A Timing-Driven Block Placer Based on Sequence Pair Model

Gang Huang Xianlong Hong Changge Qiao Yici Cai

Department of Computer Science and Technology, Tsinghua University, Beijing 100084 P.R.China

Abstract *In this paper, an effective timing-driven building block placer is proposed. Interconnection delay is modeled and included during the placing process in order to minimize the area and wirelength, as well as to satisfy the timing constraints in the algorithm. The simulated annealing technique for constrained optimization problem and the sequence pair model proposed by H.Murata et al are applied. Not only the timing constraint but also the aspect ratio are taken into account in the search process. The experimental results demonstrate the algorithm can improve the timing delay and obtain good placemen*t.

Keywords: *timing-driven, building block placement, sequence pair, simulated annealing algorithm.*

1. Introduction

High performance designs beyond deep sub-micron technology have received much attention in recent years. Interconnect delay has become a very significant factor affecting design performance. The quality of floorplanning and placement will affect the performance heavily due to they are the highest level in the physical design pipeline.

The problem of timing driven building block placement can be described as follows. The inputs of the problem are composed of 1) a set of blocks with fixed geometries and fixed pins. 2) A netlist that some pins should be interconnected. 3) Timing information of pin or pad such as output driven resistance or load capacitance of pins, intrinsic delay of blocks and unit wire resistance and unit wire capacitance. 4) Some constraints on the aspect ratio. When the inputs are given, the timing-driven building block placement problem is solved by determining the locations and orientations on all the building blocks so that the area of the chip and the estimated wire length are minimized while ensuring the delay satisfies the timing constraint.

The approaches toward the timing driven floorplanning and placement problem are often categorized into two groups, path-based[1][2] and net-based[14]. In most net–based algorithms, the timing constraint is not directly processed. The timing slack is often transformed into net weights to guide placement. It has been reported that these techniques can achieve favorable results, however, since the timing in VLSI is inherently path–oriented, the path-based approach is expected to get direct accurate result.

Previous work in timing-driving placement is mainly tailored to layout styles that have regular structure such as gate array and standard cell style. (See [1][2]). Recently, there are a few works focused on the building block placement considering timing constraints, but most of them are not path–based. And most of previous papers restrict their placements to the slicing structure (see [3][10]).

The placements with slicing structure can be expressed by the data structure of binary tree, so many search algorithms can be applied on this kind of placements conveniently. However, the generality of slicing structure placement is limited. Recently, H.Murata *et al.* proposed an effective structure called sequence pair[6], by using of which non-slicing solutions can be evaluated efficiently during the stochastic optimization process. It is proved that the placement is a NP-hard problem. Due to the complexity of the problem, some stochastic search algorithm, such as simulated annealing algorithm or genetic algorithm, would be a good choice.

In this paper, using simulated annealing algorithm and sequence pair structure, we present an efficient timing driven building block placement algorithm. Our algorithm is path-based and not restricted to slicing structure. Not only the constraint about timing delay but also aspect ratio are taken into account.

2. Timing Model

A path with the worst delay time determines the time delay of a chip. A path starts at a primary input or at an output of a latch, and terminates at a primary output or at an input of a latch. The delay between two successive logic stages is composed of three elements: 1. Intrinsic delay of blocks 2. Delays due to charging fanout and load capacitance up/down. 3. Delay due to distributed RC of interconnection. Since interconnect has become the dominating factor in circuit performance and reliability in deep submicron designs, the delay of the third part will be quite significant.

The Elmore delay[4] model provides a simple closed form expression with greatly improved accuracy for delay compared to the lumped RC model. Our timing delay model is based on the Elmore delay model and it is similar to the timing model in [1][5]

We assume pin-to-pin based delay model. Let s be the index of the source pin, and let t be the index of the sink pin . The delay del(s,t)

$$\begin{aligned} Del(s,t) = {} & B_d + C_t \times R_u \times (|x_s - x_t| + |y_s - y_t|) \\ & + C_u \times R_u \times (|x_s - x_t|^2 + |y_s - y_t|^2) \\ & + R_s \times \sum_{i \in N} (|x_s - x_i| + |y_s - y_i|) + R_s \times C_{in} \end{aligned} \quad (1)$$

where B_d is the intrinsic delay of the block which s belonged to, R_s is the output driven resistance of s , C_t is the load capacitance of t, R_u and C_u are the unit wire resistance and unit wire capacitance respectively, N is the net include the source pin s and j is a pin of N which is driven by s, C_{in} is the sum of the load capacitance of the sink pins which are driven by s. The (x_s,y_s),(x_t,y_t) and (x_j,y_j) are the coordinates of s, t and j respectively.

The maximum delay of all paths is the time delay of the chip. Our goal is to optimize area, wirelength while ensuring the time delay satisfies the constraint. The experimental example we used have pointed out all paths,

which are sets of pins. For sink pins we can find the corresponding source pin of the same net, so it is not very hard to calculate the all path delay by (1)

3. Sequence Pair Model and Its Further Discussions

3.1 Sequence Pair Model

A sequence pair is an ordered pair of Γ_+ and Γ_-, where each of Γ_+ and Γ_- is a sequence of names of given modules, for example, (Γ_+, Γ_-)=(abcdefg, gedcabf) is a sequence pair of module set {a,b,c,d,e,f,g}. When a packing is given, the sequence pair can be obtained by so called positive loci and negative loci[6]. On the other hand, if a sequence pair is given, the horizontal or vertical relations between two modules are uniquely determined. For example, the packing shown in figure 1 corresponds to sequence pair (abefcd, cbdaef) and *vis versa*. Once the relations are given, the longest paths in horizontal and vertical orientation, which is the width and height of the chip, can be obtained in o(m^2) time.(see [6])

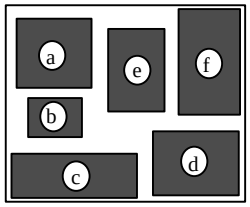

figure 1

3.2 Further Discussions

From figure 1, It can be seen that the area of the chip is not change when the positions of module e and f are exchanged. Generally, we have,

Property 1: Let (Γ_+, Γ_-) be a sequence pair. If the modules a and b are adjacent in both Γ_+ and Γ_-, that is, (Γ_+, Γ_-) has a form of (...ab... ,...ab...) or (...ab... ,...ba...), then the area of the chip is not changed when a and b are interchanged in both Γ_+ and Γ

Proof. Because a and b are adjacent in both Γ_+ and Γ_-, they have absolute same relationship (above, below, left and right) with other modules. So the horizontal longest path and vertical longest path do not change when a and b are interchanged in both Γ_+ and Γ_-, therefore the area of the chip is just the same.

It can be proved that for any sequence pair (Γ_+, Γ_-), the probability of existing at least two modules which are adjacent in both Γ_+ and Γ_- is 1-1/m^2, where m is the number of the modules.

The property 1 can be generalized as follows.

Property 2: Let A_1 and A_2 are two sequences of non-jointed subsets of the modules. If (Γ_+, Γ_-) can be represent as (...A_1A_2... ,...A_2A_1...), then the area of the chip does not change when any two modules in A_1(A_2) are exchanged or A_1 and A_2 are exchanged in both Γ_+ and Γ_-.

Example: Let (Γ_+, Γ_-) is (eabcdfg, dfabceg), A_1 is the sequence abc and A_2 is the sequence df, then the area is same when we exchange any two of the modules a,b and c or exchange d and f in both Γ_+ and Γ_-.The area is also the same if we exchange the position of A_1 and A_2 in Γ_+ and Γ_-. That is, the following sequence pairs map the chips with the same area, (ebacdfg, dfbaceg), (eabcfdg, fdabceg), (ecbafd, fdcbaeg) and (edfabcg, abcdfeg).

The property 2 is useful. We can apply this property to improve the timing delay and ensure that the area of the chip does not increase.

By the way, we show the relationship between the sequence pair and the slicing structure. In the following property, we give a special subset of all sequence pairs that can one to one corresponds to the slicing structure placements. From this point, it can be seen that in the area of optimal slicing structure placement is generally larger than that of general placement.

Property 3: Let Γ_+ is a combination of two sub-sequences A_1 and A_2, i.e.,Γ_+=A_1A_2, and Γ_- is a combination of A_1^* and A_2^*, That is Γ_-=$A_1^*A_2^*$ or Γ_-=$A_2^*A_1^*$, where A_i^* (i=1, 2) is A_i or another arrangement of all modules in A_i, then the placement (Γ_+, Γ_-) is combination of placements of (A_1, A_1^*) and (A_2, A_2^*). Furthermore, if the placement (A_1, A_1^*) and (A_2, A_2^*) also satisfy the above condition and this process can continue recursively until there is only one module left, then (Γ_+, Γ_-) maps a placement with slicing structure. On the other hand, a slicing structure placement can map a sequence pair satisfying the condition mentioned above.

Example: Sequence pair (efhgabcd,acdbehgf) maps a placement with slicing structure.

4. Simulated Annealing Algorithm for constrained optimization problem

4.1 Simulated Annealing Algorithm

The placement and routing of VLSI circuits are often so difficult that they could not be solved exactly, so some of approximate methods are used. Simulated annealing is a stochastic computational technique derived from statistical mechanics for finding near globally minimum cost solution to large optimization problems. In 1983,Kirkpatrick *el al* [11] first proposed and demonstrated the application of simulation techniques from statistical physics to problems of large combinatorial optimization.

4.2 Simulated Annealing Algorithm for constrained optimal problem

In the VLSI physical design, most optimization problems are constrained optimization problems. So it is necessary to apply the simulated annealing algorithm to handle this kind of problems. The constrained optimization problem is commonly presented as follows.

$$\begin{aligned} &\min f(s) \\ &\text{s.t. } g_i(s)\le 0,\ i=1,2,...n. \\ &\quad h_j(s)=0,\ j=1,2,...m, \end{aligned} \tag{2}$$

where f(s) is the cost function, $g_i(s)$ and $h_j(s)$ are the inequality and equality constraints respectively.

4.2.1 Inequality constrained optimization problem

In this case, the problem can be presented as

$$\begin{aligned} &\min f(s) \\ &\text{s.t. } g_i(s)\le 0,\ i=1,2,...n. \end{aligned} \tag{3}$$

The methods using simulated annealing algorithm to deal with the inequality constrained optimization problem can be grouped into the following two classes.

a) The first class method is the method of inspection. In the simulated annealing process, when the new neighbor status is created, it is checked if it satisfy the constraints or not, if not, then this status is given up and a new status is created. This process goes on until a feasible status is found, then the simulated annealing process continues. This kind of methods can ensure that the iteration is always in the field of the feasible solutions, however, sometimes it needs a lot of time to find a feasible solution.

b) The second approach is similar to the penalty function method. It is very effective to solve the constrained optimization problems. For the problem (3), we give a new cost function by adding the constrained conditions as penalty functions. So the constrained optimization problem become an unconstrained optimization problem and the general simulated annealing algorithm can be easily applied. A new cost function can be written as

$$F(s)=f(s) + L\ (\max(0,g_1(s))+\max(0,g_2(s))+\cdots+\max(0,g_n(s))), \quad (4)$$

where L is a penalty factor. If the status s is a feasible solution, $g_i(s)\le 0, i=1,2,...n$, then the penalty functions equal zero, so F(s) = f(s), that is, both functions have the same minimum in the feasible solution space, hence the optimal solution can be found. If some constraints are not satisfied, then the penalty factor makes the F(s) be rather large, so this status s could not be the optimal solution. In the iteration process, infeasible status could be accepted. The penalty functions in (4) have a very simple form and can be easily calculated. It is not usually used in traditional penalty methods of nonlinear programming problem because the penalty functions are not continuous differential. However, it is very suitable for the simulated annealing algorithm because the algorithm dose not need any analytical property of the penalty functions. In the traditional penalty methods the penalty factor is often increase during the iteration process, but we take it as a constant in our algorithm.

4.2.2 Equality constrained optimization problem

For this case, the problem becomes

$$\min f(s)$$
$$\text{s.t. } h_j(s)=0 \quad j=1,2,\ldots m. \quad (5)$$

There are also two kinds of methods to deal with the problem. One is to solve every equation of $h_j(s)=0$ and determine the feasible solution space. Unfortunately, this method is hard to be applied to the combinatorial optimization problems. The other method is also the penalty function method, that is, the cost function is

$$F(s) = f(s) + M\times(|h_1|+|h_2|+\ldots\ldots+|h_m|)$$
$$\text{or}\quad F(s) = f(s) + M\times(h_1^2+h_2^2+\ldots\ldots+h_m^2)$$

where M is the penalty factor. The experiences show that the penalty factor will not be too large because a large M probably leads the cost function to sink into a local minimum.

5. Timing Driven Building Block Placement

5.1 The goal function and the constraints

The object of placement is to find a layout of minimum area and wirelength while ensuring the delay satisfying the constraint. Let s be a configuration of a placement, then the problem can be presented as following,

$$\min(\text{Area}(s) + \text{weight}_1\times \text{wirelength}(s))$$
$$\text{s.t.}\quad \text{delay}(s) \le D$$
$$R(s) = 0, \quad (6)$$

where D is the up bound of the time delay, R(s)= max(|Ratio– width(s)/height(s)|, |Ratio – height(s)/width(s)|), and Ratio is a constant to denote the desired aspect ratio. If the ratio goal is W:H, then Ratio = max(M/H, H/M). For instance, if one likes the aspect ratio to be 2:1, then the Ratio equals to 2.

The cost function can be presented as follows,

$$\text{Cost}(s)=\text{Area}(s)+w_1\times\text{wirelength}(s) + w_2\times\max(0,\text{delay}(s)-D)+w_3\times R(s).$$

We will take a large number for w_2 to guarantee the delay less than the bound D.

5.2 The Cooling Schedule

The key step to apply simulated annealing algorithm is to choose a suitable cooling schedule. In our algorithm, the initial temperature is 80000, the reductive function is $t_k=\alpha\times t_{k-1}$,α=0.95. The initial inner loop is 3000 and it increases in the rate of 0.01, that is, the inner iteration times satisfy $L_k=\beta\times L_{k-1}$,β=0.01, which means we gradually enhance the ability of local search along with the lower of the temperature.

The creation of a neighboring solution is based on: 1) randomly pairwise interchange in Γ_+ or Γ_-, where Γ_+ and Γ_- is taken also randomly. 2) rotate the module for 90^0,$180^o$and 270^0. 3) Reflect the modules in both horizontal and vertical orientations. The third operation is used in the process of optimization of wirelength and timing delay.

5.3 The wirelength estimation

The common wirelength estimation is half-perimeter. However, for timing driven problem it seems that the radius of a net correlates to the wiring delay much better than the half-perimeter value does (see [7]). In this paper, the estimated wire length of a net N is $R(N)=\max_{j\in N}(|x_s-x_j|+|y_s-y_j|)$, where (x_s,y_s) is the coordinate of the source pin s and (x_j,y_j) denote the coordinates of other pins of the net. The total wirelength is the sum of R(N) of all nets.

5.4 The post process

After the simulated annealing procedure, we have a post process to improve the wirelength and the timing delay. By using property1 or property2 of section 3, we can create a new configuration which has different sequence pairs but with the same area. The cost function is Cost(s) =wirelength(s) +W delay(s), where weight W is taken as 10000. The post process algorithm will apply the local search approach. If the new configuration could not be found or the iteration times reach the certain time, then the search process terminate. In this way, we can further improve the wirelength and the timing delay.

6. Experiments

The timing driven placement algorithm has been implemented in the C programming language, and all experiments are performed on a SUN spark20 workstation.

6.1 Test Examples

Three building-block benchmarks, ami33T, ami49T and xeroxT, are used for the experiments. These benchmarks are obtained by adding appropriate timing information to the CBL/NCSU benchmarks ami33, ami49 and xerox, respectively. We got the examples from Dr. Donsheng Wang of University of California at Berkeley and we make a little change to adapt the placement needs.

Table 1. Benchmark specifications

example	blocks	nets	paths
ami33T	33	123	230
ami49T	49	408	116
xeroxT	10	203	86

6.2 Experimental Results

Table2 presents our placement results for examples of ami33T, ami49T and xeroxT with aspect ratio goal 1:1. In the Table3, we compare the results of timing-driven placement algorithm with the results of the general placement algorithm without timing consideration. The results demonstrate that the timing-driven placement algorithm has more dead space than that of our placement algorithm without timing constraint (The average dead space increased 5.3%), but the timing delay of the chip is improved a lot (The average timing delay decreased 33.6%). Although the timing-driven algorithm needs more time, the time is not too long to be accepted.

Table 2. The results of timing-driven placement algorithm

example	area (μm^2)	dead space (%)	wire length (μm)	time constraints (ns)	run time (min)
ami33T	1253959 (1141 1099)	7.7	86548	6	31.3
ami49T	38985772 (6286 6202)	9.0	569078	6	48.2
xeroxT	23160732 (4634 4998)	16	839846	6	38.8

Table 3. The comparison between two algorithms

example	dead space increased(%)	timing delay improved(%)	run time increased(min)
ami33T	3.2	39.9	26.8
ami49T	4.6	30.1	42.1
xeroxT	8.3	30.8	32.0

7.conclusion

A new timing-driven placement algorithm for building block layout is proposed in this paper. The experimental results show that the algorithm is effective to deal with the timing constrained placement problem.

8. Acknowledgments

The authors would like to thank Dr.Dongsheng Wang for his help. This work was supported by Chinese National Nature Science Funds and Chinese National Key Project funds under the grants 69776027 and 96-738-01-08.

References

[1] T.Hamada, C.K.Cheng and P.M.Chau, " Prime: A timing-driven placement tool using a piecewise linear resistive network approach" in 30th ACM/IEEE Design Automation Conference, pp.531-536. 1993.

[2] A.Srinivasan, K.Chaudhary and E.S.Kuh: "RITUAL: A performance-driven placement algorithm," IEEE Trans on Circuits and Systems II, Vol.39, No.11, pp.825 - 839.1992.

[3] T.Gao, P.M.Vaidya and C.L.Liu: " A new performance driven placement algorithm," in Proc. of International Conference on Computer Aided Design, pp.44 -47. 1991.

[4] W.C.Elmore: "The transient response of damped linear networks with particular regard to wideband amplifiers," J.Appl.Phys. Vol.19, pp.55 - 63. 1948.

[5] X.L.Hong, T.X.Xue, J.Huang, C.K.Cheng and E.S.Kuh," TIGER: An efficient timing - driven global router for gate array and standard cell layout design," IEEE Trans on CAD, Vol. 16, No.11.pp.1323 - 1331. 1997.

[6] H.Murata *et al.* "VLSI module placement based on rectangle-packing by the sequence pair", IEEE Tran. Computer Aided Design, Vol.15, pp.1518-1524, 1996.

[7] Y.W.Tsay *et al*, "An inproved objective for cell placement," in ASP-DAC, pp.281 - 284.1997

[8] M.Kang and W.W.-M.Dai, "General floorplanning with L-shaped, T-shaped and soft block based on bounded slicing grid structure", in ASP- DAC, pp.265-270,1997.

[9] D.Wang and E.S.Kuh, "Performance-driven interconnect global routing," Technique Report of Dept. Electrical Engineering and Computer Science of University of California at Berkeley. 1997.

[10] H. Esbensen & E.S.Kuh, "An MCM/IC timing-driven placement algorithm featuring explicit design space exploration", in IEEE Milti-Chip Module Conference, pp.170-175, 1996.

[11] S.Kirkpatrick *et al.* "Optimization by simulated annealing," Science.Vol.220, pp671-680, 1983.

[12] T.Kong and X.L.Hong, "Timing-driven floorplanning algorithm for building block layout," in Fourth International Conference on CAD & Computer Graphics. Wuhan, China, pp.669-674. 1995.

[13] V.Moshnyaga *et al*.,"Performance-driven macro-block placer for architectural evaluation of ASIC designs," IEE Proc. Circuits Devices Syst. Vol.144, No.3, pp.190-194. 1997

[14] M.Marek-Sadowsda and S.P.Lin, "Timing-driven placement," in Proc.ICCAD-89, pp.94-97, 1989.

[15] G.Huang, X.L.Hong, C.G.Qiao and Y.C.Cai, "Building block placement optimization based on sequence pair model considering area, aspect ratio and wirelength," in Fifth International Conference on CAD & Computer Graphics. Shenzhen, China, pp.333-338. 1997.