Efficient Simulated Annealing-Based Placement Algorithm for Island Style FPGAs

Runxiao Shi, Lan Ma, and Takahiro Watanabe

***Abstract*—This paper proposes a group-based very fast simulated annealing (GB-VFSA) algorithm to achieve higher-quality placement in a shorter CPU time. We first introduce a temperature-dependent perturbation model based on Cauchy distribution to generate new solutions. Then, we put high-connected blocks into one group and use a group as a unit for placement. In order to avoid premature convergence of the algorithm, multiple potential solutions are used to search the solution space at the same time. The idea "pheromone" which comes from ant colony optimization is used to realize the communication between multiple potential solutions. Experimental results using MCNC beachmarks show that GB-VFSA achieved 23% reduction in CPU time and 3.6% improvement in maximal time delay over traditional simulating annealing algorithm.**

***Index Terms*—Placement, simulating annealing, island style FPGAs, block group, potential solution group.**

I. Introduction

Field-programmable gate arrays (FPGAs) embrace enormous advantages such as low non-recurring engineering costs, shorter design cycle, high fault tolerance and outstanding parallel performance [1], which enables FPGAs to become a popular research topic in microelectronics and computer science. Also, the development of FPGAs-based high-level synthesis tools, such as Quartus II or Vivado, has brought great convenience to the FPGAs-based design. Those high-level synthesis tools have drastically reduced the R&D cycle without affecting the performance of the circuit.

Currently, the development of FPGAs shows the following trends: 1) The structure of FPGAs become more complex. Due to the development of IP core technology, more and more FPGA manufacturers embed IP cores (multipliers, block RAMs, phase-locked loops, etc.) into FPGA to help realize more complex system-on-chip design. 2) Logic resources included on the FPGA continue to increase. Introduced in 2013 by Xilinx, Virtex UltraScale series FPGAs utilize a 20nm TSMC process with up to four million logic cells. 3) FPGAs become more and more energy efficient by lowering core voltage, using gating clock technology and reducing leakage.

Placement is a critical stage in the FPGA design flow. It determines how circuit blocks (including I/O blocks and logic blocks) are mapped on the physical locations of FPGAs. The quality of placement directly influences the succeeding routing stage and the overall performance of the whole circuit. Placement is a classical NP-hard problem. As we mentioned before, the number of logic blocks on the FPGAs has experienced significant increase recently, which expands search space of placement algorithm dramatically. The larger search space requires more time for placement. The low efficiency of the placement algorithm extends the design cycle and hinders the further development of FPGAs [2]. In this paper, we propose a much more time-efficient placement algorithm without sacrificing the placement quality based on the classical simulated annealing algorithm.

Manuscript received August 15, 2018; revised September 18, 2018.

The authors are with the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu-shi, CO 808-0135 Japan (e-mail: srxssinseu@toki.waseda.jp, mimolan@toki.waseda.jp, watt@waseda.jp).

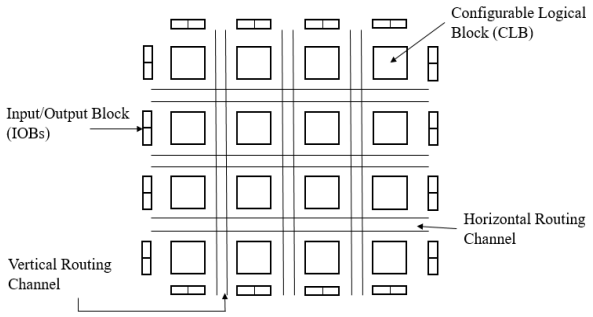


Fig. 1. General architecture of island style FPGA.

A. FPGAs Structure

FPGAs are composed of configurable logic unit blocks (CLBs), input and output blocks (IOBs) and programmable interconnect resources. CLB consists of several basic logic elements (BLEs) and local interconnects. BLE is made up by look-up table (LUT), multiplexer (MUX) and trigger. According to the different interconnection structure, FPGAs are classified into a variety of categories, such as island style FPGAs, line-based FPGAs and hierarchical FPGAs. Among them, the island style FPGA is the most popular one, and its structure is shown in Fig. 1. IOBs are placed at the edge of the FPGAs. CLBs are surrounded by the horizontal and vertical routing channels.

B. FPGAs Placement Problem Formulation

Placement problem on FPGAs is formulated as follows: Given a block set (including CLBs and IOBs) B= $\{b_1, b_2, b_3, \ldots, b_n\}$ and a net set N= $\{n_1, n_2, n_3, \ldots, n_r\}$, a block b_i of B belongs to at least one net n_i of N, T= $\{t_1, t_2, t_3, \ldots, t_p\}$ is a set of p empty slots ($p \geq n$), slot t_i's position is represented by a coordinate (x_i, y_i) where a block is assigned. The placement problem of FPGA is to assign each block $b_i \in B$ to a unique slot $t_i \in T$ to meet some optimization objectives such as routing rate, timing performance and power consumption.

Therefore, there are C_p^n possible layout cases, which form the solution space of the layout problem. Fig. 2 shows a

doi: 10.18178/ijmlc.2018.8.6.743

simple example of placement and routing results.

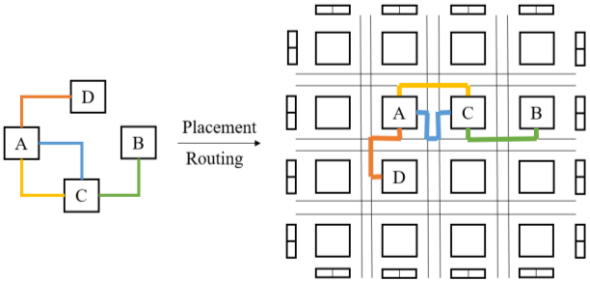


Fig. 2. Example of placement and routing results on island style FPGA.

C. Prior Work

Many algorithms have been proposed to solve the placement problem on FPGAs with different structures. The final goal of these algorithms is to save CPU time to do the placement without sacrificing the quality of the placement. Currently, FPGA placement algorithm can be divided into the following four categories: simulated annealing-based placement algorithm [3], partitioning-based placement algorithms [4], quadratic-based placement algorithms [5] and other heuristic placement algorithms such as ant colony optimization [7], genetic algorithm [6], quantum evolutionary algorithm [8]. Although these four kinds of algorithms have advantages and disadvantages, the simulated annealing algorithm has become the most mainstream FPGAs placement algorithm due to its good placement quality and open objective function.

Many researches have been carried out around simulated annealing-based placement. In [3], the authors developed a CAD tool called VPR which can execute packing, placement and routing for FPGAs. In the placement stage, VPR uses simulated annealing and can take wire length and time delay into consideration. Based on VPR, [9] attempted to achieve better placement results by using dynamically adaptive stochastic tunneling to help the potential solution jumped out of the local optimal solution. Vorwerk *et al.* [10] discussed several strategies to enable the swaps during the annealing more directional, allowing the converge much quicker without decreasing in the placement quality. [6] tried to combine the traditional SA algorithm with some heuristic algorithms, where genetic algorithm was used to approach global optimal solution. In order to save more CPU time, Matthew *et al.* [2] used parallelism calculation into FPGAs placement.

The rest of this paper is organized as follows. Section II describes a simulated annealing algorithm and its drawbacks. In the Section III, we introduce a new perturbation model based on Cauchy distribution to generate new solutions and improve the traditional SA. In Section IV, we propose GB-VFSA (Group-based Very Fast Simulating Annealing) by introducing block group and potential solution group. In Section V, we apply the GB-VFSA to the standard placement problem and demonstrate the superiority of our proposed algorithm. Finally, section VI concludes this paper.

II. Simulated Annealing (SA)

SA algorithm mimics the motion of particles during the annealing process. In the process of metal annealing, the internal energy is minimized by warming the metal to a sufficiently high temperature and allowing it to cool slowly. When heated, the particles become disordered as the temperature increases, thereby increasing the internal energy. At cooling process, the particles gradually become more ordered (ie, the state of low internal energy). At each temperature, the system can approximately reach equilibrium, and finally reach the ground state with minimal energy at room temperature.

In the FPGAs placement problem, the particles represent the logical blocks that need to be placed on the FPGAs. The random motion of particles during annealing corresponds to the random location exchange of these logical blocks during the placement process. The energy corresponds to the value of the cost function. The lowest energy ground state corresponds to the best placement result with the lowest cost we need to find.

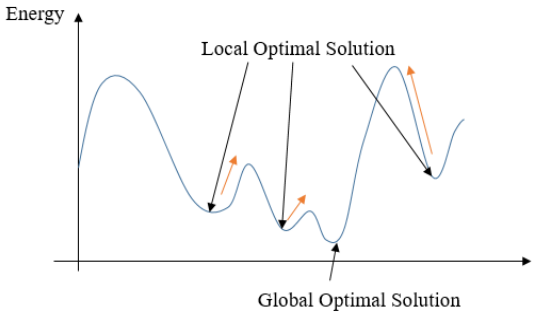


Fig. 3. Local optimal solution and global optimal solution.

A. Guidelines

The process of SA to find the optimal solution is equal to find the minimum value of the cost function in the solution space. If there are p slots on FPGA and n blocks need to be placed, the solution space will be C_p^n. As shown in Fig. 3, sometimes, our potential solution might be trapped in the local optimal solution, in order to jump out of the local optimal solution, we need to accept some aggravating solutions (red arrow in Fig. 3) according to the Metropolis Guidelines (1),

(1)

where *i* is the current solution, *j* is the new solution generated by perturbation, *P* is the probability we accept the new solution, C is the cost function, *T* is the current temperature. From the Metropolis Guidelines, we can find that the worse the new result is, the less likely we are to accept the new result. Also, in the low temperature condition, the probability we accept worse solution is very low.

B. Swap Distance Limit

Swap distance limit D_{limt} [3] is an important parameter in the simulated annealing algorithm. D_{limit} determines the upper distance limit when two logic blocks try to exchange. Fig. 4 shows us the swap area when D_{limt}=1. At initialization, D_{limit} is set to the width of the entire FPGAs. During the annealing process, D_{limit} will change according to (2).

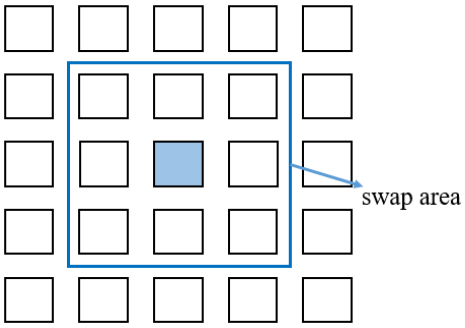


Fig. 4. Example of swap area for logical blocks.

(2)

where α represents the swap acceptance rate at specific temperature. By (2), we can keep the acceptance rate around 0.44. Because when α is over 0.44, D_{limit} will increase, the longer distance swap will be less likely to accept, then acceptance rate will decrease and go back to 0.44.

C. Adaptive Annealing Schedule

We will lower the temperature according to (3). The value of γ is determined by the adaptive annealing schedule shown in Table I.

(3)

TABLE I: ADAPTIVE ANNEALING SC

Acceptance Rate	*Update ent γ*
α >0.98	
0.8<α<0.96	
0.5<α≤0.8	
α≤0.15	

D. Weakness

Although the SA-based algorithm od poten find the global optimal solution, it re large nu swap operations and frequent calcu f cost fu which takes a lot of time and decreas fficiency the reason why we modify the alg nto the simulated annealing propos algorithm is really sensitive to the value of nt parameters, in research [11], the authors found tha ial temperature affects not only the CPU time bu the placement quality. In [9], the authors change the v α in the adaptive schedule and the critical time de nges obviously in final results. Finally, SA algorithm h sk of falling into the local optimal solution, especia n the solution space is relatively rough and the ptimal solution is close to the global optimal s . This phenomenon was called freezing proble use several potential solutions to search the solution space to decrease the probability of freezing problem as described in Section IV.

III. VERY FAST SIMULATING ANNEALING

In the classical SA algorithms, the acceptance probability of a worse solution is given by the Gibbs distribution. The existing applications and numerical examples show that the very fast simulated annealing algorithm (VFSA) [13], [17] has high computational efficiency, mainly because the perturbation model that generates new solutions based on a temperature dependent Cauchy distribution. Therefore, we modify the algorithm and use the same perturbation model.

A. Perturbation Model

The new solution in VFSA is generated by perturbing the current solution. We use an approximate Cauchy distribution to generate the new solution [13]. This process can be described by equation (4) and (5).

(4)

(5)

where, x_i^{new} represents the abscissa of the new position of logic block *i*, C is a parameter calculated by equation (5), *W* is the range of x_i, which is the width of FPGAs, *T* is the temperature of current circle, sgn() is a symbol function, u is an uniformly distributed random number in [0,1].

The advantage of using a temperature-dependent Cauchy distribution is that it enables our algorithm to perform a wide range searches at high temperature and search only near the current solution at low tem res. o, the Cauchy tribution has a flat tail [13 which ma it eas o jump ou l optimal soluti This imp emen eeds u the con of SA meth .

B. Accep robabi

The deriva accepted probabi is the generalized C tribution. Refer to [14] a about detailed deriv rocess. Equation (6) gives tance probability.

where h positive real number a cost differe in (ur algorithm, we hen *h* t to on (6) can degenera n (1). As a result from a mathematical a probability o classical SA can be seen as d case of VFSA eptance probability.

C. Cost Function

Wiring cost and timin e two ma angle o evaluat the result of placem pendin on goal optimization, the placeme be divided into wire length driven placement, net ti iven placement and path timing driven placement. In search, we choose path timing driven placement, which both wiring cost timing cos tion. The can be calcula by equati (7).

(7)

The a param we u to balance the weig of wiring t and tim st, pre_ g_cost represen the wiring cost of the current solution, Δwiring_cost denotes the difference between the current cost and new cost. We use half perimeter wire length (HPWL) to estimate the wire length of the placement. We use (8) to calculate the wire cost:

(8)

where n_i is the i-th net in N. bb_x and bb_y are the x- and y-dimensions of the bounding box of net n_i. $avg\_chanx\_W(n)$ and $avg\_chany\_W(n)$ are the average channel width (number of tracks) of the bounding box of the net n_i in the x and y directions. $q(n)$ is the compensation factor, its value will increase from 1 (when the number of terminal is 3 or fewer than 3) to 2.79 (when the nets have 50 terminals) when the terminals of the nets increase.

We calculate the timing cost according to the equation (9). In equation (9), $\Delta x_{ij}(\Delta y_{ij})$ is the length of x-(y-) direction between block i and block j. The timing cost is the sum of delay of all the connections in the circuit. A criticality (i, j) can be calculated by delay margin.

(9)

IV. Group-Based VFSA

In order to get a better placement result in a shorter period of time, based on the VFSA, we propose a group-based very fast simulated annealing algorithm (GB-VFSA). We have two kinds of groups in our proposed algorithm. The first is the logical block group. We put highly connected blocks into one group and use the group instead of a single block as the basic unit for the placement. The second group is the potential solution group. In traditional VFSA algorithms, we only have one potential solution, which makes the algorithm easy to be trapped into the local optimal solution. In our algorithm, several potential solutions will search the solution space at the same time to avoid premature convergence of the algorithm.

A. Block Group

The placement results produced by VFSA show that, if two logic blocks are connected neatly, they are often placed closely to save more wiring resources. Therefore, before starting the placement process, we put the closely connected logic blocks into one logical block in the same group. Logical blocks in the same group are easier to be placed adjacent to each other. This can speed up the convergence speed of our algorithm.

1) Grouping Basis: We need to collect two kinds of information as the basis for grouping. First, for each block, how many other blocks it has connection with. We call this information "degree" of the block. Second, for the two blocks connected with each other, how many connections they have. We call this information "weight" between two blocks. For example, in Fig. 5, degree(C) and weight(C, D) is 2.

2) Grouping Strategy: After we get these two information we will divide the blocks into two categories, central block and common block. Central block represents the block with high "degree" value. In each group, there is only one central block and other blocks are the common blocks. The common blocks in one specific group are the blocks having high "weight" with the central block. For example, if the five blocks in Fig. 5 are put into one group, block C will be the central block and the other four blocks will be the common blocks.

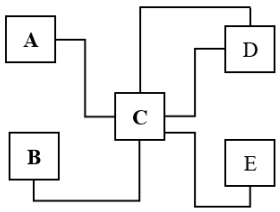
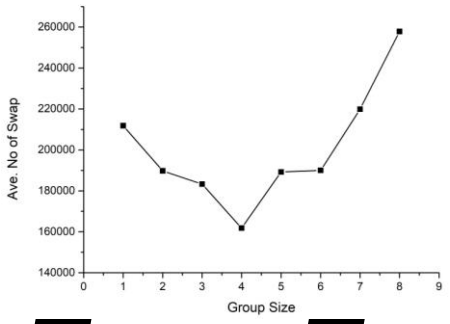
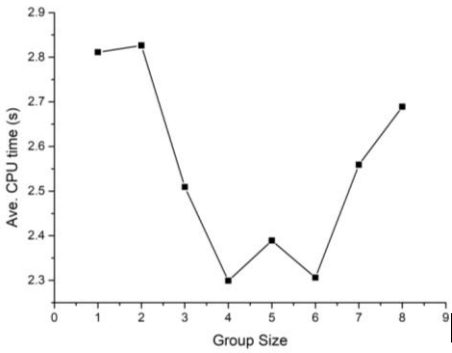


Fig. 5. Example of "degree" and "weight".

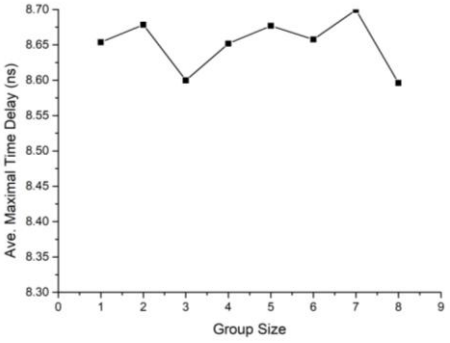
3) Group Size: The size of a group is a very important parameter in our algorithm. If the group size is too small, the new algorithm will not be much different from the VFSA, and we cannot achieve the goal of accelerating algorithm convergence to save more CPU time. If the grouping is too large, we will put together logical blocks that should not be close to each other. Therefore, we need to spend extra time dispersing these logical blocks. Here, we explore the impact of group size on the number of swap, CPU time, and maximum time delay. From Fig. 6(a), we can see that the number of swap decreases as the size of the group increases, and then increases. The number of swap also affected the CPU time to a certain extent. The trend of CPU time changed the same with the number of swap (Fig. 6(b)). The impact of group size on the maximum time delay is not obvious (Fig. 6(c)). Considering the above three perspectives, in our algorithm, the size of the group is set to 4.



(a)



(b)



(c)

Fig. 6. The effect of block group size on swap times, CPU time and maximal time delay.

4) Improved Initial Placement: Since we have already got the "degree" of each block, we can put the block with high "degree" in the center of FPGAs during the process we generate the initial placement. The central blocks having higher "degree" with their 3 common blocks will choose location first. The greater "degree" a block has in the circuit, the more centric it will be implemented on the FPGAs to save more wiring resources. Also, initial temperature is inversely proportional to the quality of the initial placement, the higher the quality is, the lower the temperature will be. Since we make the initial placement much more directional, we hope to decrease the initial temperature may be decreased and CPU time can be reduced.

5) Swap Operation: During the whole placement process, we change the location of groups by swapping the central blocks. Common blocks will also change their location when their central blocks change their location. They will be prioritized in the closest empty CLBs to their central blocks. Fig. 7 shows an example of swap operation. If central block A has changed its location, the common blocks (D and C) in this group will also change their location. They will consider the red empty CLBs first since the distance between A and red CLBs is one. If all the red CLBs have been occupied, then, they will think about the green CLBs (distance is 2).

B. Potential Solutions Group

In the traditional SA algorithm, we have only one potential solution. This potential solution eventually finds the global optimal solution by randomly exchanging the location of the logic blocks. However, as we mentioned before, the biggest problem with a single potential solution is that it is very likely to fall into the local optimal solutions. Restarting algorithm is one of the effective ways to overcome the local optimal solution [16], but the cost of restarting algorithm is usually higher. Particle swarm optimization (PSO) [19] and ant colony optimization (ACO) [18] can effectively overcome the local optimal solution. The most prominent feature of PSO and ASO is that multiple potential solutions search the solution space at the same time. Therefore, we introduce a number of potential solutions to VFSA. Several potential solutions are also used to optimize the layout at the same time, so as to reduce the possibility of falling into the local optimal solution.

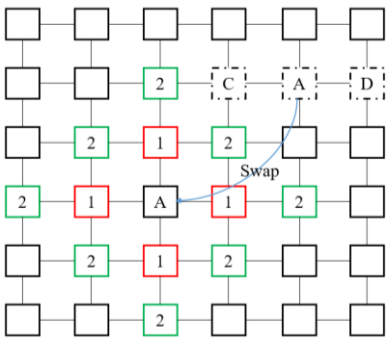


Fig. 7. Example of swap operation and distance between CLBs.

We refer to these potential solutions as potential solution group. Intuitively, due to the introduction of multiple potential solutions, the algorithm's CPU time must increase significantly. However, we still have the opportunity to obtain a better solution in a shorter period of time. First, in the previous section, we greatly improved the convergence speed of the algorithm by grouping the logic blocks and improve initial placement. At the same time, when multiple solutions are used to search the solution space, there exists a mechanism for mutual communication between these potential solutions, which makes the convergence more directional and further improves the operation speed of the algorithm. The pseudocode of GB-VFSA using both block group and potential solution groups are shown in Fig. 8.

1) *Pheromone Update:* A pheromone is a classic variable in ant colony optimization that realizes communication between multiple potential solutions. As a result, multiple potential solutions can share each other's with their excellent swap. ACO stems from the phenomenon of ants foraging in nature. Scientists have discovered that ants automatically gather on the shortest path between food and their nests. This is because the shortest path has the highest concentration of pheromones, and the ants have a higher probability of selecting paths with higher concentrations of pheromones [18]. Pheromones have the following two characteristics: First, the pheromones will continue to volatilize, the concentration will continue to decline. Second, when different ants choose the same path, the pheromone concentration of that path will increase.

```
Begin Algorithm

S=ImprovedInitialPlacement();
T=InitialTemperature();
R_limit=InitialR_limit();

while(ExitCriterion()=False)  /* when the temperature is low enough, the
                                 search will end */
{  /* outside loop starts*/
  while(InnerLoopCriterion=False)
  {/* inner loop starts*/
    for ps=1 to n { /* n denotes the number of potential solutions */
        S^ps_new=GenerateViaSwap(S^ps_current, R_limit);
        ΔC=Cost(S^ps_new)-Cost(S^ps_current);
        if(ΔC≤ 0){ S^ps_current = S^ps_new;}
        else(ΔC>0){
                R=Random(0,1);
        if(R<η×P_Gibbs+(1-η) ×P_Pheromone){S^ps_current = S^ps_new;}
          }/* all the potential solutions have update their placement*/
  } /*inner loop ends*/

    RecordBestPlacement(); /*choose the optimal solution
                             from multiple potential solutions*/
    UpdatePheromone(); /*optimal solution update pheromone*/
    T=UpdateTemperature(); /*update the annealing temperature*/
    R_limit=UpdateR_limit(); /*update maximum swap distance*/
} /* outside loop ends*/

End Algorithm
```

Fig. 8. Pseudocode of GB-VFSA.

In the model of the placement problem, ants are multiple potential solutions. The pheromone corresponds to the probability that we put logical block *i* on CLB *j*. If multiple potential solutions choose to place logical block *i* on CLB *j*

during the search process, the pheromone concentration of that path will increase.

In the model of the placement problem, ants are multiple potential solutions. The pheromone corresponds to the probability that we put logical block *i* on CLB *j*. If multiple potential solutions choose to place logical block *i* on CLB *j* during the search process, the pheromone concentration corresponding to this layout will continue to increase.

We introduce pheromones in VFSA to enable the communication between multiple potential solutions. In GB-VFSA process, at a given temperature, potential solutions are constantly gaining new solutions by perturbations. After the cycle at specific temperature is complete, these potential solutions will communicate with each other by updating the pheromones. Only potential solutions with the highest layout quality can update pheromones. Through this process, high-quality exchanges are executed between potential solutions. The update equation is shown in (10)

(10)

where τ_{ij} represents the pheromone we put block *i* on CLB *j*. $\Delta\tau_{ij}^{best}$ denotes that only the best potential solution can update the pheromone. ρ is the evaporation rate. $\Delta\tau_{ij}^{best}$ is calculated by (11)

(11)

where cost is the value of cost function defined in (8) and (9) and *Q* is the intensity of pheromone.

2) *Acceptance Probability*: In VFSA, we accept the deterioration solution according to certain probability (Equation (6)) so as to enhance the climbing ability of the algorithm. In our proposed GB-VFSA, we modified the acceptance probability formula. The new acceptance probability (Equation (12)) takes into account both the acceptance probability derived from the generalized Gibbs distribution and the exchange of information between multiple potential solutions.

(12)

where η is a parameter we used to balance the weight of probability derived from Gibbs distribution and pheromone. The calculation of $P_{\text{Pheromone}}$ is calculated by (13).

(13)

where ℓ denotes CLB where we have e[illegible] the process these potential solutions search [illegible] solution space. The new probability acceptance equa[illegible]l enable the transition from a single potential solution[illegible] space to multiple potential search space. P_{Gibbs} in [illegible]flects the individual wisdom of a single solution. [illegible] $P_{Pheromone}$ reflects the collaboration among multiple p[illegible] solutions. With pheromone updates, excellent sw[illegible]be passed between multiple potential solutions, ma[illegible]e algorithm converge to better solutions.

TABLE II: COMPARISON BETWEEN SA, VFSA AND GB-VFSA

Beachmarks	SA			VFSA			GB-VFS		
	No. of Swap	*CPU time*	*Max delay*	*No. of Swap*	*CPU time*	*Max delay*	*No. of Swap*	*CPU ti*	*elay*
tseng	1x	1x	1x	0.80670x	0.808141x	1.04014x	0.676561x	0.74691	587x
apex4	1x	1x	1x	0.58088x	0.876731x	0.990551x	0.577745x	0.7834	655x
clma	1x	1x	1x	0.64800x	0.933398x	1.036542x	0.545855x	0.7853	3829x
bigkey	1x	1x	1x	0.72611x	0.802292x	1.119234x	0.729685x	0.857	1747x
ex1010	1x	1x	1x	0.45171x	0.88504x	1.135533x	0.399435		
s298	1x	1x	1x	0.55292x	0.764472x	1.078563x	0.444863x	0.606568x	0.930411x
pdc	1x	1x	1x	0.80443x	0.848022x	1.026528x	0.49847x	0.84952x	1.026528x
apex2	1x	1x	1x	0.64267x	0.708483x	1.061198x	0.457214x	0.763784x	0.94431x
elliptic	1x	1x	1x	0.67157x	0.888713x	1.078007x	0.671119x	0.80461x	1.026857x
frisc	1x	1x	1x	0.74632x	0.837304x	1.00213x	0.57185x	0.675257x	0.922174x
Average	1x	1x	1x	0.683087x	0.86991x	1.050323x	0.586375x	0.773352x	0.965121x

TABLE III: COMPARISON BETWEEN VFSA AND GB-VFSA

Beachmarks	VFSA			GB-VFSA		
	No. of Swap	*CPU time*	*Max delay*	*No. of Swap*	*CPU time*	*Max delay*
tseng	1x	1x	1x	0.838679x	0.924238x	0.897559x
apex4	1x	1x	1x	0.994605x	0.893633x	0.955579x
clma	1x	1x	1x	0.842371x	0.841421x	0.925027x
bigkey	1x	1x	1x	1.004917x	1.068231x	0.850355x
ex1010	1x	1x	1x	0.884273x	0.857015x	0.890138x
s298	1x	1x	1x	0.804567x	0.793446x	0.862639x
pdc	1x	1x	1x	0.619658x	1.001766x	1.000102x
apex2	1x	1x	1x	0.711428x	1.078055x	0.889853x
elliptic	1x	1x	1x	0.999329x	0.905366x	0.952551x
frisc	1x	1x	1x	0.766221x	0.806465x	0.920214x
Average	1x	1x	1x	0.858419x	0.889002x	0.918892x

V. EXPERIMENT RESULTS

To evaluate the performance of our proposed GB-VFAS, several experiments are carried out to compare the number of swap during the placement, maximal tim[illegible] of circuit after placement and routing and CPU time[illegible]ement. The performance is measured by using ten M[illegible] All the experiments are done under a M[illegible] geon with

Intel i7-4700MQ CPU and 8GB DDR3L RAM.

Table II shows us the results between the traditional SA, VFSA. We can see that compared to SA, VFSA has seen significant improvements in both switching times and CPU time, improved about 32% and 13% respectively. However, there is a slight deterioration in the quality of the placement, with a maximum time delay deteriorating about 5%. VFSA controls the maximum swap distance, making the maximum swap distance decrease as the temperature drops, thus improving the quality of swap, making the convergence much more directional and saving much computing time.

From Table II and Table III, we can see that compared with SA and VFSA, our proposed GB-VFSA has improved in all three aspects: number of swap (41% and 14%), CPU time (23% and 11%) and maximal delay (3.6% and 8.2%). In less CPU time, we can achieve higher-quality placement. This result is reasonable, because GB-VFSA optimizes the algorithm in three aspects. We increase the convergence rate of the algorithm through the block group and using better initial results. We also improve the ability of jumping out of the local optimal solution by using potential solution groups.

VI. Conclusion

We propose a novel placement algorithm based on the simulated annealing, to achieve the improvement in two aspects: placement quality and CPU time. We first introduce the new perturbation model that generates new solutions based on Cauchy distribution, and achieve the first increase in the speed of the algorithm. Then, we put high-connected blocks into a group and use block group as the unit to do the placement. We also improve the quality of initial placement. At the same time, we introduce multiple potential solutions into the algorithm, and realize the communication between several potential solutions through the update of pheromone, which improves algorithm's ability to get rid of the local optimal solution and finally improve the time delay.

References

[1] J. Timothy *et al*., "Reconfigurable computing: architectures and design methods," *IEE Proceedings-Computers and Digital Techniques*, vol. 152, no. 2, pp. 193-207, 2005.

[2] A. Matthew, J. G. Steffan, and V. Betz, "Speeding up FPGA placement: Parallel algorithms and methods," in *Proc. 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*, IEEE, 2014.

[3] B.Vaughn and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," presented at International Workshop on Field Programmable Logic and Applications, Springer, Berlin, Heidelberg, 1997.

[4] M. Pongstorn, C. Ababei, and K. Bazargan, "Timing-driven partitioning-based placement for island style FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, 2005, pp. 395-406.

[5] Y. H. Xu and M. A. S. Khalid, "QPF: Efficient quadratic placement for FPGAs," in *Proc. International Conference on Field Programmable Logic and Applications*, 2005.

[6] Y. Meng, A. E. A. Almaini, and P. J. Wang, "FPGA placement optimization by two-step unified genetic algorithm and simulated annealing algorithm," *Journal of Electronics*, vol. 23, no. 4, pp. 632-636, 2006.

[7] W. Y. Xu, K. J. Xu, and X. M. Xu, "A novel placement algorithm for symmetrical FPGA," in *Proc. ASICON'07*, IEEE, 2007.

[8] G. Xiao *et al*., "Fast FPGA placement algorithm using quantum genetic algorithm with simulated annealing," in *Proc. ASICON'09*, IEEE, 2009.

[9] M. J. Lin and J. Wawrzynek, "Improving FPGA placement with dynamically adaptive stochastic tunneling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1858-1869, 2010.

[10] V. Kristofer, Andrew Kennings, and Jonathan W. Greene, "Improving simulated annealing-based FPGA placement with directed moves," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 2, pp. 179-192, 2009.

[11] R. Hermida *et al*., "Placement by thermodynamic simulated annealing," *Physics Letters A*, vol. 317, no. 5, pp. 415-423, 2003.

[12] Z. Q. Li and H. A. Scheraga, "Monte Carlo-minimization approach to the multiple-minima problem in protein folding," Proceedings of the National Academy of Sciences, vol. 84, no. 19, pp. 6611-6615, 1987.

[13] I. Lester, "Very fast simulated re-annealing," *Mathematical and Computer Modeling*, vol. 12, no. 8, pp. 967-973, 1989.

[14] T. Constantino, "Possible generalization of Boltzmann-Gibbs statistics," *Journal of Statistical Physics*, vol. 52, no. 1, pp. 479-487, 1988.

[15] J. P. P. Thadeu, "Traveling salesman problem and Tsallis statistics," *Physical Review E*, vol. 51, no. 1, 1995.

[16] A. H. Bernardo, R. M. Lukose, and T. Hogg, "An economics approach to hard computational problems," *Science*, vol. 275, no. 5296, pp. 51-54, 1997.

[17] I. Lester and B. Rosen, "Genetic algorithms and very fast simulated reannealing: A comparison," *Mathematical and Computer Modelling*, vol. 16, no. 11, pp. 87-100, 1992.

[18] D. Marco, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28-39, 2006.

[19] K. James, "Particle swarm optimization," *Encyclopedia of Machine Learning*, Springer US, pp. 760-766, 2011.

Runxiao Shi was born in Nanjing, China on August 30, 1995. He received his B.E. degree from Southeast University (Nanjing, China) in 2017. He is currently working toward the M.E. degree in Graduate School of Information, Production and Systems, Waseda University, Kitakyushu-shi, Japan under the guidance of Prof. Takahiro Watanabe. In 2016, he was chosen as the scholarship student of IPS, Waseda University. His research interests include computer aided design of FPGAs, especially placement and routing algorithm. He is also carrying out some research on the interposer-based multi-FPGAs.

Lan Ma was born in Suzhou, China on February 9, 1995. She received her B.E degree from Southeast University, Nanjing, China in 2017. Currently, she is working toward M.E degree in Graduate School of Information, Production and Systems, Waseda University, Kitakyshu, Japan under the guidance of Prof. Takahiro Watanabe. She is carrying out her research on placement and scheduling algorithm on the reconfigurable systems.

Takahiro Watanabe was born in Ube, Japan in October 1950. He received his B.E. and M.E. degree in electrical engineering from Yamaguchi University (), and the Dr. Eng. From Tohoku University. In 1979, he joined R&D Center of TOSHIBA Corp., where he worked in the field of LSI design automation. In August 1990, he joined Yamaguchi University, and in April 2003, he moved to Waseda University, Graduate School of Information, Production and Systems. His current research interests are EDA algorithm, microprocessor, NoC, FPGA and their applications. He is a member of IEICE, IPSJ and IEEE.