A Parallel Multistage Metaheuristic Algorithm for VLSI Floorplanning

Takayoshi Shimazu Shin'ichi Wakabayashi Shinobu Nagayama
Faculty of Information Sciences, Hiroshima City University
3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima 731-3194, Japan

Abstract This paper proposes a parallel multistage metaheuristic algorithm to solve the floorplanning problem in VLSI layout design, in which the sequence-pair representation is adopted as the coding scheme of each chromosome. The proposed method consists of three stages, and the first and second ones are based on genetic algorithm, and the last one is based on tabu search. As the stage proceeds, a set of solutions are gradually refined. The proposed method is a parallel algorithm, running on a PC cluster. The proposed parallel floorplanning algorithm was implemented using the MPI (Message Passing Interface) library on a PC cluster. Experimental results show the effectiveness of the proposed algorithm.

1 Introduction

The floorplanning problem is an essential design step in VLSI layout design, and it determines the coarse placement of a given set of modules [9]. This problem is known to be difficult and time-consuming to solve, since the number of possible placements of modules increases exponentially as the number of modules increases. Many approaches have been proposed to solve the problem in a practical computation time [9]. Among those approaches, metaheuristics such as genetic algorithms [5], simulated annealing, and tabu search [10] are known to be effective to produce a good solution for the problem.

In this paper, we propose a parallel metaheuristic algorithm to solve the floorplanning problem in VLSI layout design. The proposed method is mainly based on genetic algorithm (GA) [3], and to shorten the execution time, we introduced parallel processing into it. Furthermore, to strengthen the ability of local search, tabu search [4] was combined with GA. The proposed method consists of three stages, and the first and second ones are based on genetic algorithm, and the last one is based on tabu search. As the stage proceeds, a set of solutions are gradually refined.

The proposed method is a parallel algorithm, running on a PC cluster. We implemented the proposed parallel metaheuristic algorithm with the MPI (Message Passing Interface) library [8] on a PC cluster consisting of 14 PCs. Experimental results show the effectiveness of the proposed method.

2 Preliminaries

2.1 Floorplanning Problem

Floorplanning is a generalization of the placement problem in VLSI building block layout, and it determines the coarse placement for a given set of modules [9].

A *module* $m_i \in M (0 \leq i < n)$ is a subcircuit whose shape is a rectangle with a given height and width in real number. A *floorplan* of a set of modules is a non-overlapping placement of given modules. The minimum bounding rectangle of a floorplan is called the *chip*. Each module has several terminals on its boundary, which are to be interconnected with wires. Interconnections among terminals are specified by a netlist. The floorplanning problem is to find a floorplan of M onto a chip with the minimum area and wire length. This problem is known to be NP-hard, and hence good heuristics are generally solicited.

2.2 Representation of Chromosomes

We adopt sequence-pair [6] to represent each individual of the proposed floorplanning method. Let $x^i (0 \leq i < P)$ be i-th individual in the population. x^i represents a candidate solution of the floorplanning problem, and is denoted by $x^i = (\Gamma+^i, \Gamma-^i, \Theta^i)$. $(\Gamma+^i, \Gamma-^i)$ is the sequence-pair whose respective elements are composed of module names, that is, $(\gamma+^i_0, \gamma+^i_1, \cdots, \gamma+^i_j, \cdots, \gamma+^i_{n-1})$ and $(\gamma-^i_n, \gamma-^i_{n+1}, \cdots, \gamma-^i_j, \cdots, \gamma-^i_{2n-1})$. Θ^i shows orientation of modules, that is, $\Theta^i_j = \{0, ..., 7\}, (2n \leq j < 3n)$. Each position of an individual is called the *locus* of chromosome.

2.3 Objective Functions

The objective of the floorplanning problem is to minimize the total wire length as well as the chip area. In the proposed algorithm, any solution of the problem is represented as a sequence-pair. The chip area is calculated from a sequence-pair by constructing the H/V constraint graphs, and finding the longest paths on H/V constraint graphs [6]. The orientation of modules, which are specified by Θ^i, are also considered. When calculating the chip area, coordinates of each module can be also determined. From the module placement,

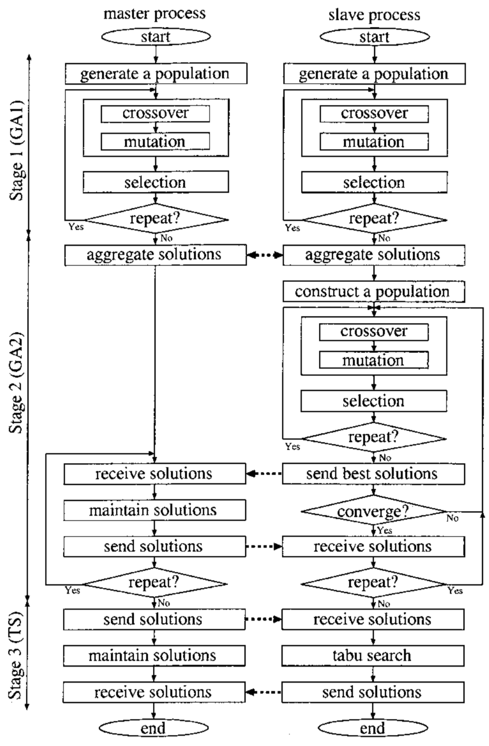

Figure 1: Flow chart of the algorithm.

the total wire length is calculated with the half perimeter of the net bounding box.

In this paper, we treat the floorplanning problem as a multi-objective optimization problem, and introduce the concept of *Pareto-optimality* [3] to this problem. For the floorplanning problem discussed in this paper, we consider three objective functions. The first one is the chip area, denoted A, the second one is the total wire length, denoted W, and the third one is the weighted sum of A and W, denoted $F(A, W)$, which is defined as follows.

$$F(A, W) = A + k \times W \tag{1}$$

where k is a constant.

3 The Algorithm

3.1 Overview of the Algorithm

The proposed algorithm is based on metaheuristics, and consists of three stages. It is a parallel algorithm, which is implemented on a PC cluster. The algorithm consists of one master process and several slave processes. The master process controls overall behaviors of the algorithm, and slave processes executes their own tasks under the controls of the master. Figure 1 shows the flow chart of the algorithm.

The first stage was based on a genetic algorithm. The objective of this stage is to explore the search space globally in a shorter computation time. To achieve this objective, a GA-based floorplanning algorithm, which was originally proposed by the authors in [7], is executed on each PC in a PC cluster in parallel. In this stage, there is no difference between the master and slave processes. In each process, an initial set of chromosomes (i.e., floorplans) are randomly generated, and the GA-based floorplanning method is executed independently. To realize an efficient search, when calculating the total wire length of nets, any terminal of a net is assumed to be located on the center of a module. This assumption is very effective to reduce the time to evaluate each chromosome, since there is no need to consider positions of terminals as well as the precise orientation of each module.

After completing the first stage, the second stage of the algorithm begins. In this stage, first, all chromosomes obtained in all processes are merged into one set. Each process randomly selects chromosomes from this set to form an initial set of chromosomes, and starts the second stage. In this stage, each slave process periodically communicates with the master process. Each slave sends best chromosomes at that time to the master. If no improvement has been observed for a certain period in some slave, this slave demands the master to send new chromosomes to it to update the set of chromosomes. In this second stage, each chromosome is evaluated more precisely than the first stage. When calculating the total wire length, the position of each terminal of a net as well as orientations of modules are considered.

The third stage of the algorithm is based on tabu search. The master process selects best chromosomes among all chromosomes produced by slave processes in the second stage, and assigns some of them to each slave process to further improve it by tabu search. When assigning a chromosome to a slave process, the master also specifies the objective function to be optimized, since there are three objective functions, A, W, and $F(A, W)$, and tabu search cannot be used to achieve Pareto optimization for a set of solutions. To implement tabu search, neighborhood of a solution should be defined. In the proposed method, a solution (floorplan) s is regarded as a neighborhood of solution t, if s can be obtained by interchanging some pair of modules with modification of their orientation.

3.2 GA-based Floorplanning Method

As mentioned, the first and second stages of the proposed method are GA-based floorplanning methods, and those are based on the algorithm proposed by the authors in [7]. The base algorithm in the first and second stage is an ordinary generational GA. It keeps a set of chromosomes as a population, and by applying crossover, mutation, and selection operations, the next generation is periodically produced until a terminal condition is satisfied. In the following, each operation is briefly explained.

(1) Crossovers In [7], we have presented two crossover operators, called CTPX and PPEX, for the adaptive GA-based floorplanning method. In this paper, those two operators are used in the proposed parallel GA. CTPX is designed for preserving the characteristics of parents to offspring. On the other hand, PPEX is designed to explore the search space widely. Using two types of crossover operators enables the proposed method to explore the solution space effectively and efficiently. For lack of space, we omit the details of those two crossover operators.

(2) Mutation In [7], the mutation operator MM has been presented for the floorplanning problem. This mutation operator is also used in the proposed parallel GA. Mutation is occurred with mutation rate p_m per each gene of $\Gamma+$ and Θ. For each sequence-pair, MM randomly selects two module names and exchanges them, that corresponds to the movement of a module to some random location on the oblique grid. For the orientation of each module, MM flips the height and width of the module randomly.

(3) Selection As mentioned, the floorplanning problem discussed in this paper is formulated as a multi-objective optimization problem. For multi-objective optimization based on Pareto optimality, several selection operators have been proposed [2]. Among them, we adopt a crowded tournament selection operator proposed by Deb, *et al.*[2]. This selection operator is a variation of ordinary tournament selection operators. Using this operator, the set of chromosomes maintained in the GA is gradually converged to a Pareto-optimal solution set. The diversity among non-dominated solutions is also considered.

3.3 Parallel Implementation

To implement the proposed parallel floorplanning algorithm, we assume the message-passing model as a parallel programming model. In this model, the underlying hardware is assumed to be a collection of processors, each with its own memory. A processor has direct access only to the instructions and data stored in its local memory. An interconnection network supports message passing between processors. Processor A may send a message containing some of its local data values to processor B, giving processor B indirect access to these values. In our implementation, the whole program was implemented on a PC cluster, which connects multiple PCs with a high-speed local area network.

We adopted the MPI (message passing interface) library to write a parallel program, running on a PC cluster. MPI is the most popular message-passing library standard for parallel programming[8].

4 Experiment

To investigate the performance of the proposed parallel metaheuristic floorplanning algorithm, some experiments were conducted. We implemented the proposed parallel algorithm with the C language with the MPI library [12]. The proposed method was executed on a PC cluster, which consisted of 14 PCs, in which 7 PCs had Pentium 4 3.4GHz CPUs with 3GB memory, and the other 7 PCs had Pentium 4 3.2GHz CPUs with 3GB memory. All PCs were connected with a 1 Gbps Ethernet.

For the lack of space, we show experimental results for one benchmark data obtained from [11]. In this benchmark data, the number of modules is 200, the number of nets is 1274, and the lower bound of the chip area (i.e., the total area of all modules) is 17.57 (mm^2). We executed the proposed method in 5 times.

In experiments, we adopted the following parameter values: For the first and second stages, each process had a population of 25 chromosomes, and executed the GA process until the number of generations reached to $100,000$ and $50,000$, respectively. The probabilities of crossover and mutation were set to 0.6 and 0.005, respectively. For the first stage, the ratio of selecting CTPX and PPEX as crossover operators was set to $6:4$, and for the second stage, this ratio was set to $4:6$. In the second stage, in every 200 generations, each slave process sent their best 15 chromosomes at that time to the master. In the third stage, in tabu search at each slave process, for each solution obtained in the second stage, the maximum number of updating solutions was set to $18,000$.

Table 1 shows experimental results. In this table, "Area", "Wire length", "Weighted sum" and "CPU" in each column represent the chip area, the total wire length, the weighted sum of the chip area and total wire length, and the computation time, respectively. For the chip area, the dead space in percentage was also shown. "GA1" represents results when only the first stage was executed for $150,000$ generations. "GA1+GA2+TS" represents results of the proposed method. "Area", "Wire" and "Sum" in the row represent the results of best solutions for optimizations of

Table 1: Experimental results.

			Area (mm^2) (dead space (%))	Wire length (mm)	Weighted sum	CPU (sec)
GA1	Area	Best	**19.89 (11.67)**	2828.77	34.03	2090.31
		Ave.	**19.98**	2854.61	34.25	
	Wire	Best	20.75 (15.31)	**2691.97**	34.21	
		Ave.	21.94	**2725.15**	35.57	
	Sum	Best	20.20 (13.04)	2697.52	**33.69**	
		Ave.	20.12	2769.83	**33.97**	
GA1+ GA2+ TS	Area	Best	**18.69 (6.01)**	3914.88	38.27	5140.33
		Ave.	**18.73**	3796.50	37.72	
	Wire	Best	25.33 (30.63)	**2100.58**	35.83	
		Ave.	24.53	**2119.80**	35.13	
	Sum	Best	19.36 (9.26)	2433.76	**31.53**	
		Ave.	19.39	2457.78	**31.68**	

the chip area, the total wire length, and their weighted sum, respectively. "Best" and "Ave" in the row mean the best and average values of results.

From this table, we have the following observations. First, the multistage scheme of the proposed method could successfully reduce the computation time without degrading results. Second, a set of Pareto optimal solutions could successfully be obtained. Note that, in this experiment, 25 solutions were simultaneously produced as final results. From those results, users could select the best one according to their own criterion. Third, parallel processing was useful for reducing the computation time. If the proposed algorithm was executed on one CPU, its computation time would be nearly 20 hours. From those results, we conclude that the proposed method was effective to produce a good floorplan in a practical computation time.

5 Conclusion

We have proposed and implemented the parallel multistage metaheuristic floorplanning algorithm in VLSI building block layout. There are several possible extensions of the proposed method. First, since the timing should be considered in recent LSI design, we have a plan to modify the objective functions so that given timing constraints should be met. Second, it is interesting to modify the proposed method so that the floorplanning algorithm will be run on a larger PC cluster. Finally, in modern floorplanning, in addition to timing constraints, there are several other design constraints such as placement constraints of modules and fixed outline constraints of the chip area [1]. It is important to extend the proposed algorithm so as to handle those design constraints.

References

[1] S. N. Adya and I. L. Markov: "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE Trans. VLSI Systems,* Vol.11, No.6, pp.1120–1135 (2003).

[2] K. Deb: *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons (2001).

[3] D. E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company (1989).

[4] F. Glover and M. Laguna: *Tabu Search*, Kluwer Academic Publishers (1997).

[5] P. Mazumder, E. M. Rudnick: *Genetic Algorithms for VLSI Design, Layout & Test Automation*, Prentice Hall (1999).

[6] H. Murata, K. Fujiyoshi and Y. Kajitani: "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.15, No.12, pp.1518–1524 (1996).

[7] S. Nakaya, T. Koide and S. Wakabayashi: "A VLSI floorplanning method based on an adaptive genetic algorithm," *Journal of Information Processing Society of Japan*, Vol.43, No.5, pp.1361–1371 (2002), in Japanese.

[8] P. S. Pacheco: *Parallel Programming with MPI,* Morgan Kaufmann (1997).

[9] S. M. Sait and H. Youssef: *VLSI Physical Design Automation: Theory and Practice*, IEEE Press (1995).

[10] S. M. Sait, H. Youssef, H. R. Barada and A. Al-Yamani: "A parallel tabu search algorithm for VLSI standard-cell placement," *Proc. IEEE International Symposium on Circuits and Systems,* Vol.II, pp.581–584 (2000).

[11] http://www.cse.ucsc.edu/research/surf/GSRC/bench1.html

[12] http://www-unix.mcs.anl.gov/mpi/mpich1/