Optimal Design of Full Adder Circuit using Particle Swarm Optimization Algorithm

R. Das[1], A. Kumar[1], S. Kumar[1], P. K. Prasad[1], R. Kar[1], D. Mondal[1], S.P. Ghoshal[2]
[1]Department of Electronics and Communication Engineering, NIT Durgapur, India-713209
[2]Department of Electrical Engineering, NIT Durgapur, India-713209
[1]rupjyotidas90@gmail.com

***Abstract*—With increasing complexity of electronic circuits, the design and optimization of electronic circuit needs to be automated with high degree of reliability and accuracy. In order to optimize hardware requirement of digital combinational circuits, evolutionary and innovative techniques need to be enforced at various levels such as gate level and device level. This paper presents the use of one of the evolutionary techniques, i.e., Particle Swarm Optimization (PSO) algorithm. It is motivated by the social behaviour of organisms for the optimal design of combinational logic circuit with a reduced gate count in MATLAB platform. Results are presented to confirm that the PSO based algorithm is superior to Human Design Method in terms of time, effort and especially the gate count required to design the digital combinational circuits. The paper shows that PSO based algorithm converges faster than other algorithms such as genetic algorithm and also reduces the computational complexity.**

Keywords—Combinational Circuit; Optimization; Human Design Method; Particle Swarm Optimization.

I. Introduction

Different methods available for designing the combinational circuits can be broadly classified as Human Design and Automated Computational Intelligent methods. Some of the human design methods for circuit minimization are Karnaugh Map [1], Quine' McCluskey [2] and Sasao method [3]. Some of the automated computational intelligent methods are based on Genetic Algorithm, Artificial Neural Network (ANN), Fuzzy Logic and Particle Swarm Optimization (PSO) [4]. In combinational circuit design, optimization is required to reduce the gate count and thereby reducing the circuit area and the cost. This improves the power consumption and the system reliability, which are the basic requirements in today's portable circuits. One of the bottlenecks with human design is that it becomes cumbersome and problematic when number of inputs and outputs or complexity of the function is increased. One more advantage of computational method over human design is that it can be made automated using programming which results in faster and better optimization.

Recently, researchers have designed combinational circuit using Genetic Algorithm [9]. It uses number of chromosomes and their fitnesses are being updated over the generations and the chromosome having the best fitness finally survives.

Genetic Algorithm takes more time to converge, resulting in its limited real world applications. Recently a new technique called Particle Swarm Optimization (PSO) has emerged as a computation method that optimizes a problem by iteratively trying to enhance the particle solution in order to get the near global optimal results [5]. PSO finds its applications in optimization problems in science and engineering involving nonlinear objective function [6]. This paper proposes a method to optimally design the combinational circuit using PSO implemented in MATLAB platform.

II. Particle Swarm Optimization

R. Eberhart and J. Kennedy first proposed the concept of PSO technique [7] as an evolutionary computation technique in 1995. It is a random optimization technique based on the social behaviour of a flock of birds or a school of fish and the swarming theory [8]. In PSO, it starts with initializing a population of random solutions and searches for the optimum solution by updating those random solutions in each generation. The population of potential solutions is made up of particles which fly in search of the optimum solution in multi-dimensional search space. Each particle is associated with a particular position and velocity in the swarm. The velocity and position of each particle is updated in each iteration. Each particle updates its personal best position (p_best) that it has reached with updated position and velocity in the multi-dimensional search space with respect to the desired function and there is global best (g_best) which is the overall best position reached by any particle in the swarm towards its target in the multi-dimensional search space. In PSO, the velocity and the position of each particle are changed in each iteration towards its p_best and g_best locations. Velocity can be weighted by using separate random numbers for acceleration towards p_best and g_best locations.

PSO differs from Genetic Algorithm (GA) as PSO has no evolutionary operators such as crossover and mutation. GA deals with discrete values whereas PSO can deal with any values [9].

The approach of PSO algorithm used for evolving digital circuits was first reported in [5]. The steps of the PSO algorithm as implemented for the design of the digital combinational circuit are given below:

978-1-4673-9939-5/16/$31.00 ©2016 IEEE

Step 1. In multi-dimensional search space, initialize the group of particles with random positions and velocities.
Step 2. These initialized random positions are assigned as p_best of the corresponding particles.
Step 3. Fitness function of each particle is calculated for these initial random positions.
Step 4. p_best of the particles with the highest fitness is assigned as g_best.
Step 5. Update the velocity of each particle according to (1) and the position of each particle according to (2).
Step 6. Fitness of each particle is calculated for those updated positions.
Step 7. For each particle in swarm, compare the particle's current fitness with the fitness achieved so far with p_best position. If the current fitness of a particle is better, then set this position as the p_best.
Step 8. Identify the particle which has the best fitness value and assign the position of the particle as g_best.
Step 9. Repeat steps 5-8 until the terminating condition is reached or the maximum iterations are reached.

To update the velocity and the position of the particle, the following equations are used:

$$V_{I(N+1)} = W \times V_{IN} + C_1 \times rand_1 \times (p\_best_I - X_{IN}) + C_2 \times rand_2 \times (g\_best - X_{IN}) \quad (1)$$

$$X_{I(N+1)} = V_{I(N+1)} + X_I \quad (2)$$

where X_{IN} and V_{IN} are the position and the velocity of the i^{th} particle at the n^{th} iteration, respectively. $rand_1$ and $rand_2$ are two random numbers. W is known as the inertial weight which controls the velocity of the particle in the multi-dimensional search space. p_best_I is the i^{th} particle's best position. g_best is the best position achieved so far by all the particles. C_1 and C_2 are known as cognition and social components, respectively. They are the acceleration constants used to change the velocities of particles towards p_best and g_best.

III. Designing Of Digital Combinational Circuit Using PSO

PSO is used in this paper to design and optimize the digital circuit. The above described algorithm is illustrated in Figure 1 [10]. Initially, the particles are initialized randomly. After that the particles are evaluated in terms of circuit and those are compared with the desired circuit. Here the 'desired circuit' refers to the circuit which needs to be designed having exactly the same outputs for corresponding input combinations as given in the truth table for digital circuits. After each iteration, the fitnesses are evaluated for those circuits by comparing their outputs with the outputs of the desired circuit given by the truth table. The fitness is incremented by one whenever the outputs of each evolved circuit are matched with the corresponding outputs of the truth table. This process is repeated for all possible combinations of inputs. For example, in case of a Full Adder circuit there are three inputs. So, eight different combinations of input are possible. As a result, the process of updating the fitness is repeated for eight times for each particle. Then in the next iteration, the velocity of each particle is updated according to (1) and the position of the particle is updated as per (2). Then with the updated positions of the particles, fitnesses are evaluated. This process is repeated until the terminating condition is met. Terminating condition can be the maximum iteration or the maximum fitness achieved.

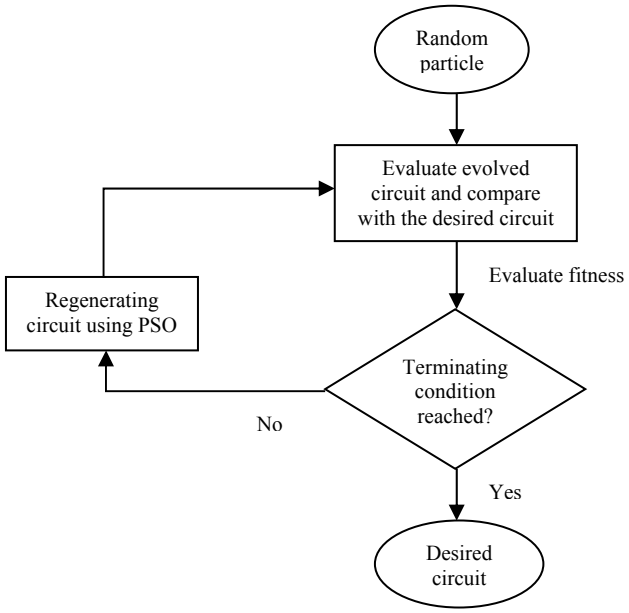

Fig. 1. Flow chart of circuit design.

A. Representation of Digital Combinational Circuit

The digital circuits that need to be optimized using PSO are represented in the form of grid as shown in Figure 2 [5], [11], [12].

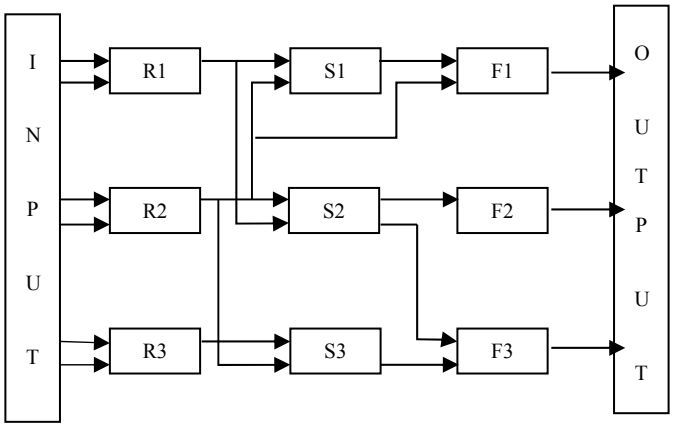

Fig. 2. Structure of the Grid.

The grid shown in Figure 2 has *m* rows and *n* columns. So, it contains *mn* number of elements. Each element in the grid is nothing but a basic gate having 2 inputs and 1 output. All these *mn* gates constitute the circuit. In the experiment of Full Adder, there are 3 inputs. So, a 3×3 grid is used. The inputs to

978-1-4673-9939-5/16/$31.00 ©2016 IEEE

the first column of the grid come from the primary inputs, i.e., the inputs of circuit to be implemented. In this design, they are A, B and C. The inputs to all other column elements may come from primary inputs or outputs of any other columns. For example R1, R2 and R3 denote the outputs of the gates in the first column, respectively, and S1, S2 and S3 denote the outputs of the gates in the second column, respectively. F1, F2 and F3 constitute one single element which represents the overall output of the circuit.

B. Encoding the circuit

A MATLAB code is written to implement PSO algorithm to optimize the digital circuit. In the MATLAB program the grid of Figure 2 is depicted in the form of a matrix. The order of the matrix depends on grid size. In this paper the 3x3 grid is represented with the help of a 7x3 matrix, each row of the matrix denotes a gate. First and third column elements denote its inputs and the second column elements denote the type of gates. For this purpose Table I and Table II [10] are used to describe those gates. Five different types of gates AND, OR, XOR, NOT and WIRE have been used. WIRE is a gate where output is equal to input. Switch case function is used to describe the operation of each gate. The grid is converted to matrix in a column-wise manner. The program starts scanning that grid from the first column. Then it goes through all the rows of column 1 from top to bottom. After that it starts scanning the next column in the same top to bottom manner. The process continues till the last row of the last column is reached. So, the 1st row of the matrix is for the first column first row element of the grid. The output of this row will be R1.

TABLE I. ENCODING OF 2ND COLUMN

Matrix value of 2nd column	Type of gate
1	AND
2	OR
3	XOR
4	NOT
5	WIRE

TABLE II. ENCODING OF 1ST AND 3RD COLUMNS

Matrix value of 1st column and 3rd column	Input
1	A
2	$\overline{A}$
3	B
4	$\overline{B}$
5	C
6	$\overline{C}$
7	R1
8	R2
9	R3
10	S1
11	S2
12	S3
13	F

Initially, p is the number of swarm particles (in this case p = 5), random circuits gates and input matrices are initialized. For this purpose matrix X1 to X5 have been used as shown in Figure 3 [10].

In the following way these matrices are encoded into the circuit. For example, consider X1 matrix. X1(1, 1) = 1 indicates that the first input to the first column-first row element of the grid is A. X1(1, 3) = 3 indicates that the second input to the first column-first row element of the grid is B. X1(1, 2) = 1 indicates that the first column-first row element of the grid is AND gate. So, R1 = A AND B.

$$X1 = \begin{pmatrix} 1 & 1 & 3 \\ 3 & 2 & 5 \\ 0 & 0 & 0 \\ 7 & 3 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{pmatrix} \quad X2 = \begin{pmatrix} 1 & 1 & 5 \\ 3 & 1 & 5 \\ 0 & 0 & 0 \\ 7 & 2 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{pmatrix} \quad X3 = \begin{pmatrix} 1 & 3 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 7 & 1 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{pmatrix}$$

$$X4 = \begin{pmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{pmatrix} \quad X5 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 5 \\ 3 & 3 & 3 \\ 7 & 3 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 9 & 3 & 10 \end{pmatrix}$$

Fig 3. Initial Random Matrix.

If an element in the matrix is 0, then there is neither an input nor a gate. For example, X1(3, 1) = 0 indicates there is no input and X1(3, 2) = 0 indicates there is no gate at the first column third row element in the grid.

In the next step, digital circuit is formed corresponding to each matrix. For each circuit all possible binary input combinations, i.e., primary inputs are applied and their corresponding output values (i.e., values of F) are obtained. Then those outputs are compared with the outputs of the desired circuit. If the output of the developed circuit matches exactly with the desired circuit, the fitness value for the corresponding circuit is incremented by 1. In this paper, Full Adder circuit is designed which is a three input combinational circuit having 2^3 (=8) possible input combinations. So, the prime objective is to get the fitness value equals to eight.

After calculating the fitnesses of all the circuits corresponding to all the matrices, position matrix having the highest fitness value of a particular particle till current iteration is recorded. That matrix becomes p_best matrix for that particle. The position matrix having the highest fitness value is recorded and that matrix is assigned as g_best matrix. In each iteration, the positions of the particles are updated according to (1) and velocities of the particles are updated according to (2). These processes are repeated until the terminating condition is reached which is maximum number of iterations or the desired fitness value.

The value of the parameters of PSO algorithm cannot be generalized. It depends on the applications, for which it is being implemented [13], [14]. The value of $rand_1$ and $rand_2$ used in (1) should be in between 0 and 1. The velocity has been initialized as zero in many papers. In this paper, the velocity has been initialized as 0.1. Due to updating particle

978-1-4673-9939-5/16/$31.00 ©2016 IEEE

velocity in each iteration, it may become too high or too low. So, it is required to set some upper and lower bounds of the velocity. For this purpose (3) and (4) [15] are used.

$$V_{min} = 0 \quad (3)$$

$$V_{max} = (X_{max} - X_{min}) \times 0.2 \quad (4)$$

The cognition and social components, C_1 and C_2 in (1) should be selected in such a way that sum of them is equal to 4. In this paper, $C_1 = C_2 = 2$ are taken. The value of inertia weight, W ranges from 0.4–0.9. But the optimum solution can be achieved with low processing time by using dynamic inertia weight through time varying process. The time varying techniques for inertia weight can be either linear or nonlinear and decreasing or increasing. Linearly decreasing time varying technique has been widely used for inertia weight. In this technique, inertia weight value is linearly decreased from its maximum value, W_{max} to its minimum value, W_{min}. Linearly decreasing inertia weight has been defined in (5) [16].

$$W = W_{min} + (W_{max} - W_{min}) \times (Iter_{max} - Iter) / Iter_{max} \quad (5)$$

where $Iter_{max}$ is the maximum number of iterations and $Iter$ is the current iteration.

Both human design method and the PSO based method use the following truth table to design the Full Adder circuit. The Full Adder circuit designed using human method from the truth table (using Karnaugh map) is shown in Figure 4.

TABLE III. TRUTH TABLE OF FULL ADDER

Inputs			Outputs	
A	*B*	*C*	*Sum*	*Carry*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

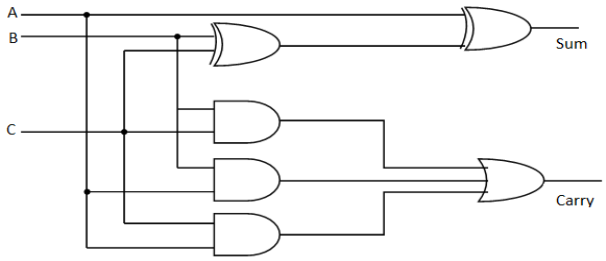

Fig 4. Full Adder circuit designed by using K-map (Human Design).

IV. Simulation Results And Discussions

A. Generating circuit for Sum

The matrices used are shown in Figure 3. After applying all the input combinations from Table III, the outputs for those circuits after 1st iteration are obtained as shown in Table IV.

The fitness of each circuit is calculated by comparing the outputs from Table IV with 'Sum' from Table III and is shown in Table V.

As seen from Table V the maximum fitness is achieved after 1st iteration is 5 which corresponds to matrix X2. So, after 1st iteration X2 is taken as g_best matrix as shown in Figure 5. As this is the 1st iteration the matrices X1 through X5 are taken as their corresponding p_best matrices. After 46th iteration, fitness value of 8 is obtained. After applying the input combinations from Table III, the output from those five circuits after 46th iteration are obtained as shown in Table VI.

The fitness of the circuits is calculated from the outputs of the same circuits and is shown in Table VII. As seen from Table VII the maximum fitness (=8) is achieved which corresponds to matrix X4. So, after 46th iteration X4 is taken as g_best matrix as given in Figure 6. The optimal circuit for 'Sum' corresponding to g_best matrix after 46th iteration is shown in Figure 7.

TABLE IV. OUTPUT OF RANDOM CIRCUIT AFTER 1ST ITERATION

Inputs			Outputs				
A	*B*	*C*	F_{X1}	F_{X2}	F_{X3}	F_{X4}	F_{X5}
0	0	0	1	1	1	0	0
0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	1
0	1	1	0	0	0	1	1
1	0	0	1	1	0	1	1
1	0	1	0	0	1	1	0
1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	0

TABLE V. FITNESS AFTER 1ST ITERATION

	X1	X2	X3	X4	X5
Fitness	4	5	4	3	4

$$g\_best = X2 = \begin{pmatrix} 1 & 1 & 5 \\ 3 & 1 & 5 \\ 0 & 0 & 0 \\ 7 & 2 & 8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 4 & 0 \end{pmatrix}$$

Fig 5. g_best matrix after 1st iteration.

TABLE VI. OUTPUT OF RANDOM CIRCUIT AFTER 46TH ITERATION

Inputs			Outputs				
A	*B*	*C*	F_{X1}	F_{X2}	F_{X3}	F_{X4}	F_{X5}
0	0	0	1	1	1	0	0
0	0	1	0	0	0	1	1
0	1	0	0	0	0	1	1
0	1	1	0	1	1	0	0
1	0	0	1	1	1	1	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	1
1	1	1	0	1	1	1	0

TABLE VII. FITNESS AFTER 46TH ITERATION

	X1	X2	X3	X4	X5
Fitness	4	4	4	8	6

$$g\_best = X4 = \begin{pmatrix} 1 & 2 & 4 \\ 1 & 3 & 5 \\ 1 & 0 & 1 \\ 3 & 1 & 3 \\ 5 & 1 & 4 \\ 0 & 0 & 0 \\ 3 & 3 & 8 \end{pmatrix}$$

Fig 6. g_best matrix after 46th iteration.

978-1-4673-9939-5/16/$31.00 ©2016 IEEE

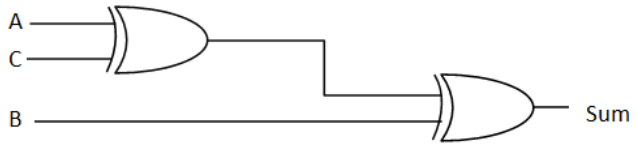

Fig 7. Sum circuit after 46th iteration.

B. Generate circuit for Carry

The same matrices as shown in Figure 3 have been used for generating the Carry circuit. After applying the input combinations from Table III, the outputs from those five circuits after 1st iteration are obtained and are shown in Table VIII. The fitness of each circuit is calculated by comparing the output from Table VIII with 'Carry' from Table III and is shown in Table IX. As seen from Table IX the maximum fitness achieved after 1st iteration is 7 which corresponds to matrix X4. So, after 1st iteration X4 is taken as g_best matrix as shown in Figure 8.

TABLE VIII. OUTPUTS OF RANDOM CIRCUITS AFTER 1ST ITERATION

Inputs			Outputs				
A	*B*	*C*	F_{X1}	F_{X2}	F_{X3}	F_{X4}	F_{X5}
0	0	0	1	1	1	0	0
0	0	1	0	1	1	0	0
0	1	0	0	1	1	0	1
0	1	1	0	0	0	1	1
1	0	0	1	1	0	1	1
1	0	1	0	0	1	1	0
1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	0

TABLE IX. FITNESS AFTER 1ST ITERATION

	X1	X2	X3	X4	X5
Fitness	4	1	4	7	4

$$\text{g\_best} = X4 = \begin{pmatrix} 1 & 1 & 5 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 8 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 3 & 11 \end{pmatrix}$$

Fig 8. g_best matrix after to 1st iteration.

As this is the 1st iteration the matrices X1 through X5 are taken as their corresponding p_best matrices. After 105th iteration fitness value of 8 is obtained. After applying the input combinations from Table III, the outputs for those five circuits after 105th iteration are obtained and are shown in Table X. The fitnesses of the circuits are calculated from the outputs of the same circuits and are shown in Table XI. As seen from Table XI the maximum fitness (=8) is achieved which corresponds to matrix X3. So, after 105th iteration X3 is taken as g_best matrix as shown in Figure 9. The optimal circuit for 'Carry' corresponding to g_best matrix after 105th iteration is shown in Figure 10.

TABLE X. OUTPUT OF RANDOM CIRCUIT AFTER 105TH ITERATION

Inputs			Outputs				
A	*B*	*C*	F_{X1}	F_{X2}	F_{X3}	F_{X4}	F_{X5}
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	0	1	1	0	0
1	0	0	0	1	0	0	0
1	0	1	1	1	1	1	1
1	1	0	0	1	1	0	0
1	1	1	1	1	1	1	1

TABLE XI. FITNESS AFTER 105TH ITERATION

	X1	X2	X3	X4	X5
Fitness	6	7	8	6	6

$$\text{g\_best} = X3 = \begin{pmatrix} 1 & 1 & 5 \\ 0 & 2 & 3 \\ 1 & 3 & 5 \\ 0 & 0 & 0 \\ 3 & 1 & 9 \\ 0 & 0 & 0 \\ 7 & 2 & 11 \end{pmatrix}$$

Fig 9. g_best matrix after 105th iteration.

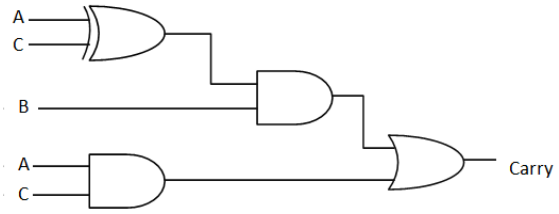

Fig 10. Carry circuit after 105th iteration.

C. Generate circuit for Full Adder

Combining the circuits obtained from Figure 7 and Figure 10, the optimized Full Adder circuit is realized and is shown in Figure 11.

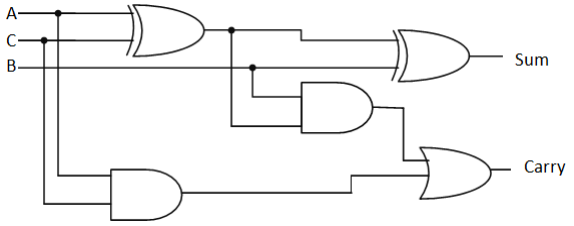

Fig 11. Optimized Full Adder circuit using PSO.

D. Comparison

The proposed design circuit has been compared with the human design circuit using Karnaugh map and with the circuit designed by using PSO [10].

Table XII justifies that the proposed PSO based Full Adder circuit design takes lesser number of iterations as compared with that of [10] and yields a lesser gate count as compared with that of human design.

978-1-4673-9939-5/16/$31.00 ©2016 IEEE

TABLE XII. COMPARISON OF DIFFERENT FULL ADDER CIRCUIT

		Human Design	PSO by [10]	PSO in this paper
Sum circuit	Gate count	2	2	2
	Iterations	-	101	46
Carry circuit	Gate count	4	4	4
	Iterations	-	358	105
Full Adder	Gate count	6	5	5
	Iterations	-	459	151

V. CONCLUSION

Comparing the proposed design circuit with the human design circuit, there is a reduction of one gate in the proposed full adder design. So, there will be less area and less power consumption in the proposed design which is suitable for designing portable low power circuits. On the other hand, comparing with the circuit designed by Ushie [10], there is no reduction in gate count. But, designed circuit shown in [10] contains 3 XOR and 2 AND gates whereas, the proposed design contains 2 XOR, 2 AND and 1 OR gates. In the proposed design 1 XOR gate is replaced by 1 OR gate. Though apparently there is no reduction in gate count but there is reduction in the area as OR gate consumes less area than that of XOR gate. Apart from that there is also a reduction in the number of iterations require to achieve the optimal design. The number of iterations in designing the 'Sum' circuit is reduced by 54.45% in the proposed design. The number of iterations in designing the 'Carry' circuit is reduced by 70.67% in the proposed design. There is an overall reduction of 67.10% in the proposed design in terms of number of iterations. So, comparing with the circuit designed in [10] using PSO there is a reduction in area and also in the number of iterations. Due to reduction in area, chip count can be increased and due to reduction in iterations, computational complexity is reduced.

Finally, it has been found that PSO based technique is very useful to design digital circuits. Using PSO, digital circuits can be designed with lesser number of gates in comparison to human design methods. However, parameters used in PSO are not generalized. So, the parameter values in PSO have been selected carefully to attain optimum solution with lesser computational complexity.

REFERENCES

[1] M. Kamaugh, "A Map Method for Synthesis of Combinational Logic Circuits," Transactions of the AIEE, Communication and Electronics, Vol. 721, pp. 593-599, 1953.
[2] W.V. Quine, "A way to Simplify Truth Functions," American Mathematical Monthly, Vol. 62:9, pp. 627-631, 1955.
[3] T. Sasao, editor, "Logic Synthesis and Optimization", Kluwcr Academic Press, 1993.
[4] C. Yonggang, Y. Fengjie, S. Jigui, "A New Particle Swam Optimization Algorithm," Journal of Jilin University, vol. 24(2), pp. 181-183, 2003.
[5] V.G. Gudise and G.K. Venayagamoorthy, "Evolving Digital Circuits Using Particle Swarm," Neural Networks, Proceedings IEEE Joint Conference vol. 1, pp. 468-472, 2003.
[6] V.G. Gudise and G.K. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Back Propagation as Training Algorithms for Neural Networks," IEEE Swarm Intelligence Symposium, April 2003.
[7] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," Proceedings, IEEE International Conference on Neural Networks, Vol. 4, pp. 1942-1948, 1995.
[8] R.C. Eberhart, J. Kennedy, "A New Optimizer Using Particle Swarm Theory, Proceedings of 6th International Symposium on Micro Machine and Human Science," Nagoya, Japan, IEEE Service Center Piscataway NJ, pp. 39-43, 1995.
[9] S.J. Louis, "Genetic Learning for Combinational Logic Design," 2003.
[10] Ushie, J. Ogri, O.J.A. Etim, I. Prosper, "Optimizing Digital Combinational Circuit Using Particle Swarm Optimization Technique," Latin American Journal of Physics Education, Vol. 6, No. 1, pp. 72-77, March 2012.
[11] C.A.C. Coello, E.H. Luna and A. Hernandez, "Use of Particle Swarm Optimization to Design Combinational Logic Circuits," Embedded Systems, Vol. 2606, pp. 398-409, Springer Link 2003.
[12] J.P. Yang, I. Shou, Dashu, Taiwan; C.K. Kung, F.T. Liu, Y.J. Chen, "Logic Circuit Design by Neural Network and PSO Algorithm," Proceedings IEEE Conference, pp. 456-459, PCSPA 2010.
[13] Y. Shi, R.C. Eberhart, "Parameter Selection in Particle Swarm Optimization," Evolutionary Programming VII, vol. 1447, pp. 590 – 600, Springer Link 1998.
[14] R.C. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources," in Proceedings. IEEE Conference. Evolutionary Computing, vol. 1, pp. 81–86, May 2001.
[15] Y.D. Valle, G.K. Venayagamoorthy, S. Mohagheghi, J.C. Hernandez, R. G. Harley, "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems," IEEE transactions on evolutionary computation, vol. 12, no. 2, pp. 171-195, April 2008.
[16] S. Masrom, S.Z.Z. Abidin, N. Omar, K. Nasir, "Time-Varying Mutation in Particle Swarm Optimization," Intelligent Information and Data- base Systems, 5th Asian Conference, ACIIDS, Kuala Lumpur, Malaysia, Proceeding, Part 1, pp. 31 – 40, March 2013.

978-1-4673-9939-5/16/$31.00 ©2016 IEEE