Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 7 (3): 380-385, 2007
ISSN 1812-5654

Digital Circuit Structure Design via Evolutionary Algorithm Method

[1]K.H. Chong, [2]I.B. Aris, [2]M.A. Sinan and [2]B.M. Hamiruce
[1]Department of Electronic Engineering, Faculty of Engineering and Science, Setapak Campus
Universiti Tunku Abdul Rahman, Jalan Genting Klang, Setapak, 53300 Kuala Lumpur, Malaysia
[2]Department of Electrical and Electronics Engineering, Faculty of Engineering,
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

Abstract: In this study, a new method for automatic optimization of digital circuit design method has been introduced. This method is base 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 8 types of possible logic gates, Wire 1, Wire 2, NOT 1, NOT 2, XOR, XNOR, NAND, NOR, AND and OR. The structure of gates are arranged in a m * n matrix form which m is the number of input variables. The experimental results have shown that this method can produce the circuit design base on user specified performance requirement. The representation approach also has been implemented with a computer program which can give better achievement in terms of quality solution and speed of convergence.

Key words: Digital structure design, evolutionary electronics, optimization, evolutionary algorithm

INTRODUCTION

Digital system plays a major role in the recent electronic technology due to many advantages. The main component to form a digital system is the combination of various types of logic gates. 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.

Mano (2002) commented that the conventional digital circuit design is a very complex task which required much knowledge in domain-specific rules. The design procedures involve the 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.

Besides conventional method, Evolutionary Algorithm (EA), have been widely applied to complex optimization problems as reported by Zebulum *et al.* (2002). One of the applications can be presented the nation of structure design for the digital circuit. The EA was used to conceive arrangements of digital gates that could solve a specific problem, such as the parity function. This ideal was highly original, since the evolutionary algorithm performed thorough automatic circuit design from scratch; no human knowledge was needed.

GA search technique is found by Holland (1992) and extensively applied into optimization tasks by Goldberg (1989). GA is a stochastic search method that mimic the metaphor of natural biological evolution. Evolutionary algorithms operate 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 (1993). In his research, he combines GA with knowledge-based systems and uses

Corresponding Author: I.B. Aris, Department of Electrical and Electronics Engineering, Faculty of Engineering,
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia
Tel: 603-86567121/6324 Fax: 603-86567099

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.

Sadiq *et al.* (2002) were implemented Fuzzified Simulation Evolution (SimE) Algorithm for combination digital logic design. They proposed two fitness operator for functional fitness and objective fitness. The functional fitness called Multilevel Logic Based Goodness Measure which is based on the assumption that the higher the level of a gate in a multilevel logic circuit, the more minterms are covered at the output of that gate. Therefore, the goodness of a gate is affected by the number of minterms covered at its output and the level where the gate is located. The objective fitness targeting the optimization in term of area, delay and power consumption.

Nilagupta and Ou-Thong (2003) proposed an approach to enhance the designing process by minimizing the number of transistors used in designing a combinational logic circuit by using GA. The chromosome representation is based on Louis 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 study, we proposed a method to design the structure of the combination logic circuit using EA. 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 output function with the most minimum usage of logic gates. The types of gate used can be a combination of Wire, NOT, XOR, XNOR, NAND, NOR, AND and OR. 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 the Genetic Algorithm (GA) operator such as crossover, mutation and selection. The best solution found is decoded back to the digital circuit structure.

APPROACH AND METHODS

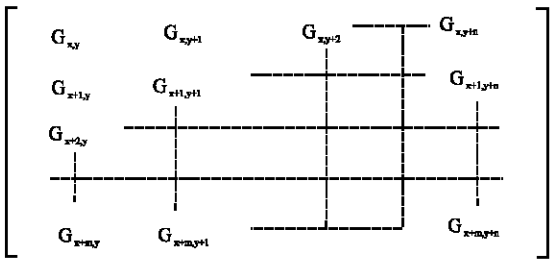
In this study a selected optimization method is based on a genetic algorithm. Genetic Algorithms (GA) are adaptive heuristic search algorithm premised on the evolutionary ideal of natural selection and genetic. The basic concept of GA is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Darwin of survival of the fittest.

The GA works by creating many random solutions to the problem at hand. This solutions will be subjected to an imitation of the evolution of species. All these solution are coded as genetic chromosomes and it will be made to mate by hybridization, also throwing in the occasional spontaneous mutation. The offspring generated will include some solutions that are better than the original.

The proposed approach represented the solution inside the block in Fig. 1, i represents the number of the original input signals and o are the desired outputs signals. The detail solutions inside the block are encoded into a string of chromosomes and subjected to the process of evolutionary algorithms.

The architecture of the proposed system is shown in Fig. 2. We use an encoding system 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 bits. A group of bits string used to represent 10 types of possible logic gates, Wire 1, Wire 2, XOR, XNOR, NAND, NOR, AND and OR, which 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 which x is represented the gate in row and y is represented the gate in the column. The m inputs is connected to y column 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.



(1)

Table 1: Logic gates representation

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

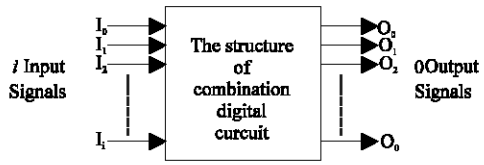


Fig. 1: Block representation of combination digital circuit structure

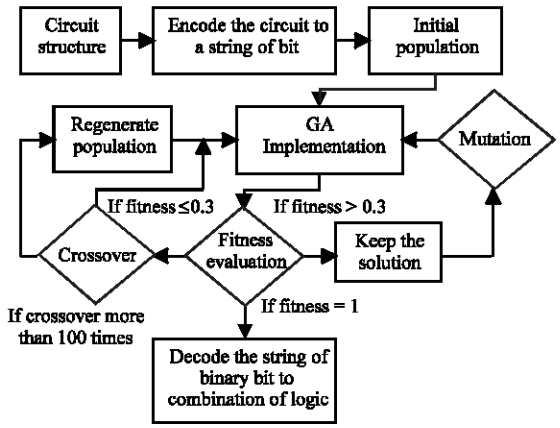


Fig. 2: Block diagram of the proposed system

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 is depend 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 is as follows:

```
begin
    create a random population of N designs;
    evaluate the fitness of each design in the population;
    while (there is no design with fitness = 1.0) do
        conduct crossover
        replace the old generation by the new generation of designs
        re-evaluate fitness of the designs in the population
        if (any design has fitness more than 30%) then
            implement the mutation
        if (any design has crossover more that 100 times) then
            replace the population with new random designs
        end while;
        generate the design representation with fitness = 1.0
    end
```

The design has fitness of 1 if it output value satisfies the constraint specification. Otherwise the structure design is deviated from the performance specification. The structure design modification is done by crossover and mutation operator. Crossover is the technique of exploring new regions of the search space to produce design other than the random population which generated in the initial stage. Crossover is a randomized approach to exchange information between two chromosomes. The new population is produced by crossover. The process of crossover can creates new structures from the existing structures. Crossover operation creates a variation in the population by producing offerings that combine characteristics from the two parents but are different from either of the parents. After the crossover operation, when the new chromosome achieved fitness percentage of 30%, the particular individual from the present generation moves to the mutation process.

Mutation is an approach to prevent the solutions in the population falling into a local optimum of the solved problem. Mutation operation randomly changes the offspring resulted from crossover.

The fitness of the process is the correctness of the obtained logic circuit in matching the truth table of the required function. In this study, we proposed a fitness functional called constraint fitness. The constraint fitness is based on the comparison of circuit output with the constraint output. The formula for Constraint Fitness (CF) is:

$$F_C = \frac{1}{x} \tag{2}$$

$$x = 1 + \sum_{i=1,2,3,....2^m}^{m} \left| c_o - f_o \right| \tag{3}$$

F_C is achieved 1 when x is equal to 1. c_o is constraint output set by the user and f_o is functional output generated from the circuit. When the functional output is exactly equal to the constraint output, this will cause the magnitude of the summation $|c_o - f_o|$ equal to 0. Consequently, F_C will equal to 1 due to x is 1. It is possible to get more that one constraint fitness in a random generated population. In order to determine the global optimum of the design, we accumulated all the population with the constraint fitness equal to 1 for further evaluation.

In this research, we are targeting the optimization of number of gates used in the structure design. After obtained the constraint fitness, 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 base on the chromosome with the constraint fitness of 1. Each string of chromosome is evaluated for the gate optimization fitness. The algorithm for this fitness is:

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

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

The numerator of [5] is the summation of the gates which can produces the constraint fitness equal to 1. The denominator part is the summation of the gate per structure. The fitness of GOF should in between 0 and 1. The global optimal is achieved when met the smaller fitness of GOF.

IMPLEMENTATION AND RESULTS

The proposed system was written in MATLAB language. 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 K-Map method.

For all the experiments, population size N = 1000, percentage of crossover reproduction C = 60, percentage of mutation reproduction M = 40. Combination logic gates boolean function of the designs are illustrated in the different table. These results show that the proposed system is able to produce quite effective solution of constraint-directed logic synthesis.

Example 1: The minterm for the output of the first example is as following:

$$F(a,b,c,d) = \Sigma(2,3,5,6,8,9,12,15)$$

Table 2 shows the first example result. Convergence to the optimum was achieved in generation 69. If this circuit developed by human, the proposed approach produces less number of gate counts by 80% compare with the other method as shows in Table 2.

Figure 3 shows the digital circuit schematic for Example 1 which was designed by the proposed method. The design just needs 1 AND and 2 XOR gates to generate the required output. Meanwhile, the solution for Example 1 that designed by convention K-Map method is illustrated in Fig. 4. The circuit needs 4 NOT gates, 6 AND gates and 5 OR gates to produce the necessary output. Obviously, the circuit designed by the proposed method needs lesser gate if compared to the conventional method.

Table 2: Comparison result between the proposed method and human design method of Example 1

Method	Logic function	No. of gates
Proposed method	F = (((bd) ⊕ a) ⊕ c)	3
Human design	F = abc + abc +acd+ acd + abcd + abcd	15

Table 3: Comparison result between the proposed method and human design method of Example 2

Method	Logic function	No. of gates
Proposed method	F = (((bd) ⊕ a) ⊕ c)	5
Human design	F = abc + abc +acd+ acd + abcd + abcd	7

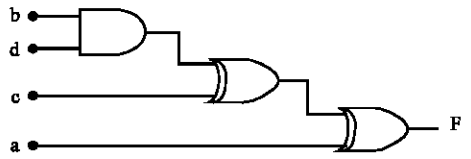


Fig 3: Circuit of Example 1 which was designed by the proposed method

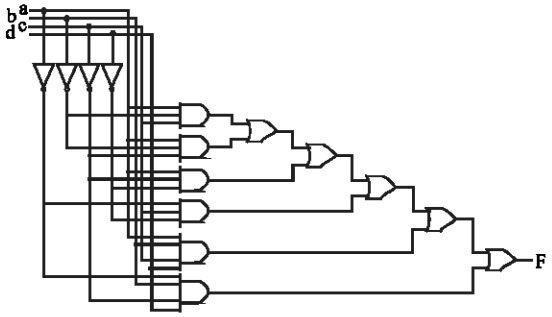


Fig 4: Circuit of Example 1 which was designed by K-Map

Example 2: The minterm of the second example is as following:

$$F(a,b,c,d) = \Sigma(7,10,11,13,14,15)$$

Table 3 shows the second example results. Convergence to the optimum was achieved in generation 175. The proposed approach produces less number of gate counts by 28.57% compare with the other method.

For example 2, the digital circuit schematics of both designs are shown in Fig. 5 and 6, respectively. The design done by proposed method just needs 3 AND and 2 OR gates to generate the required output. While, the design done by convention K-Map method needs 5 AND gates and 2 OR gates to produce the necessary output. Clearly, the circuit design implemented by the proposed method requires smaller number of gate if compared to the conventional design.

Table 4 shows the result of 5 Constraint Fitnesses (CF) for Example 1. Each CF achieved value of 1 in different population. CF 1 reached fitness at population of 60, CF 2 at population of 39, CF 3 at population of 31, CF 4 at population of 69 and CF 5 at population of 44.

Table 4: The constraint fitnesses for CF 1 to CF 5

Gen.	CF 1	CF 2	CF 3	CF 4	CF 5
1	0.1250	0.1111	0.1429	0.1111	0.0909
2	0.1111	0.0769	0.1111	0.0909	0.0909
3	0.1111	0.2000	0.1250	0.1111	0.1111
4	0.0909	0.1000	0.1111	0.1111	0.1111
5	0.1000	0.0909	0.1000	0.1250	0.1429
6	0.1429	0.1111	0.0833	0.1667	0.1000
7	0.2000	0.1111	0.0909	0.1111	0.1429
8	0.1000	0.0769	0.1667	0.0909	0.1111
9	0.1111	0.0769	0.1250	0.1111	0.1111
10	0.1111	0.2500	0.0833	0.1667	0.0909
11	0.1111	0.0909	0.0909	0.1000	0.1429
12	0.1111	0.1111	0.0909	0.1111	0.1111
13	0.0909	0.1111	0.0833	0.0909	0.2000
14	0.1429	0.1250	0.1111	0.1111	0.1111
15	0.1111	0.1111	0.0769	0.1111	0.1111
16	0.1111	0.0769	0.1111	0.2000	0.2000
17	0.1000	0.1111	0.1111	0.0909	0.0769
18	0.1111	0.0714	0.1111	0.1111	0.1250
19	0.1111	0.1250	0.1000	0.1111	0.1111
20	0.1111	0.0769	0.1111	0.1111	0.1111
21	0.0769	0.0769	0.1111	0.1111	0.1429
22	0.0909	0.1250	0.1429	0.1111	0.0833
23	0.1111	0.1111	0.1250	0.0909	0.1000
24	0.0909	0.0909	0.1000	0.1000	0.2000
25	0.0667	0.1111	0.1250	0.2000	0.0909
26	0.1429	0.1667	0.1000	0.1250	0.0909
27	0.1111	0.1111	0.1111	0.1429	0.1000
28	0.1111	0.1000	0.1250	0.1111	0.1000
29	0.1000	0.1111	0.1111	0.1111	0.0909
30	0.1429	0.3333	0.0833	0.1111	0.1111
31	0.1429	0.1429	1.0000	0.0909	0.0909
32	0.1000	0.0909	1.0000	0.1111	0.1250
33	0.1000	0.1429	1.0000	0.1111	0.0833
34	0.2000	0.0909	1.0000	0.1250	0.1111
35	0.1111	0.1250	1.0000	0.1111	0.1111
36	0.0909	0.1111	1.0000	0.1111	0.1000
37	0.1111	0.0909	1.0000	0.0909	0.1000
38	0.1111	0.1111	1.0000	0.0769	0.1111
39	0.1000	1.0000	1.0000	0.1429	0.1111
40	0.1111	1.0000	1.0000	0.1111	0.1111
41	0.1429	1.0000	1.0000	0.1111	0.0714
42	0.0909	1.0000	1.0000	0.1000	0.0909
43	0.0909	1.0000	1.0000	0.1111	0.1667
44	0.1429	1.0000	1.0000	0.1111	1.0000
45	0.1111	1.0000	1.0000	0.1429	1.0000
46	0.0909	1.0000	1.0000	0.2000	1.0000
47	0.1000	1.0000	1.0000	0.1111	1.0000
48	0.0833	1.0000	1.0000	0.1111	1.0000
49	0.1429	1.0000	1.0000	0.1111	1.0000
50	0.0909	1.0000	1.0000	0.1429	1.0000
51	0.0714	1.0000	1.0000	0.2000	1.0000
52	0.1111	1.0000	1.0000	0.1429	1.0000
53	0.1429	1.0000	1.0000	0.1111	1.0000
54	0.1000	1.0000	1.0000	0.1000	1.0000
55	0.1111	1.0000	1.0000	0.1250	1.0000
56	0.1111	1.0000	1.0000	0.1429	1.0000
57	0.1111	1.0000	1.0000	0.1250	1.0000
58	0.1111	1.0000	1.0000	0.0909	1.0000
59	0.1111	1.0000	1.0000	0.1111	1.0000
60	1.0000	1.0000	1.0000	0.0909	1.0000
61	1.0000	1.0000	1.0000	0.1111	1.0000
62	1.0000	1.0000	1.0000	0.1111	1.0000
63	1.0000	1.0000	1.0000	0.0909	1.0000
64	1.0000	1.0000	1.0000	0.1111	1.0000
65	1.0000	1.0000	1.0000	0.1111	1.0000
66	1.0000	1.0000	1.0000	0.1429	1.0000
67	1.0000	1.0000	1.0000	0.1111	1.0000
68	1.0000	1.0000	1.0000	0.0769	1.0000
69	1.0000	1.0000	1.0000	1.0000	1.0000

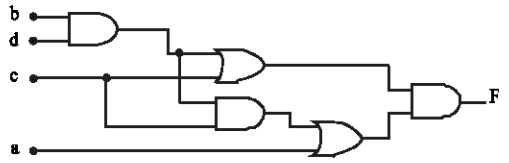


Fig 5: Circuit of Example 2 which was designed by proposed method

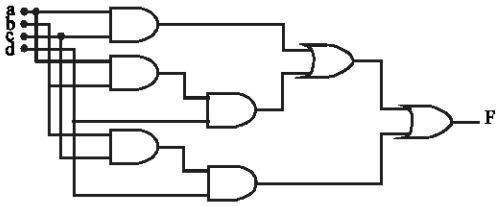


Fig 6: Circuit of Example 2 which was designed by K-Map

Table 5: Gate optimization fitness for Example 1 and Example 2

No.	Example 1	Example 2
1	0.8	0.5
2	0.7	0.6
3	0.8	0.5
4	0.9	0.3
5	0.7	0.4
6	0.8	0.6
7	0.5	0.5
8	0.6	0.7
9	0.8	0.5
10	0.7	0.4

The Gate Optimization Fitness (GOF) for Example 1 and Example 2 is shown in Table 5. The lowest GOF for Example 1 is 0.6 whereas 0.3 for Example 2.

DISCUSSION

The solutions 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:

- The proposed method can generate the circuit output function in nonstandard form.
- The proposed method can optimize the circuit output function with XOR and XNOR function.
- The proposed method excludes the using of NOT gate in the design.

The significant findings of the research are as follows:

- The proposed method uses decimal number instead of binary number for the chromosome representation. The length of the chromosome representation is only 10-bit. Therefore, the process duration for the crossover and mutation operation can be reduced.

- The proposed method can archive the functional and optimal solution within a small population of chromosome as well as the number of generation.

The obtained results for the digital circuit structure are based on the gate count instead of transistor count, area, delay and power consumption. Therefore, the results are bench marking with the conventional design method. The obtained result shows that the proposed method manages to produce the digital circuit structure with lesser gate count.

This research is limited to the small scale digital structure design. The implemented result shows encouraging for the 4-bit digital circuit design. This is a quite complex task to work on both functional and optimization of the digital circuit structure. The on going research is focusing on higher bit digital circuit design. The authors try to implement the variable length of chromosome representation in the design in order to archive the optimal solution in the shorter duration.

CONCLUSIONS

The obtained result shown that Evolutionary Algorithm (EA) can be applied into synthesis of logic circuits design. Although these first results are encouraging, more attempts are needed before EA can be used for performance constrained of large scale digital logic circuits design.

REFERENCES

Goldberg, D.E., 1989. Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, Cambridge, Massachusetts.

Louis, S.J., 1993. Genetic Algorithms as a Computational Tool for Design. Ph.D Thesis, Department of Computer Science, Indiana University.

Mano, M.M., 2002. Digital Design, Prentice Hall.

Nilagupta, P. and N. Ou-Thong, 2003. Logic Function Minimization base on Transistor Count using Genetic Algorithm. ICEP 2003 Genetic Algorithm, Transistor minimization, Digital Circuit Design, Automatic Design, 012546.

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

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