 1 NOR, 1 XOR, and 1 XNOR. The design needs more gates by Karnaugh Minimizer, 3 OR and 5 AND for a total of 8. On the other hand, Boolean algebra approach requires one gate less 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.

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, 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.

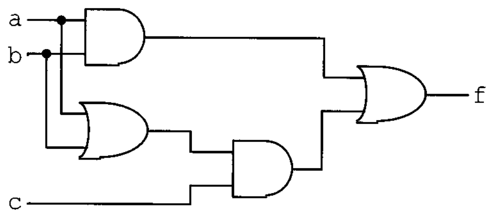


Fig. 8. The schematic diagram for D2 which is designed by the proposed method.

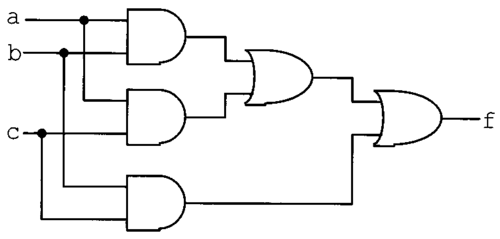


Fig. 9. The schematic diagram for D2 which is designed by Karnaugh Minimizer.

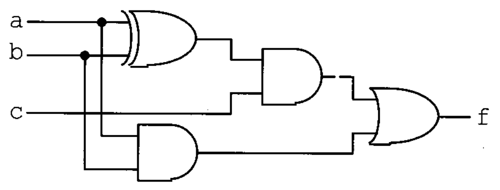


Fig. 10. The schematic diagram for D2 which is designed by Boolean algebra.

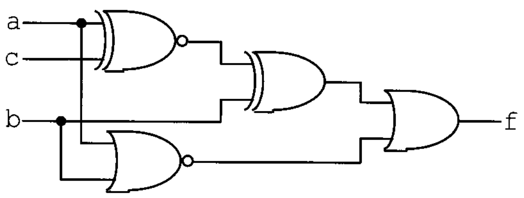


Fig. 11. The schematic diagram for D3 which is designed by the proposed method.

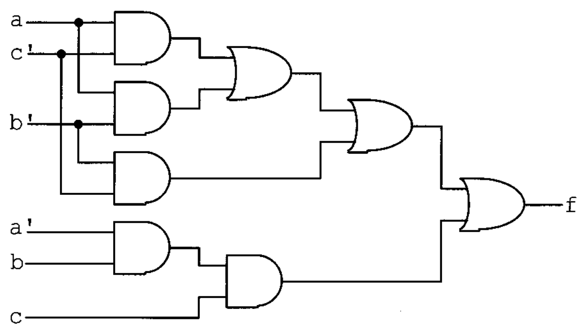


Fig. 12. The schematic diagram for D3 which is designed by Karnaugh Minimizer

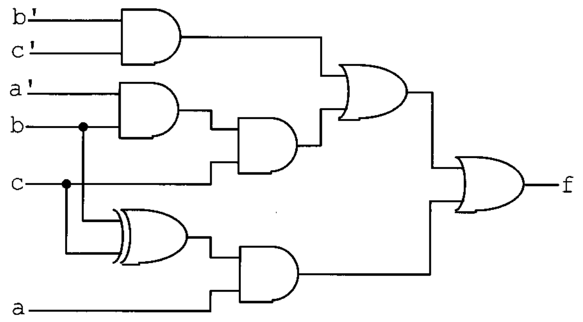


Fig. 13. The schematic diagram for D3 which is designed by Boolean algebra.

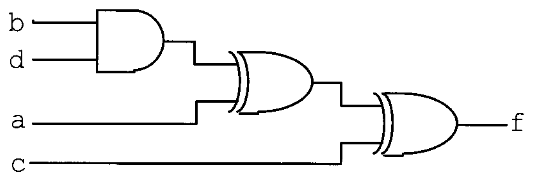


Fig. 14. The schematic diagram for D4 which is designed by the proposed method.

For D5, in order to produce the expected output bit, the proposed method just requires 5 gates; 2 AND and 3 OR. Moreover, in the case of Karnaugh Minimizer approach, the total number of gates required is 7; 2 OR and 5 AND. Boolean algebra method needs 10 gates; 1 XNOR, 2XOR, 2 OR, and 5 AND. The schematic diagram for the designs is shown in Figure 17, Figure 18 and Figure 19, respectively.

D6 is a 5-bit digital circuit design. Obviously, the proposed method provides optimized design compared to Karnaugh Minimizer. The circuit design only needs 5 gates, 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. On 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.

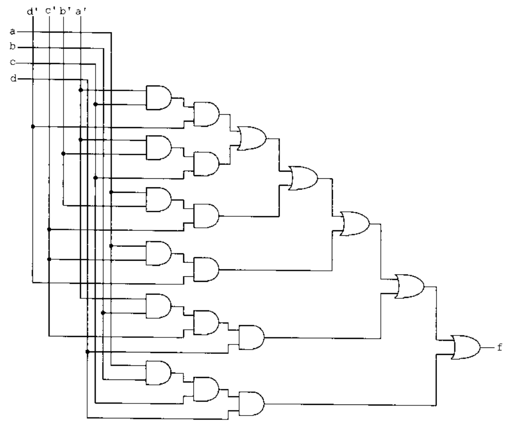


Fig. 15. The schematic diagram for D4 which is designed by Karnaugh Minimizer.

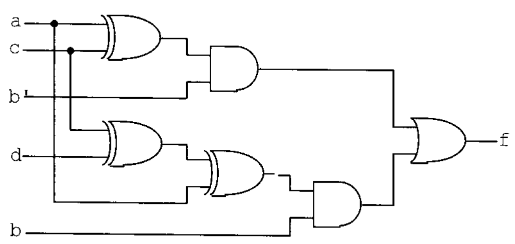


Fig. 16. The schematic diagram for D4 which is designed by Boolean algebra.

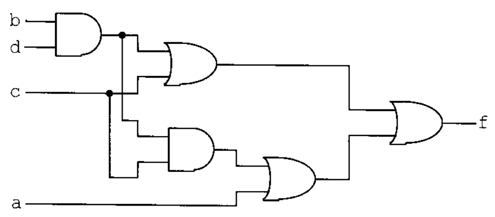


Fig. 17. The schematic diagram for D5 which is designed by the proposed method.

By referring to the experiment results, the proposed method is able to design a circuit in nonstandard output function and can optimize the circuit by using XOR and XNOR gates. In fact, the optimization using XOR and XNOR can be performed 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.

Furthermore, the result also indicates that the proposed method can give better design compared 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 is unable to optimize the logic function using XOR and XNOR operation.

Moreover, the number of gate counts for Karnaugh Minimizer and Boolean algebra method exclude the 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 gate counts for both of these designs should be more if consideration is given to the compliment inputs.

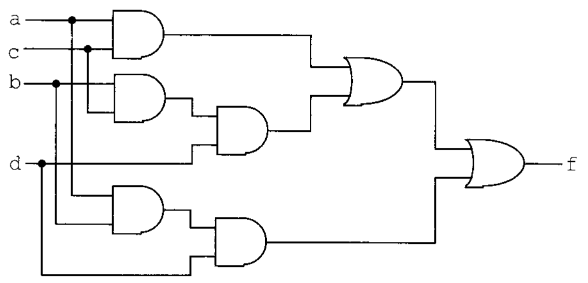


Fig. 18. The schematic diagram for D5 which is designed by Karnaugh Minimizer.

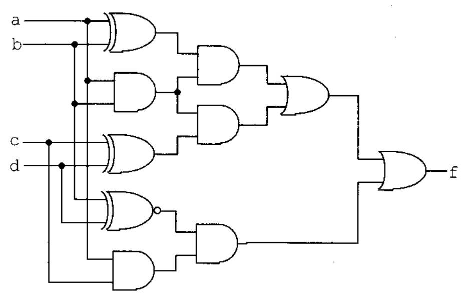


Fig. 19. The schematic diagram for D5 which is designed by Boolean algebra.

4. 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 uses decimal numbers instead of binary numbers 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 chromosomes as well as the number of generation.
v) The proposed method offers flexibility of gate selection in the design. For example, the designer can select only 4 types out of 8 types of gates that are 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. The EA method can produce the global optimal solution but the drawback of this method is that the process duration becomes longer for more complicated structure.

Reference

[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.

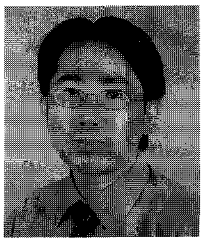
[7] Mano, M. M.. Digital Design, Prentice Hall, 2002.

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

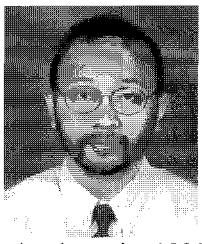
[9] Shuriksoft Software, "Karnaugh Minimizer", commercial software, http://karnaugh.shuriksoft.com/

[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.

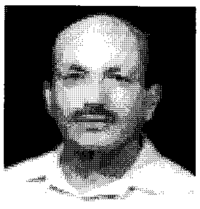
[11] Zebulum, R. S., Pacheco, M. A. C., Vellasco, M. M. B.R. Evolutionary Electronics: Automatic Design of Electronic Circuit and Systems by Genetic Algorithms. CRC Press, 2002.



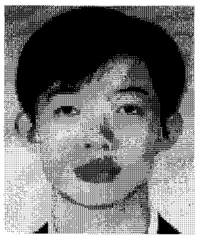
K. H. Chong, graduated with a B.Eng (Hons) in Electronics and Electrical, and a M.Sc. from the University Putra Malaysia in Electrical Power (2000) and Electronic Engineering (2002), respectively. His current research interests include artificial intelligence, evolutionary electronics, industrial process control, and automatic control systems.



Ishak Bin Aris received his B.Sc. in Electrical Engineering from the George Washington University, USA in 1988. He also received his M.Sc. and Ph.D. degrees in Power Electronics Engineering from the University of Bradford, United Kingdom, in 1991 and 1995, respectively. His areas of interest include power electronics and drive systems, robotics, artificial intelligence, SOC, and automotive electronics.



S. M. Bashi, graduated from the University of Mosul in Electrical and Electronics Engineering (1969). He received his Ph.D. in Simulation of Power Transmission System from Loughborough University of Technology, England (1980). His areas of research interest include power system analysis and design, quality of power supply, simulation and application of power electronics systems, and machines drives.



S. P. Koh, graduated with a B.Eng (Hons) in Electronics and Electrical, and a M.Sc. from Universiti Putra Malaysia in Control and Automation. Since 2002, he has been with the Department of Electrical and Electronics Engineering, Faculty of Engineering, Universiti Tenaga Nasional, Malaysia. His research interests include artificial intelligent, lasers, advanced mechatronics and control systems.