Optimising digital combinational circuit using particle swarm optimisation technique

Ushie, James Ogri, Obu Joseph Abebe Etim, Iniobong prosper
Department of Physics, Department of Physics, Department of Physics, University of Calabar, University of Calabar, University of Calabar, Calabar.

E-mail: ushjames@yahoo.com, abebeobu@yahoo.com, ini2etim@yahoo.com.

(Received 28 November 2011, accepted 27 February 2012)

Abstract

Human methods of circuit minimisation are tedious and limited to systems with four or five numbers of inputs. In order to save time and labour involved in designing digital combinational logic circuit, a standard algorithm that is suitable for digital combinational logic circuit with little modification which handle circuit with more than five inputs variables is developed. Employing MATLAB, the circuits were coded into particles using Particle Swarm Optimisation (PSO) techniques. This was then used to optimise a full-adder circuit. The result obtained, after optimisation for full-adder circuit using PSO technique is shown to have a minimum number of gates (five gates) compared to human designer method which has six gates.

Keywords: Digital combinational logic circuit, Human designer method, MATLAB, Particle Swarm Optimisation.

Resumen

Los métodos humanos de minimización de circuitos son tediosos y se limita a los sistemas con cuatro o cinco números de entrada. Con el fin de ahorrar tiempo y mano de obra involucrada en el diseño de circuitos digitales de lógica combinatoria, se desarrollado un algoritmo estándar que es adecuado para el circuito digital de lógica combinatoria con muy pocas modificaciones que se encarga del circuito con más de cinco variables de entrada. Con el empleo de MATLAB, los circuitos fueron codificados en partículas usando técnicas de optimización por enjambre de partículas (PSO). Este se utilizó entonces para optimizar un circuito sumador completo. El resultado obtenido, después de la optimización de circuito sumador completo utilizando la técnica de PSO se demuestra que tienen un número mínimo de puertas (cinco puertas) en comparación con el método de diseño humano que tiene seis puertas.

Palabras clave: Circuito digital de lógica combinatoria, método de diseño humano, MATLAB, Optimización por enjambre de partículas.

PACS: 07.50.Ek, 07.05.Tp, 07.05.Mh **ISSN 1870-9095**

I. INTRODUCTION

In digital circuit, minimisation is required to reduce the component count and size in a circuit, thereby reducing cost, physical size and weight, and hence increase system reliability and lowers power consumption, which is a prime requirement in modern circuit. There are several methods of circuit minimisation, examples, human methods (Boolean algebra, Karnaugh Map, Quine' McCluskey, etc.) and computational intelligence method such as Genetic Algorithm [1] Fuzzy Logic, Artificial Neural Network (ANN) and Particle Swarm Optimisation (PSO), [2]. The computational intelligence method has a significant advantage over the human methods because it has the ability being automated through programming.

The process of minimisation can be viewed as an optimisation process in that they both seek the best solution for a physical model [3]. In other words, it is a technique used for improving or increasing the value of a model. Examples of classical methods of optimisation include the gradient method, steepest descent and simplex method. They are useful in finding the optimum of continuous and differentiable function. These techniques, however, have limited scope in practical applications [4], since most day-to-day practical problems involve objective functions that are not continuous and differentiable. The limitation of the classical methods of optimisation has necessitated the development of modern optimisation methods.

Here, we have developed a code (see appendix) using PSO techniques for digital minimisation (written in MATLAB) and then used it to optimise a full-adder circuit.

II. THEORY OF PSO

Particle Swarm Optimisation (PSO) is a population-based stochastic optimisation technique developed by Eberhart and Kennedy [5] following inspiration got from the social

behaviour of a flock of birds or school of fish [6]. In PSO, population of potential solutions called particles "are flown" in search of the required solution, and each particle is updated in the process. The search and update process resembles the social interaction of the swarm of birds or a school of fish as they seek a common objective in a multi-dimensional search space. Each particle in the swarm keeps a record of the best "position" it has attained in the search space with respect to the objective function called the personal best (pbest), while the swarm keeps record of the overall best "position" attained by any particles, called global best (gbest). Each particle profit from the discoveries and it previous experience of the other particle during the exploration and search process, as they seek to achieve higher objective function values.

PSO differ from traditional optimisation method in that population of potential solution is used in the search, direct fitness information is used instead of function derivatives, and relative knowledge is used to guide the search, [7].

III. ALGORITHM FOR EVOLVING COMBINATIONAL CIRCUIT USING PSO

The PSO algorithm used for evolution and minimisation of digital combinational logic circuits was first implemented by Venus and Ganesh [2]. It runs as follows:

i. Initialise a population of particles with random "position" and "velocity" in n-dimensional of the problem space i
ii. Evaluate the fitness of each particle in the swarm to obtained pbest.
iii. Compare each particle's fitness with its previous best fitness obtained. If the current value is better than pbest, then set pbest equal the current value and pbest location equal to the current location in n-dimensional space.
iv. Compare pbest of particle with each other and update the swarm gbest location with the greatest fitness.
v. Change velocity and position of the particle according to Eqs. (1) and (2).
vi. Repeat step (ii) to (v) until convergence is reached based on some designed multiple criteria or it iteration limit expires.

The equation for updating particle's velocity and position are;

$$V_{IN} = W * V_{IN} + C_1 * rand_1 * (P_{IN} - X_{IN}) + C_2 * rand_2 * (P_{IN} - X_{IN}), \quad (1)$$

$$X_{IN} = X_{IN} + V_{IN}, \quad (2)$$

where V_{IN} and X_{IN} represent the velocity and position of the i^{th} particle with n-dimensions respectively, $rand_1$ and $rand_2$ are two random functions, W is inertial weight which controls the exploration and exploitation of the search space because it dynamically adjust velocity (from 0.4 to 0.9m/s), C_1 and C_2 are acceleration constants which change the velocity of a particle towards pbest and gbest.

IV. EVOLUTION OF A DIGITAL LOGIC CIRCUIT USING PSO

We used the particle swarm theory described above to evolve digital logic circuits by implementing the basic process of hardware evolution as illustrated in Fig. 1. The "desired" circuit refers to the circuit required to map 100% exactly the output for corresponding inputs typically given by the truth table for digital circuits. After each generation, the fitness is evaluated against the desired function to be implemented, given by the truth table. If the output of the circuit is equal to the output of the truth table for the corresponding inputs, then the fitness is increased by one. This is carried out for all inputs listed in the truth table. This process is repeated until the fitness value of the gbest particle is equal to the number of the truth table outputs.

In order for the system to know the function of each gate the switch case selection of the MATLAB were used and after each case, wordings such as AND gate, OR gates etc were used and each switch case represent a gate, the basic gate used in this study is comprised of AND, OR, NOT, XOR and a wire. A wire means no gate.

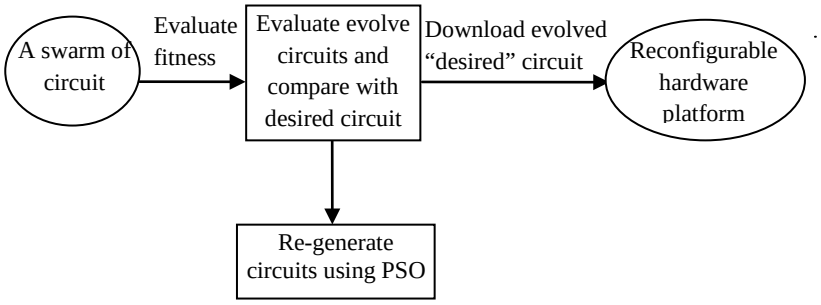

FIGURE 1. "Desired" circuit hardware evolution.

The matrix shown in Fig. 2 represents a circuit with M rows and N columns. The elements of the circuit are the logic gates which are selected from a predefined library of 1 or 2-input and 1-output gates. The inputs to the first column of the matrix come from the truth table of the function to be implemented. For all other columns, the input may come from any of the previous column outputs.

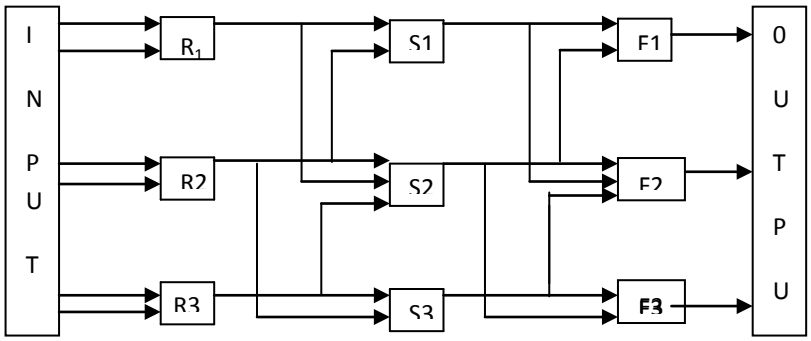

FIGURE 2. Structure of random matrix (inputs to each gate are obtained from gates in the previous columns), Venus and Ganesh [2].

A. Coding an input circuit

The gate selection for the circuit is done at random according to Eqs. (1) and (2). After each generation the expression is evaluated, approximated and compared to Fig. 2 to know the gate or input selected. A MATLAB program was coded and simulated for the implementation of the PSO algorithm. The MATLAB program is then applied to modify the matrix of each particle. This process is repeated until the gbest particle is equal to the number of the truth table outputs. The program samples 3 inputs variable to a circuit and gates from; AND, OR, NOT, XOR and WIRE to evolve circuit of desired interest.

For circuit evolution with PSO one matrix is used to represent gates/inputs interconnectivity. The size of the matrix in this case is 7 by 3. Elements in first and third column represent the inputs while the elements in the second column represent the gates. As illustrated in Fig. 3, gate is represented as: AND=1, OR=2, XOR=3, NOT=4 and WIRE=5.

The inputs are as well represented for convenience as follows;

A=1 ~A=2 B=3 ~B=4 C=5 ~C=6 R_1=7 R_2=8 R_3=9 S_1=10 S_2=11 S_3=12

F_1=F_2=F_3=F_{OUT} third column output, R_1, R_2 & R_3) first column gate output.

(S_1, S_2 & S_3) second column gate output

	AND 1	OR 2	XOR 3	NOT 4	WIRE 5	
1(A)						
2(~A)						
3(B)						
4(~B)						
5(C)						
6(~C)						
7(R_1)						
8(R_2)						
9(R_3)						
10(S_1)						
11(S_2)						
12(S_3)						
13(F)						

FIGURE 3. Gate/input interconnectivity representation.

This map illustrates the relationship between the coding of the numbering of the elements in the matrix and its actual interpretation in digital circuit as explain bellow. For example, considering the circuit of matrix as presented x_1 below. Individual elements of the matrix can be explained as follow:

X (1, 1) = 1 indicates that the input at this point is A=1, X (1, 2) =1 indicates an AND gate.
X (1, 3) = 3 in the third column shows that the second input to the AND gate is B = 1.
X (2, 1) = 3 indicates that the input at this point is B=1, X (2, 2) =2 indicates an OR gate.
X (2, 3) = 5 indicates that the second input to the OR gate is C = 1.

When the input or gate in the matrix indicates 0, it implies NO input or NO gate as in X (3, 1), X (3, 2), X (3, 3), X (5, 1), X (5, 2), X (5, 3), X (6, 1), X (6, 2), X (6, 3) and X (7, 3). X (4, 1) =7 indicates that the input at this point is R_1 (R_1 output of first column gate as indicated in Fig. 1), X (4, 2) =3 indicates an XOR gate.

X (4, 3) = 8 indicates that the input at this point is R_2 (R_2 output of first column gate as indicated in Fig. 2).
X (7, 1) =10 indicates that the input at this point is S_1 (S_1 output of first column gate as indicated in Fig. 1). X (7, 2) =4 indicates an NOT gate.

For implementation of the full adder circuit, individually initialised circuit were presented in matrix form as given in Eqs. (3 – 7) below:

$$X_1 = \begin{bmatrix} 1 & 1 & 3 \\ 3 & 2 & 5 \\ 0 & 0 & 0 \\ 7 & 3 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{bmatrix} \quad (3) \qquad X_2 = \begin{bmatrix} 1 & 1 & 5 \\ 3 & 1 & 5 \\ 0 & 0 & 0 \\ 7 & 2 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{bmatrix} \quad (4)$$

$$X_3 = \begin{bmatrix} 1 & 3 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 7 & 1 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{bmatrix} \quad (5) \qquad X_4 = \begin{bmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 0 & 0 & 0 \\ 7 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 7 & 3 & 11 \end{bmatrix} \quad (6)$$

$$X_5 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 5 \\ 3 & 3 & 3 \\ 7 & 3 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 9 & 3 & 10 \end{bmatrix} \quad (7)$$

The full-adder truth table generally used for both human design method and the PSO method is shown Table I. However, the full-adder circuit obtained by human design methods have three inputs and two outputs as are shown in Fig. 4.

TABLE I. Full adder truth table.

S/N	A B C_{IN}	S	C_{OUT}
1	0 0 0	0	0
2	0 0 1	1	0
3	0 1 0	1	0
4	0 1 1	1	1
5	1 0 0	1	0
6	1 0 1	1	1
7	1 1 0	1	1
8	1 1 1	1	1

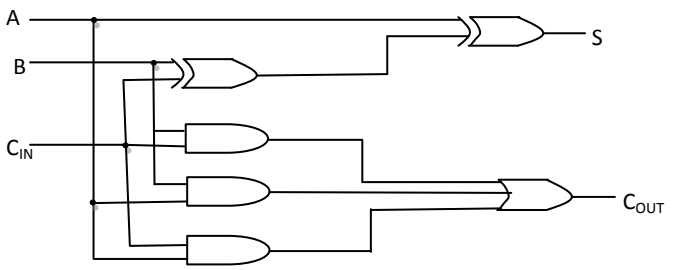

FIGURE 4. Full-adder circuit by human design method.

V. RESULTS

The summary of sum result for 1st, 50th, 100th, and 101st, iterations for the gbest matrix [Eq. (8), Eqs. (9-11), Eqs. (12-14), and Eq. (15), respectively], and their corresponding circuit (Figs. 5 - 7) are as presented below. Notice that the circuit for 1st and 50th iterations are the same since they have the same number of fitness.

Summary of sum result for 1st iteration

$Fout_2$ = [1 1 1 0 1 0 1 0]
best fitness = [4 5 4 3 4]
maximum fitness = [4 5 4 3 4]

$$X_2 = \begin{bmatrix} 1 & 1 & 5 \\ 3 & 1 & 5 \\ 0 & 0 & 0 \\ 7 & 2 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{bmatrix}, \quad (8)$$

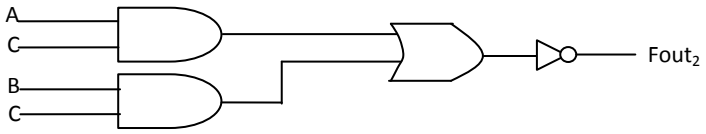

FIGURE 5. gbest of initial sum circuit used.

Summary of sum result for 50th iteration

$Fout_1$ = [1 1 1 0 1 1 0 0]
$Fout_2$ = [1 1 1 0 1 1 0 0]
$Fout_3$ = [1 1 1 0 0 0 0 0]
$Fout_4$ = [1 1 1 0 1 1 0 0]
$Fout_5$ = [1 1 1 0 1 1 0 0]
best fitness = [5 5 5 5 5]
maximum fitness = [5 5 5 5 5]

$$X_1 = \begin{bmatrix} 1 & 1 & 3 \\ 3 & 1 & 5 \\ 0 & 0 & 0 \\ 7 & 2 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{bmatrix} \quad (9) \qquad X_2 = \begin{bmatrix} 1 & 1 & 4 \\ 3 & 1 & 5 \\ 0 & 0 & 0 \\ 7 & 2 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{bmatrix} \quad (10)$$

$$X_3 = \begin{bmatrix} 1 & 1 & 6 \\ 3 & 1 & 5 \\ 0 & 0 & 0 \\ 7 & 2 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{bmatrix}. \quad (11)$$

Summary of sum result for 100th iteration

$Fout_1$ = [1 0 1 0 1 0 0 1]
Fout3 = [1 0 1 0 1 0 0 1]
$Fout_5$ = [1 1 1 1 1 0 0 1]
best fitness = [6 6 6 6 6]
maximum fitness = [6 4 6 2 6]

$$X_1 = \begin{bmatrix} 1 & 1 & 5 \\ 2 & 2 & 4 \\ 1 & 2 & 2 \\ 4 & 1 & 4 \\ 4 & 1 & 4 \\ 0 & 0 & 0 \\ 8 & 3 & 5 \end{bmatrix} \quad (12) \qquad X_3 = \begin{bmatrix} 1 & 1 & 5 \\ 2 & 2 & 4 \\ 1 & 2 & 2 \\ 4 & 1 & 4 \\ 4 & 1 & 4 \\ 0 & 0 & 0 \\ 8 & 3 & 5 \end{bmatrix} \quad (13)$$

$$X_5 = \begin{bmatrix} 1 & 2 & 5 \\ 1 & 1 & 5 \\ 1 & 3 & 3 \\ 6 & 1 & 4 \\ 4 & 0 & 4 \\ 0 & 0 & 0 \\ 9 & 3 & 5 \end{bmatrix}. \quad (14)$$

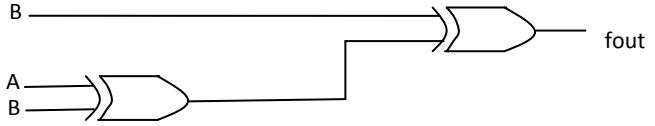

FIGURE 6. Sum gbest Circuit for 100th Iteration.

Summary of sum result for 101st iteration

Fout$_5$ = [0 1 1 0 1 0 0 1]
best fitness = [6 6 6 6 8]
maximum fitness = [6 6 6 2 8]

$$X_5 = \begin{bmatrix} 1 & 2 & 5 \\ 1 & 1 & 5 \\ 1 & 3 & 3 \\ 6 & 1 & 4 \\ 4 & 0 & 4 \\ 0 & 0 & 0 \\ 9 & 3 & 5 \end{bmatrix}. \quad (15)$$

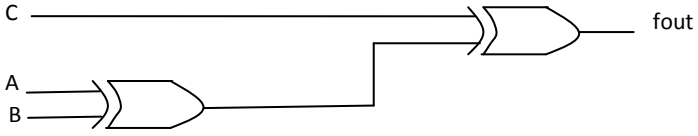

FIGURE 7. Sum gbest Circuit for 101ST Iteration.

The summary of carry result for 1st, 100th, and 358th, iterations for the gbest matrix [Eq. (16), Eqs. (17-21), and Eq. (22), respectively], and their corresponding circuit (Figs. 8 - 9) are as presented below.

Summary of carry result for 1st iteration

Fout$_4$ = [0 0 0 1 1 1 1 1]
best fitness = [4 1 4 7 4]
maximum fitness = [4 1 4 7 4]

$$X_4 = \begin{bmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{bmatrix}. \quad (16)$$

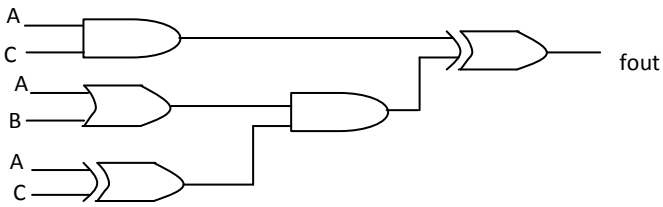

FIGURE 8. Carry gbest for the Initial Circuit

Summary of carry result for 100th iteration

Fout$_1$ = [0 0 0 1 1 1 1 1], Fout2 = [0 0 0 1 1 1 1 1]
Fout$_4$ = [0 0 0 1 1 1 1 1], Fout$_5$ = [0 0 0 1 1 1 1 1]
best fitness = [7 7 6 7 7]
maximum fitness = [7 7 6 7 7]

$$X_1 = \begin{bmatrix} 1 & 1 & 5 \\ 2 & 2 & 3 \\ 1 & 4 & 6 \\ -2 & -1 & -2 \\ 10 & 1 & 2 \\ 0 & 0 & 0 \\ 6 & 3 & 1 \end{bmatrix} \quad (17) \qquad X_2 = \begin{bmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{bmatrix} \quad (18)$$

$$X_3 = \begin{bmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{bmatrix} \quad (19) \qquad X_4 = \begin{bmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{bmatrix} \quad (20)$$

$$X_5 = \begin{bmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{bmatrix}. \quad (21)$$

Summary of carry result for 358th iteration

Fout$_1$ = [0 0 0 1 0 1 1 1]
best fitness = [7 7 8 7 7]

maximum fitness = [7 7 8 7 7]

$$X_3 = \begin{bmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 6 \\ 3 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{bmatrix}. \quad (22)$$

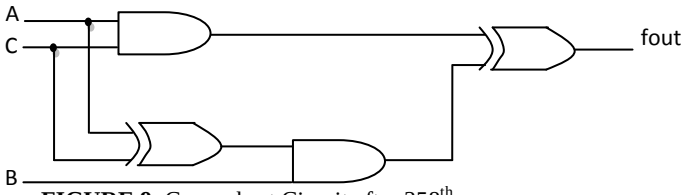

FIGURE 9. Carry gbest Circuit after 358th

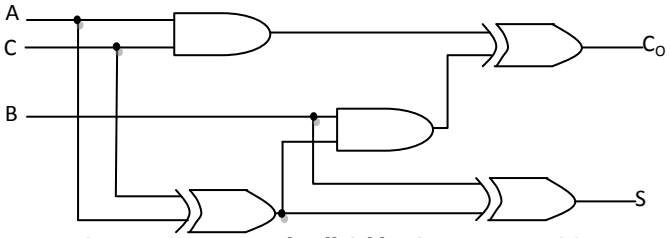

FIGURE 10. Minimised Full Adder Circuit using PSO.

VI. SUMMARY AND CONCLUSION

We have minimised a full-adder circuit using PSO method, from six gates (2 XOR, 3 AND and 1 OR gates) obtained from human-designer method to five gates (3 XOR, and 2 AND gates). The five components designed, evolved circuit using PSO satisfies the "desired" circuit in this case that is expected to have a fitness of eight. From our work, we determined that the gbest of the carry circuit evolved for the 358 generations and the gbest of the sum evolved for the 101 generations.

After the simulation of PSO-designed circuit on an electronic work bench, it was seen that the PSO approach is an improvement over the human designer method because it has minimum number of gates as summarised in the truth table illustrated in Table II. The result presented in the truth table shows that the output of the simulated circuit are the same with that of full-adder truth table.

TABLE II. Simulated outputs for corresponding input.

S/N	A B C	SUM	CARRY
1	0 0 0	OFF	OFF
2	0 0 1	ON	OFF
3	0 1 0	ON	OFF
4	0 1 1	OFF	ON
5	1 0 0	ON	OFF
6	1 0 1	OFF	ON
7	1 1 0	OFF	ON
8	1 1 1	ON	ON

TABLE III. Comparing PSO and human designer for full adder circuit.

HD2	PSO
6 GATES	5 GATES
2 XOR, 3 AND and 1 OR gates	3 XOR, and 2 AND gates

REFRENCES

[1] Sulshil, J. L., *Genetic Learning for Combinational Logic Design* (2003), sulhi@csunr.edu/htt.//www.cs.unr.edu/~sushil 05/08/2006.
[2] Venus, G. G. and Ganesh, K. V., *Evolving Digital Circuit Using Particle Swarm* (2003), http://www.ieee+plore.ieee.org 12/12/2005.
[3] Beale, E. M., *Introduction to Optimisation*, (John Wiley Sons, New York, 1988), pp. 1-2.
[4] Rao, S. S., *Optimisation: Theory and Application*, (Wiley, Delhi, 1978), pp. 1, 284-289, 193 and 298.
[5] Kennedy, J. and Eberhart, R. C., *Particle Swarm Optimisation* (1995),
[6] Hu, X., *Particle Swarm Optimisation Tutorial* (2002), www.ncbi.nlm.nih.gov 11/05/2005.
[7] Paquet, U. and Engelbrecht, A. P., *Training Vector Machine with Particle Swarm* (2003), http://www.ieee+plore.ieee.org 15/12/2006.