A Novel Placement Algorithm for Symmetrical FPGA

Wenyao Xu* Kejun Xu Xinmin Xu
Institute of Electronic Circuit and Information System, Zhejiang University, Hangzhou 310027, P. R. China
*Email: wenyao.xu@gmail.com

Abstract
Placement becomes a vital current concern in FPGA CAD flow. This paper presents a novel FPGA placement algorithm based on ant colony optimization (ACO), a new meta-heuristic algorithm characterized by inherent parallelism, positive feedback mechanism, and stochastic decision policy with swarm intelligence. We test the performance of our proposed algorithm using a set of Microelectronics Center of North Carolina (MCNC) benchmark circuits on island-style architecture FPGA, and have a comprehensive comparison with simulated annealing (SA), genetic algorithm (GA) and hybrid meta-heuristic approach mixed GA and SA. The experimental results show that our placement algorithm can achieves promising performance and is a potential approach for FPGA placement.

1. Introduction

Placement is a pivotal step in FPGA CAD flow. It determines how circuit blocks (including I/O blocks and logic blocks) in the netlist are mapped onto physical locations in FPGA, considering area, speed and power dissipation. The quality of the placement greatly affects the following routing phase and the holistic performance of FPGA. Due to the fact that placement is NP-hard, the challenges of the placement become tremendous with increasing device density which reaches ten-million gates per chip. Recently more and more researchers are apt to using meta-heuristics to get the optimization solution of this problem, such as Simulated Annealing (SA), Genetic Algorithm (GA), Tabu Search, Particle Swarm Optimization (PSO) and hybrid meta-heuristic algorithm.

In this paper, we present a new method for FPGA placement based on ant colony optimization (ACO), which is a novel population-based, nature-inspired meta-heuristic algorithm suitable for solving NP-hard combinatorial optimization (CO) problems. To the best of our knowledge, this is the first application of ACO on FPGA placement.

2. Background

In FPGA placement phase, the blocks in the circuit netlist, including input pads, output pads and logic blocks, are assigned to specified physical locations in FPGA, as shown in Figure 1.

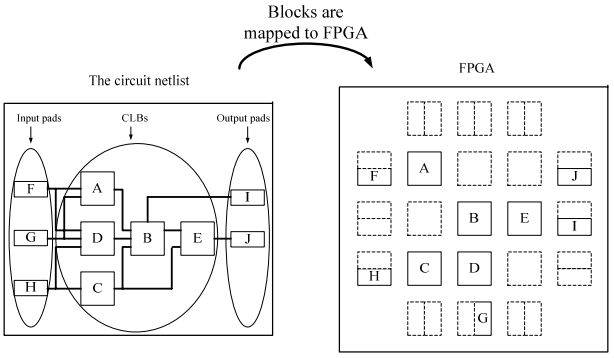

Figure 1. The Processing of FPGA Placement

Ant colony optimization is a novel and animate algorithm which is inspired by research on the behavior of ant colony. Many simulations and experimental results demonstrate that it is a robust algorithm based on population and a promising way for getting sub-optimal solutions in a reasonable running time. Our main design concept of the placement algorithm is prompted by them because FPGA placement also belongs to combinatorial optimization problems.

3. FPGA Placement Algorithm Based On ACO

FPGA placement algorithm mimics the foregoing search strategy. Pseudo-code of an ACO-based placer is shown in Figure 2. The placement algorithm consists of an initialization for the pretreatment and a loop for finding the optimization solution. In the following we will state how ACO works in FPGA placement concretely.

1-4244-1132-7/07/$25.00 © 2007 IEEE

```
Begin Algorithm
 STEP 1:
     InitialParameter();
     InitialPlacementOrder ();
 STEP 2:
     While( ExitCriterion( )==False){
           For s = 1 to n { /* n denotes num_blocks */
               i= PlaceOrder(s);
               For k = 1 to num_ants {
                  ConstructAntSolution(k,i,j);   /*Ant[k] maps
                                                 block[i] to CLB[j]
                                                 by Eq.(1)*/
                  UpdateTabu(j);
               }
           }
        RecordBestPlacement();
        UpdatePheromone();
     }
End Algorithm
```

Figure 2. Pseudo-code of an ACO-based placer

Parameters should be set properly as the performance of ACO is enormously sensitive to them. Commonly we evaluate them through theory and preliminary experiments.

Every ant in the colony assigns block i to physical location j, consequently all the blocks in the circuit netlist will be mapped to the specified physical FPGA array. The optimization goals are to minimize the required wiring, balance the wiring density and maximize circuit speed. For instance, a colony has m ants. The probability that ant k (k=1, 2 … m) assigns block i to physical location j is given by Eq.1:

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\notin tabu_k}\left(\tau_{il}^{\alpha}\eta_{il}^{\beta}\right)} & if\ \ j\notin tabu_k\\ 0 & otherwise\end{cases} \qquad (1)$$

where τ_{ij} represents the pheromone value, associated with the desirability of mapping block i to location j; η_{ij} denotes the initial heuristic information, also called 'potential goodness', which guides the early several placements. After some iterations (based on the strategy), the experiences acquired by ants, stored by τ_{ij}, accumulates. With that the heuristic information would automatically loses impact in the following process. Intuitively we consider that the more centric a CLB locates in FPGA, the stronger ability of connection it has; the greater connection a block has in the circuit netlist, the more centric it should be implemented. Hence we evaluate heuristic information in Eq.2:

$$\eta_{ij}=Cross_i\cdot Connect_j \qquad (2)$$

Where $Cross_i$ denotes the connective extent between block i and other blocks. The straightforward way to evaluate $Cross_i$ is counting how many blocks are connected to block i. As shown Figure 1, block D connects to block F, block G, block H and block B, thus Cross of block D is 4, so are others; the calculative formula is Eq.3:

$$Cross_i=\sum_{k=1}^{num\_net} num\_pin(k) \qquad (3)$$

$Connect_j$ is defined for the sum distance degree between location j and other locations. The expression of Connect is given by Eq.4.

$$Connect_j=\sum_{i\in clb} dis\tan ce(i,j) \qquad (4)$$

As figure 3, the sum distance from location 5 to other locations is $Connect_5$= (2+1+2+1+1+2+1+2) =12, and the sum distance for location 1 to others is $Connect_1$= (1+2+1+2+3+2+3+4) =18.

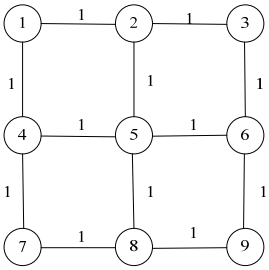

Figure 3. The distance of locations in FPGA

Moreover, α and β are positive parameters, whose values determine the relative importance of pheromone versus heuristic information.

Accordingly, the probability p_{ij}^{k} is a compromise between the pheromone value and the heuristic information on coupling (i, j). To avoid that different blocks would be assigned to the same location, we must construct a data

structure, called a tabu list, whose length is the same as the number of blocks, for each ant. This work is actualized by the function Update_Tabu$_k$(j). When block i is implemented on an available block such as location j, the tabu list will record this circs with the purpose that location j couldn't be reused by other blocks. These construction steps are repeated until a complete assignment is obtained. After a cycle, the tabu list will be emptied and every ant is free to choose any location in the new construction. By the way, the best placement should be recorded when all the ants finish their jobs, realized by the function RecordBestPlacement() in Figure 4.

After all the ants have assigned logic blocks implementing the circuit to physical locations in FPGA, an effective evaluation is a must for estimating the quality of the current placement solution. Here our optimization goals are the caused delay and the consumed area, so we refer to Eq.5 as the fitness functions which are evolved from, whose validity has been widely accepted. In this function $Cost_{delay}$ reflects the caused delay and $Cost_{area}$ stands for the consumed area objectively in FPGA placement

$$\left.\begin{aligned} Fitness &= \frac{1}{\lambda Cost_{area} + (1-\lambda) Cost_{delay}} \\ Cost_{area} &= \sum_{i=1}^{N_{nets}} q(i) \cdot [bb_x(i) + bb_y(i)] \\ Cost_{delay} &= \sum_{\vee i,j \subset circiut} Delay\_cost(i,j) \end{aligned}\right\} \qquad (5)$$

Then it is the stage for the pheromone update, the aim of which is to increase the pheromone values associated with good or promising placement solutions, and to decrease those that are associated with bad ones. Usually, it is achieved by decreasing all the pheromone values through pheromone evaporation, and by increasing the pheromone values associated with a chosen set of good results. In order to avoid the premature of the searching processing, we import some characteristics of MMAS to our FPGA placement algorithm. Thus the updating function is given by Eq.6:

$$\tau_{ij}^{new} = \begin{cases} \tau_{\min} & if\ \tau_{ij}^{new} < \tau_{\min} \\ (1-\rho)\tau_{ij}^{old} + \Delta\tau_{ij}^{best} & if\ \tau_{\min} \le \tau_{ij}^{new} \le \tau_{\max} \\ \tau_{\max} & if\ \tau_{ij}^{new} > \tau_{\max} \end{cases} \qquad (6)$$

Where $\Delta\tau_{ij}^{best}$ is the pheromone update value defined by:

$$\Delta\tau_{ij}^{best} = \begin{cases} Q \cdot Fitness & ,if\ ant_{best}\ mapping\ B_i\ to\ L_j \\ 0 & ,otherwise \end{cases} \qquad (7)$$

Where ρ is the evaporation rate. The large value of ρ will improve the global searching ability but lessen the convergence rate of the algorithm, and vice verse. Q is the parameter called the intensity of pheromone, and it has slight effect on the performance of the algorithm. Explicitly the values of the pheromone are limited between the minimum $\tau_{\min}$ and the maximum $\tau_{\max}$, and only the best ant can update the pheromone trails. T.StÄutzle and H. Hoos suggest that both of them should be chosen experimentally based on the problem at hand, and usually they have strong influence on the algorithm performance. Here we evaluate them with the following formula, which is from theoretical analysis and experiments:

$$\begin{cases} \tau_{\max} = \dfrac{Fitness_{best}}{1-\rho} \\ \tau_{\min} = \dfrac{\rho \cdot Fitness_{init}}{3} \end{cases} \qquad (8)$$

Where $Fitness_{best}$ is the best fitness we forecast. From preliminary experiments we summarize that it is approximately 35% of the initial fitness.

The number of ants enormously affects the running-efficiency of this algorithm. On the basis of the experiments we set it as the following function:

$$m = \begin{cases} 0.67 \cdot num\_blocks & if\ num\_blocks < 100 \\ 10 \cdot \sqrt{num\_blocks} & if\ num\_blocks > 100 \end{cases} \qquad (9)$$

Concerning the assignment order, we prefer to use a preordering schedule of the blocks, which is constructed according to some heuristic information, instead of a random one. It's recognized that the block having stronger connections should have greater probability to be placed in the center of FPGA. The blocks are ordered according to the value of Cross (count in Eq.3), which is realized in the function InitialOrder().

In this way, the optimal placement (with the correspond-

ing high fitness value) will be found when the exit criterion is true, which indicates that the explorative process is complete.

4. Experiment

Now the placement algorithm described above is compared to SA, GA and GASA comprehensively. The state-of-the-art V-Placer utilizes SA as placement algorithm. GA and GASA are used by [1] and [2] respectively. To evaluate their performance objectively, we use the number of tracks required to complete global routing for measuring the quality of placement solution. Here we use the same routing tools, V-Router, with the same default parameter setting. Table 1 lists the tracks required by global routing based on different placement algorithms. In addition, we use the results presented in [1] for GA and [2] for GASA.

The cost function defined in [3] can reflect the quality of placement solution, which includes the delay aspect and the area aspect. We further compare our placer with SA and GASA in terms of placement cost. In view of clarity, we use unitary results listed in Table 2. As shown in it, the placer based ACO achieves promising performance compared to SA and GASA.

Table 1 Comparison of Tracks among SA, GA, GASA and ACO

Circuit	Tracks Required by Global Routing			
	SA	GA	GASA	ACO
9symml	5	5	5	5
alu2	6	6	6	6
apex7	5	5	5	5
e64	8	8	8	8
example2	5	5	5	5
k2	9	10	9	9
terml	5	5	5	5
too-lrg	7	7	7	7
vda	8	8	8	8
Total	58	59	58	58

Table 2. The comparison results of placement cost among SA, GASA and ACO

Circuit	Cost		
	SA	GASA	ACO
9symml	1x	1.0043x	1.0012x
alu2	1x	1.0048x	1.0034x
apex7	1x	1.0000x	1.0002x
e64	1x	0.9986x	0.9988x
example2	1x	0.9978x	1.0009x
k2	1x	0.9998x	1.0000x
term1	1x	1.0000x	1.0015x
too-lrg	1x	0.9989x	1.0007x
vda	1x	1.0000x	0.9950x
Average	1x	1.0005x	1.0002x

5. Conclusion

In this paper we present a novel FPGA placement algorithm based on ACO meta-heuristic, and compare it with other placement algorithms. ACO is a meta-heuristic which is characterized as good global search ability and robustness. Like ACO, the placer based on it can obtain satisfied solutions. The experimental results show its competitive performance on FPGA placement. However, to the best of our knowledge, there are only a few applications in this domain. The author hopes this paper could prompt the further study of ACO among researchers in the development of performance.

References

[1] M. Yang, etc, Proc. of IEEE PhD Research in Microelectronics and Electronics (PRIME'05), An Evolutionary Approach for Symmetrical FPGA Placement, pp.143-146, 2005

[2] M. Yang, etc, Journal of Electronics, FPGA Placement Optimization by Two-Step Unified Genetic Algorithm and Simulated Annealing Algorithm, pp.632-636, Oct. 2005.

[3] V. Betz, J. Rose, and A. Marquardt, Architecture and CAD for Deep-Submicron FPGAs. Boston: Kluwer Academic Publishers, 1999