FPGA Placement Optimization using Firefly Algorithm

Subhrapratim Nath[1a*], Avik Kumar Chakravarty[1b], Sudipta Ghosh[1c]
Subir Kumar Sarkar[2d]

[1]Departments of [a,b]CSE & [c]ECE, Meghnad Saha Institute of Technology, Kolkata 700150, India

[2]Electronics & Telecommunication Engineering, Jadavpur University, Kolkata 700032, India

*{[a]*suvro.n, [b]avikkrc, [c]sudipta.ghosh123 }@gmail.com*

[d]su_sircir@yahoo.co.in

Abstract: Field Programmable Gate Array (FPGA) technology has been surging in popularity across all industries. Digital circuit design using FPGA provides many benefits over previous methods like ASIC implementation. However, the fixed hardware structure of FPGA demands an efficient placement and routing technique for high-performance goals. It has already been demonstrated that metaheuristic algorithms like Particle Swarm Optimization (PSO) can successfully optimize circuit designs for the FPGA placement and routing problem. The limitation of PSO lies in the explosion of particle swarms due to unbounded exploration. This paper proposes the application of Firefly Algorithm (FA) as an alternative and competitive metaheuristic solution for FPGA placement optimization. The design optimization of a single BCD counter circuit is performed by applying the proposed algorithm on Xilinx software generated netlist and the simulation result is compared with the existing PSO based placement techniques aiming at better placement optimization and utilization of FPGA resources.
Keywords: FPGA placement; routing; firefly algorithm; wirelength minimization; metaheuristics.

1 INTRODUCTION

Field Programmable Gate Array (FPGA) is an end-user configurable IC used to implement digital circuits. It consists of configurable logic blocks (CLBs) each of which can be programmed to function as a simple logic gate or perform some complex combinational logic. FPGA owes its popularity to the many benefits it provides. These include higher performance from hardware parallelism, lower cost and easier maintenance due to re-programmability and lower time-to-market because of rapid prototyping. All these advantages have caused FPGAs to gradually replace Application Specific Integrated Circuits (ASICs) since their invention. Availability of higher level tools for FPGA programming has boosted the adoption of these chips by beginners to this technology.

The number of logic blocks in FPGAs have increased significantly with fabrication geometry shrinking to deep-submicron levels and the consequent exponential growth in packaging density. FPGA hardware components being fabricated before circuit design, routing channels are limited and so their efficient utilization is very critical to achieve the maximum possible placement and routing without compromising on circuit performance.

Several optimization techniques have been successfully applied for the FPGA placement problem which include min-cut [1], quadratic [2], simulated annealing (SA) [3], genetic algorithm (GA) [4], ant colony optimization (ACO) [5] and particle swarm optimization (PSO) [6]-[8]. The recursive partitioning procedure in min-cut algorithm causes the solution to vary where the global optima may not reach and the quadratic technique is comparatively fast but cannot optimize timing and wire length at once. Evolutionary computing methods like SA and GA, and Metaheuristic algorithms like ACO and PSO converge to a global optimum in most cases unlike gradient based optimization techniques that get stranded in local optima. Metaheuristic is a higher-level heuristic in the sense that it searches for a heuristic that best suits the current problem, but it itself remains independent of a problem's intricacies and maintains a more general viewpoint of optimization than a heuristic. Swarm intelligence is a group behavior in which a number of decentralized agents keep sharing information with one another following some simple rules and, over time, this leads to the emergence of a collective intelligence or global knowledge that any individual agent is unaware of. Many of these FPGA Placement algorithms employ swarm intelligence for performing population-based stochastic optimization and are able to escape local optima because of their non-linear iterative characteristics.

Simple PSO has been applied in [6] while the results of improved variants of PSO have been explored in [7] and [8]. PSO is a population based optimization technique, mathematically modelled on the basis of bird flocking behavior in an unexplored space. In PSO, a particle in the search space represents a solution to a problem where each particle's fitness is defined by the quality of its position. The particles in the search space move around with a certain velocity, which depends on its own historical best position together with the best position obtained so far in the swarm. In an iterative way, the position as well as the velocity of each particle gets updated until it converges to an optimized solution. PSO operates with unbounded exploration of the search space which can cause individual members of the particle swarm to drift farther and farther apart from the solution region. This results in a phenomenon of divergence termed as explosion of the swarm and it finds no solution at all.

This paper proposes the application of another population-based metaheuristic, the Firefly Algorithm (FA) [9], to the FPGA placement problem. FA is a nature-inspired metaheuristic approach where the collective behavior of fireflies is mathematically modelled for solving optimization problems using swarm intelligence. The proposed new approach based on FA is applied to the FPGA Placement problem where the design of a BCD counter specified in X74_168 [6] is implemented on a Xilinx FPGA device and the netlist output generated is used as input to Firefly Algorithm implemented in MATLAB for placement optimization. The results are compared with the existing PSO based FPGA placement techniques. The remaining of this paper is organized as follows: Section 2 gives a brief overview of FPGA architecture and the associated placement and routing problem; Section 3 deals with FA and its application to FPGA placement; Section 4 describes the experimental details and results; followed by conclusion in Section 5.

2 PRELIMINARIES

2.1 FPGA Architecture

A Xilinx FPGA consists of three main configurable components – configurable logic blocks (CLBs), interconnects and input or output blocks (IOBs). For a symmetrical array FPGA as shown in Fig. 1, the CLBs are uniformly distributed in a matrix with programmable interconnects between them and the IOBs are located along the periphery of the matrix.

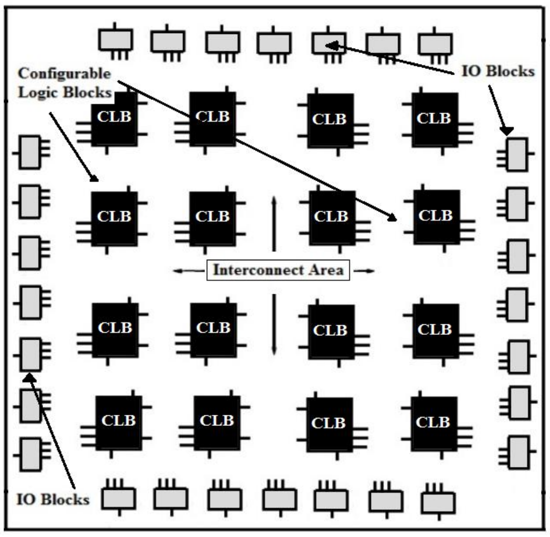

Fig. 1. Symmetrical FPGA Model

The CLB is the functional unit of the FPGA which can be programmed to implement any user-defined logic. The interconnects contain programmable switches that contribute most of the flexibility of the FPGA by constructing a programmable routing network linking the inputs and outputs of CLBs and IOBs. The I/O pads serve as the interface between the CLBs and the external pins of the FPGA IC for making off-chip connections. Static Random Access Memory (SRAM) cells are used as look-up tables (LUTs) to store the user-defined functionality of the FPGA. The circuit design is converted into a bit-stream and transferred onto the SRAM cells to program the FPGA as required.

2.2 FPGA Placement and Routing Protocol

FPGA placement means mapping the partitioned blocks of the circuit netlist onto the logic blocks of the FPGA. Routing implies mapping the interconnections specified in the netlist to the routing resources of the FPGA to preserve the connectivity between the partitioned blocks of the circuit. Efficient utilization of FPGA resources can be attained by intelligent placement and routing using Computer Aided Design (CAD) tools.

A placement algorithm attempts to optimize certain goals by comparing different possible placements of the circuit onto the FPGA. The optimization objective may be broadly defined to be one or a combination of minimizing the total wire-length, balancing the density of wiring or maximizing the speed of the circuit. Placement is an NP-complete problem and thus some form of heuristics may be applied to obtain practical solutions within reasonable time restrictions.

This work is based on the objective of minimizing the total length of interconnects in the circuit i.e. wire-length driven placement optimization. Therefore, minimizing the cost function in (1) is the primary objective in solving the problem.

$$\text{Cost Function} = \sum_{c1,c2} \left(|x_{c1} - x_{c2}| + |y_{c1} - y_{c2}| \right) \qquad (1)$$

where $c1, c2$ is a pair of CLBs that are at the two ends of an interconnect and (x_{c1}, y_{c1}) & (x_{c2}, y_{c2}) are the locations of the CLBs $c1$ and $c2$ respectively.

3 FIREFLY ALGORITHM BASED FPGA PLACEMENT

3.1 Firefly Algorithm

The Firefly Algorithm (FA) is a biologically inspired algorithm derived from the social behavior of fireflies. Most firefly species produce short and rhythmic flashes by bioluminescence. Though the true functions of such signaling systems are still being debated, these flashes are primarily meant to attract mating partners. They can also attract potential prey or serve as a warning mechanism for the swarm.

FA was designed by Xin-She Yang [9] in 2009 and follows three idealized rules:

a) Fireflies are considered to be unisex. So, a firefly will move towards the brighter members of the swarm, regardless of their sex.

b) The attractiveness between fireflies are proportional to their brightness. Given two fireflies, the one with lower light intensity will move towards the brighter one. The brightest firefly in a swarm will explore the search space randomly.

c) The brightness of a firefly will be determined by the fitness value of its location in the search space, obtained from the objective function for the optimization problem.

3.2 Principle of Firefly Algorithm

The mathematical model of the Firefly Algorithm [10]-[12] involves two main components: the variation of light intensity and the determination of attractiveness. For purposes of simplicity, it is assumed that the attractiveness of fireflies is dependent on their brightness, which in turn depends on the fitness values of the search space. This means $I(x) \propto f(x)$ where $I(x)$ gives the light intensity of the firefly at the location x and f is the objective function. The attractiveness β, however, is not only dependent on the brightness of the firefly but also on the perception of that brightness by other fireflies, which is affected by the distance between the fireflies. So, it is imperative to take into consideration that light intensity $I(r)$ decreases with distance r according to the inverse square law as in (2).

$$I(r) = I_0 / r^2 \qquad (2)$$

where I_0 is the source intensity. Also, some light energy will be absorbed by the medium and the resulting intensity will vary with distance as given in (3).

$$I(r) = I_0 e^{-\gamma r} \qquad (3)$$

where γ is the fixed light absorption coefficient of the medium. The combined effect of inverse square law and absorption can be approximated as the Gaussian function shown in (4).

$$I(r) = I_0 e^{-\gamma r^2} \qquad (4)$$

which helps to avoid the singularity at $r = 0$ in (2). Since a firefly's attractiveness is proportional to its brightness as perceived by other fireflies, the attractiveness $\beta(r)$ of the source firefly to the observing firefly at a distance r from the source can be defined as

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{5}$$

To reduce the time complexity, the exponential function is approximated with $1/(1+r^2)$ so that (5) becomes

$$\beta(r) = \frac{\beta_0}{1+\gamma r^2} \tag{6}$$

Characteristic distance $\Gamma = 1/\sqrt{\gamma}$ is the distance over which the attractiveness changes significantly from β_0 to $\beta_0 e^{-1}$ for (5) and to $\beta_0/2$ for (6). The parameter γ influences the attractiveness in (6) and consequently also determines the speed of convergence.

The Cartesian distance between two fireflies i and j at locations x_i and x_j respectively is given by

$$r_{i,j} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{7}$$

where $x_{i,k}$ is the k-th component of the spatial coordinate x_i of the i-th firefly. For two-dimensional space, this becomes

$$r_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{8}$$

The movement of a firefly i attracted to a brighter firefly j is given by (9) as

$$x_{i+1} = x_i + \beta_0 e^{-\gamma r_{i,j}^2} \left(x_j - x_i\right) + \alpha\varepsilon \tag{9}$$

where the second term is due to attraction and the third term is a random component with α being the randomization parameter. ε is a random vector that follows the Gaussian distribution or uniform distribution. In this algorithm, ε is implemented by $(rand - 1/2)$ where $rand$ is a random number generator uniformly distributed in [0,1] and the parameters $\beta_0 = 1$ and $\alpha \in [0,1]$ are used.

3.3 Firefly Algorithm for FPGA Placement

A firefly represents a possible solution for FPGA placement. It includes information regarding the position of the CLBs, the IOBs and interconnects on the FPGA fabric. A swarm of such fireflies, each representing a possible placement of the logic blocks, is initiated with random values. Since IOB and CLB positions can never overlap, one of them is kept fixed at each update of the swarm. IOB positions are fixed while CLB positions are iterated over until the fitness function reaches a relatively better value and then the CLB positions are fixed and IOB positions are varied. This two-phase process continues repeatedly throughout the optimization process until the fitness function shows no change in value for updates of the swarm.

The Cartesian co-ordinates of m CLBs and n IOBs of a circuit can be represented by two matrices of dimensions $2\times m$ and $2\times n$ respectively and the circuit blocks are allowed to move within some defined limits of the FPGA placeholders. When the IOBs are fixed, the matrix of CLB co-ordinates becomes the position vector of a firefly. The fitness of the circuit placement is calculated using (1) and the brightness of the firefly is derived proportionally from that fitness. A swarm of such fireflies is obtained by randomly generating block placements within the defined physical bounds of placeholders and the fitness of each firefly is calculated using (6). In a process known as updating the swarm, FA is executed on the population which, for each possible pair of fireflies in the swarm, causes the brighter firefly in a pair to draw the other firefly towards itself using (9). A change in position of a firefly to an unvisited point in the search space indicates that a new circuit placement is obtained for which the fitness must be calculated. After the assessment of all new positions, the next swarm update may be carried out identically by FA. Finally, when a satisfactory fitness is reached or the swarm has converged and can improve no further, the position vector of the brightest firefly gives the global optimum for the placement problem with acceptable approximations introduced by heuristics.

4 EXPERIMENTAL SETUP

The input for an algorithm performing placement optimization is the digital circuit netlist generated using a CAD tool. A netlist is a description of the connectivity of an electronic circuit which divides the circuit into logical blocks and defines the interconnections between those blocks. For the purpose of experimenting with the proposed FA based FPGA placement algorithm, the digital circuit of a simple BCD counter specified as X74_168 [6] is considered. The design of the counter is implemented in Xilinx software and the netlist output is used to target hardware implementation on a Xilinx FPGA device. X74_168 requires 7 CLBs and 14 IOBs for Xilinx XC4000 FPGA implementation. The netlist is input to Firefly Algorithm implemented in MATLAB for placement optimization. The CLB & IOB locations and interconnect routes for which the cost is defined in (1) are obtained as output from FA. The parameters of FA are tuned to find the most effective placement and routing of the counter circuit onto the FPGA IC. The netlist is also fed to Conventional PSO algorithm as in [8] with acceleration constants $c_1 = c_2 = 1.01$ and weight $w = 0.75$ and the generated output is used for comparison. The experiments on both the algorithms are conducted 25 times.

5 RESULTS & DISCUSSION

The best values of the cost function for BCD counter achieved on applying the proposed FA based algorithm and conventional PSO based algorithm, and the corresponding generated minimum wirelengths are recorded in Table I. The proposed FA based placement algorithm is compared with PSO in [6] and TVIW PSO in [7] in order to evaluate the efficiency of their performance.

Table 1. Comparison of placement optimization algorithms.

Optimization Algorithm	Swarm Size	No. of Iteration	Final Wirelength	Best Value Cost Fn.
PSO in [6]	25	2000	337	--
Conventional PSO	40	1000	252	51.2
TVIW PSO in [7]	30	1000	236	36.11
FA	40	1000	221	32.2

The proposed Firefly (FA) algorithm based placement yields the best cost function values for the BCD counter design. Although FA runs with increased swarm size compared to TVIW PSO in [7] and PSO in [6], FA based placement technique outperforms both the algorithms, generating lowest final wirelength value of ‘221’. The results of conventional PSO algorithm and FA on final CLB and IOB placement are shown in Fig. 2 and Fig. 3 respectively.

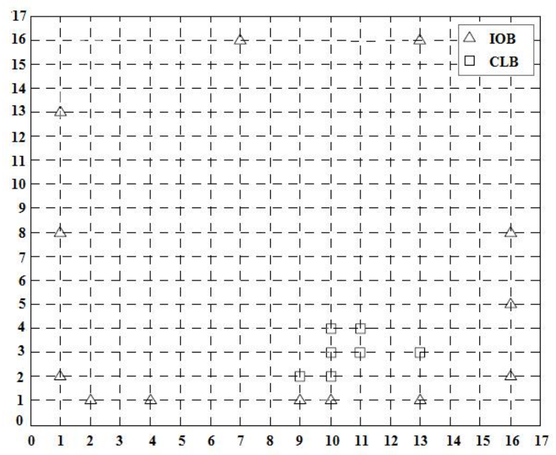

Fig. 2. Conventional PSO based Final placement of CLB and I/O blocks.

The average system execution time in case of Firefly based placement technique is higher compared to conventional PSO but it generates lowest Cost function at 32.2 for BCD counter problem which is even less compared to best value cost function of TVIW PSO in [7]. So, it can be clearly stated that the proposed FA based placement technique can effectively optimize the placement of CLB and I/O blocks thereby ensuring minimized final wire length.

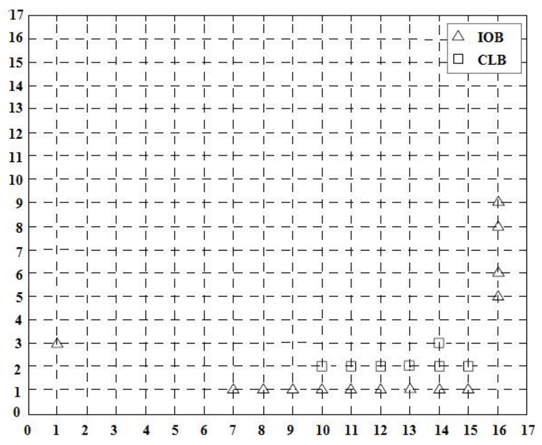

Fig. 3. Firefly based Final placement of CLB and I/O blocks.

6 CONCLUSION

This paper proposes new approach based on Firefly algorithm for FPGA Placement optimization solution. The parameters of Firefly Algorithm are tuned to find the most effective placement and routing of the BCD counter circuit onto the FPGA IC and it generates lowest minimum wirelength and better cost function value than the existing PSO based placement algorithms. Although the proposed FA based placement algorithm finds limitation with high but tolerable time complexity, it performs better in comparison to the other existing PSO based FPGA placement technique thereby achieving the best performance of the circuit and the most efficient utilization of FPGA resources.

REFERENCES

[1] P. Maidee, C. Ababei and K. Bazargan, 2005. Timing-driven partitioning-based placement for island style FPGAs, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 3, pp. 395-406.

[2] Y. Xu and M. A. S. Khalid, 2005. QPF: efficient quadratic placement for FPGAs, Int’l Conf. Field Programmable Logic and Applications, pp. 555-558.

[3] X. Guo, T. Wang, Z. Chen, L. Wang and W. Zhao, 2009. Fast FPGA placement

algorithm using Quantum Genetic Algorithm with Simulated Annealing, IEEE 8th Int'l Conf. ASIC, pp. 730-733.

[4] M. Yang, A. E. Almaini and L. Wang, 2007. FPGA placement by using a genetic algorithm, EngineerIT, pp. 50-53.

[5] K. Wang and N. Xu, 2009, Ant Colony Optimization for symmetrical FPGA placement, IEEE 11th Int'l Conf. Computer-Aided Design and Computer Graphics, pp. 561-563.

[6] G. K. Venayagamoorthy and V. G. Gudise, 2004. Swarm intelligence for digital circuits implementation on field programmable gate arrays platforms, NASA/DoD Conf. Evolvable Hardware, pp. 83-86.

[7] M. El-Abd, H. Hassan and M. S. Kamel, 2009. Discrete and continuous particle swarm optimization for FPGA placement, IEEE Cong. Evolutionary Computation, pp. 706-711.

[8] P. K. Rout, D. P. Acharya and G. Panda, 2010. Novel PSO based FPGA Placement Techniques, Int'l Conf. on Computer & Communication Technology, pp. 630-634.

[9] X. S. Yang, 2009. Firefly Algorithms for Multimodal Optimization, 5th Int'l Symp. Stochastic Algorithms, Foundations and Applications, pp. 169-178.

[10] S. Yang, S. S. S. Hosseini and A. H. Gandomi, 2012, Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect, Applied Soft Computing, vol. 12, no. 3, pp. 1180-1186.

[11] Sh. M. Farahani, A. A. Abshouri, B. Nasiri and M. R. Meybodi, 2011, A Gaussian Firefly Algorithm, Int'l Journal of Machine Learning and Computing, vol. 1, issue 5, pp. 448-453.

[12] X. S. Yang, 2011, Chaos Enhanced Firefly Algorithm with Automatic Parameter Tuning, Int'l Journal of Swarm Intelligence Research, vol. 4, issue 2, pp. 1-11.