Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(5):985-989

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

A novel FPGA segmentation method based on the improved ant colony optimization

Fei Yang

School of Computer Science and Technology, Hubei Polytechnic University, Huangshi, P. R. China

---

ABSTRACT

The performance of Field Programmable Gate Array (FPGA) is closely related with its own reasonable segmentation, and the routing of FPGA forms when different functions perform on the segmented FPGA to meet different demands. For this reason, a novel FPGA segmentation method is proposed based on the improved ant colony optimization. Experimental results suggest that the proposed approach is feasible and correct.

Key words: Field Programmable Gate Array; Segmentation Method; Ant Colony Optimization

---

INTRODUCTION

FPGA is a kind of Application Specific Integrated Circuit (ASIC). It is a kind of general user programmable device, first developed in 1984 by the Xilinx Group. FPGA and CPLD (Complicated Programmable Logic Device), both programmable logic devices, have resulted from PAL (Programmable Array Logic) and GAL (General Array Logic), et al. Compared with previous PAL and GAL, FPGA /CPLD owns bigger scale, and it can take the place of tens of or even thousands of IC chips. Such a FPGA /CPLD is actually a subsystem component. This kind of chips has been accepted by electronic engineering designers worldwide and widely applied in aerospace, communication, medical, industrial control and other fields, especially suitable for the sample development and small-lot production of products.

FPGA is flexible in construction. Its logic units, programmable interconnect and FO units are all programmable, can realize different function and meet kinds of design demand [1]. Due to its rapid speed, low consumption and wide application, FPGA is especially suitable for the design of complicated systems [2]. It can also realize dynamic allocation, reconstruction of on-line systems (changing the circuit function upon demands at different moment during the system operation to make the system own several space-related or time-related tasks), hardware softening and software hardening [3].

SEGMENTATION DESIGN OF FPGA

The Segmentation Design of FPGA is the precondition of the Routing problem [4]. The segmentation has decided the layout of modules on the chip and the location of different pins on the module, and the interconnection information among different pins provides reference for the routing through the netlist [5].

Instead of discussing the specific relations between logic chips and pins, we will abstract the problem and the details are as below [6]. The segmentation design problem of FPGA includes row-based and array-based FPGA [7], namely one-dimensional problem and two-dimensional problem. Most of present studies focus on one-dimensional FPGA, and some studies on two-dimensional problems only propose basic ideas based on one-dimensional studies. In this paper, we try to explore how to use ant colony algorithms to solve the segmentation design of FPGA, and we focus on one-dimensional FPGA for convenience. First several signs are introduced for description.

Symbol	Meaning
L	The number of columns for the channel
N	The average number of nets contained by each channel
T	The number of tracks contained by channels
K	The maximum of segments occupied by a net
D	The allowable maximum of nets with endpoints on the same column
$h(x,l)$	The probability that net's length is l and its left endpoint is x
$f(l)$	The probability that net's length is l
$g(x)$	The probability that net's left endpoint is x

It is easy to get

$$f(l)=\sum_{x=1}^{L} f(x,l), g(x)=\sum_{l=1}^{L} h(x,l)$$

It is a FPGA that has been segmented. In the three tracks contained by the channel, the white circle represents segment points, so the first track has been segmented into two segments, the second three and the third two. Whether this kind of segmentation is reasonable or not depends on the probability that it has met the nets. The higher the probability is, the more reasonable the segmentation is.

Here is a brief description for K -segmentation problem. K -segmentation design problem: given L, N, T, K and $h(x,l)$, design a kind of track segmentation to maximize the success probability of K -segment routing problem (that will be discussed in the next chapter). In other words, this kind of segmentation should meet nets as many as possible to complete K -segment routing.

THE PROPOSED APPROACH

This algorithm is based on the graph theory and solves FPGA from three stages, similar to the matching algorithm and minimal spanning tree. Besides, the second and third stages of the algorithm are the same as that of the matching algorithm. The following is some improvement on the first stage.

Similar to algorithms based on Kruskal, the segmentation problem is converted into one to find a forest that meets the following two special requirements in a K-partite graph: The weight sum is maximal, and any two vertexes of each tree do not belong to one part simultaneously.

The basic idea is to combine another classical algorithm (the Prim algorithm) of the minimal spanning tree with ant colony algorithms to generate an ant colony algorithm based on Prim algorithms to get better solutions.

Prim algorithms are mainly used to get the minimal spanning tree in a weighted connected simple graph. (G, w) represents a weighted connected simple graph, then Prim algorithms proceed as below.

(1) Any $x_0 \in V(G)$, $l(x_0)=0, l(x)=\infty(x \neq x_0), V_0=\{x_0\}$, $T_0=x_0$ and $k=0$.

(2) Given $\forall x \in N_G(x_{k-1}) \mathrm{I} \overline{V}_{k-1}$ and $\overline{V}_{k-1}=V-V_{k-1}$, if $w(x_{k-1}x)<l(x)$, take the place of $l(x)$ with $w(x_{k-1}x_k)$. Select $x_k \in \overline{V_{k-1}}$, $l(x_k)=\min\{l(x): x \in \overline{V_{k-1}}\}$. Suppose $e_k=ux_k$, $u \in V_{k-1}$, so that $w(e_k)=l(x_k)$. Let $V_k=V_{k-1} \mathrm{U}(x_x)$, $T_k=T_{k-1}+e_k$.

(3) If $k<v-1$, turn to 2). If $k=v-1$, stop.

Now, the problem is to find a maximal forest full of trees. Here, the Prim algorithm is used to get these trees through slight improvement. In the previous Kruskal algorithm, edges are selected according to the weight value and the requirement that no null exists, and only inverting the selection order (the ascending order) in the minimal spanning tree can get the desired forest.

However, in Prim algorithms, edges are selected based on the marking number of vertexes and whether their endpoints are selected ones or not. Two alternatives here: 1) first assign negative value to the weight of edges in the graph, and added a number (the value should be larger than the maximal absolute value of edge weights) to it, then find the minimal forest satisfying constraint 2 in the obtained graph, finally restore the original problem and get the optimal solution. 2) Modify control criteria for the algorithm.

The first stage turns into that any $x_0 \in V(G)$, $l(x_0)=0, l(x)=0(x \neq x_0), V_0=\{x_0\}$, $T_0 = x_0$ and $k=0$. The second stage changes: given $\forall x \in N_G(x_{k-1}) \mathrm{I} \overline{V}_{k-1}$ and $\overline{V}_{k-1} = V - V_{k-1}$, if $w(x_{k-1}x) > l(x)$, then take the place of $l(x)$ with $w(x_{k-1}x_k)$. Select $x_k \in \overline{V_{k-1}}$ so that $l(x_k) = \max\{l(x): x \in \overline{V_{k-1}}\}$. Suppose $e_k = ux_k$, $u \in V_{k-1}$, so that $w(e_k) = l(x_k)$. Let $V_k = V_{k-1} \mathrm{U}(x_x)$, $T_k = T_{k-1} + e_k$. The termination condition of this algorithm is the same as that of Kruskal algorithms: no edges satisfying constraint 2 exist any more.

The improved Prim algorithm can be used to find the spanning tree with the maximal weight sum in the graph, and finally get the solution. However, the counterexample shows that neither the improved Prim algorithm nor Kruskal algorithm can obtain the optimal solution.

In a word, both Kruskal and Prim algorithms can be used to solve segmentation design of FPGA through transformation. They are so greedy that they try to add edges with maximal weights into the tree each time, so what they finally obtain is only a spanning tree with the maximal weight sum, but the desired is a forest. Therefore, they could not get the optimal solution because the selection of the first tree will influence the subsequent selections.

Ant colony algorithms can break down this deadlock. Due to their randomness, ant colony algorithms can achieve a nice result in solving TSP problems. Here, their great random search ability is combined with Prim algorithms to find the optimal solution to the problem.

In ant colony algorithms, the pheromone of ants starting from one vertex is stored in the edge, and in Kruskal algorithms, the selection of edges is only relevant with their weight value, so it is hard to combine Kruskal algorithms with ant colony algorithms.

Prim algorithms with the second improved alternative described in the above are adopted, namely obtaining the maximal tree through modifying control criteria of algorithms. First, list all candidates during the selection of each edge and label vertexes according to the labeling criteria in improved Prim algorithms, then select edges through ant colony optimization algorithms. When each ant has finished the construction of forests, ant colony optimization algorithms based on the Metropolis criterion are adopted to decide of which ant-passed edges the pheromone will be increased, and the iteration goes on until its number has reached the preset maximal one.

Since the above has analyzed the construction of m -partite weighted graph, the discussion here mainly focuses on how to find a forest in a m -partite weighted graph satisfying the following constraints: The weighted sum is maximal, and any two vertexes of any tree do not belong to one part simultaneously. The detailed implementation process is shown as below (relevant mathematical procedures will be shown in appendixes).

Step one: initialization. As to each edge e_{ij}, $\zeta = \max\{w(e_{ij})\} \eta_{ij} = w(e_{ij})/\zeta, \tau_{ij} = 0.05$ (set the original pheromone concentration and visibility of each edge). Determine the ant number m.

Step two: generate a satisfying maximal forest as to each ant through the combination of Prim algorithms and ant colony optimization algorithms.

Step three: update of pheromone.

Step four: termination conditions.

(1) Design Requirements. Network sharing platform for manufacturing resources aims to provide uniform 'intermediary' services for scattered manufacturing enterprises in a group, set up a bridge among each collaborative manufacturing enterprise and realize optimal utilization of manufacturing resources and information sharing. The design pf this platform should satisfy the following requirements:

1) Realize flexible inquiry and online retrieval of resources and information of each manufacturing enterprise, reach

dynamic management of manufacturing resources in a group and provide reliable manufacturing resource information for arranging and implementing manufacturing plans.

2) Timely publish manufacturing resource use and latest allocation situation on the internet so that related manufacturing enterprises can timely master the latest resource use conditions.

3) Users on each platform should have the communication channel on the platform.

4) Manufacturing resources owned by each member unit are covert to each non-member unit, so the platform should have good security mechanism, including validity verification for log-in and restricted use of system management function.

Development of the platform system should be able to promote sharing of various kinds of information in the group, timely know resource conditions of each enterprise, improve work efficiency and enhance harmony and consistency of decision-making.

RESULTS AND DISCUSSION

For the convenience of comparison, the instance data are obtained here through the experimental method and their parameters are the same as that in the paper. $L=100$ (the maximal length of tracks is 100), $m=50$ (50 instance groups), $n=50$ (each instance group contains 50 instances at most). The detailed generation process of instances is: implement the following iteration as to each instance group: 1) different instance group contains different number of instances, select one number that obeys the normal distribution. 2) Randomly select the location of the left endpoint of the instance from number 1 to 99, suppose it is i. 3) Randomly select one number from 1 to $100-i$ as the length of the instance. Return to step 2) after step 3) ends, and the iteration goes on until the number of generated instance is the same as that in step 1). An input instance is obtained in this way and it will be transformed into a 50-partite adjacent matrix.

The matrix is the input of the above-mentioned algorithm. After the implementation of the algorithm, edges that can be merged will be merged and the length sum of edge sets will be reduced greatly. The following is the experimental data.

Table 1 Net Length Comparisons

Algorithm	Improved Prim Algorithm		MST Algorithm	
Order of instances	Before the algorithm	After the algorithm	Before the algorithm	After the algorithm
1	3371	230	3973	234
2	3522	246	3753	253
3	4324	248	3918	246
4	3769	244	3959	241
5	4246	295	3499	225
6	3424	233	3797	230
7	3891	233	3410	239
8	4283	243	3509	233
9	4506	237	3786	252
10	3979	242	4062	235
Weight sum	39316	2453	37674	2388

It can be seen from the comparison in Table 1 that, some data obtained here is even poorer than that of MST Algorithm, but most are better. The reason is that the moving of ants during the optimization process is stochastic. In a word, the experimental result is better.

CONCLUSION

This paper combined ant colony optimization algorithm with classical graph theory algorithms, and proposed the ant colony algorithm based on the Prim algorithm for the segmentation design of FPGA. It aims to find a special maximal forest. On one side, it has adopted the greedy criterion in Prim algorithms, and on the other side, it has used the ant colony algorithm to increase the randomness for the edge selection, so it can also be called a Prim algorithm with ant colony optimization idea. Experimental data show the algorithm can achieve satisfactory results.

Acknowledgements
This work was supported by the Science and Technology Research Foundation of Education Bureau of Hubei Province (No. B2013064).

REFERENCES

[1] Olawale Yakubu Adisa. *Construction Management and Economics*, **2010**, 28(5): 509-526
[2] Shane Jennifer S. *Journal of Management in Engineering*, **2009**, 25(4): 221-229
[3] M. Dorigo, L M Gambardella. *IEEE Transaction on Evolutionary computations*, **1997**, 1(1): 53-66
[4] M. Dorigo, A. Colorni, V. Maniezzo. *IEEE Transactions on Systems, Man, and Cybernetics*, **1996**, 26(1): 29-41
[5] L M Gambardella, E.D Tailard, M. Dorigo. *Journal of the operational research society*, **1999**, 50(2), 1167-1176
[6] A. Colorni, M. Dorigo, V. Maniezzo. *Belgian Journal of Operations Research, Statistics and computer Science*, **1994**, 34: 39-53
[7] D Costa, A Hertz. *Journal of the Operational Research Society*, **1997**, 48: 295-305