INFORMATICA, 1996, Vol. 7, No. 3, 349–360

PERFORMANCE DRIVEN PLACEMENT USING TABU SEARCH

Andrew LIM

Information Technology Institute
11 Science Park Road, Singapore 0511, Republic of Singapore

Abstract. In this paper, we present an effective performance driven placement with global routing algorithm for macro cells. Our algorithm uses a hierarchical, divide and conquer, quad-partitioning approach. The quad-partitioning routine uses the *Tabu Search* technique. Our algorithm uses the concept of proximity of regions to approximate the interconnection delays during the placement process. In addition, our algorithm can handle modules whose positions are fixed or are restricted to a particular subregion on the layout frame. Our experimental results indicate the superiority of our placement method in terms of quality of solution and run time when compared to Lin and Du (1990).

Key words: CAD, VLSI, module placement, global routing, interconnection delay, quad-partitioning, tabu search.

1. Introduction. Advances in VLSI technology resulted in circuit devices with smaller feature sizes, switching delays and driving strength. The decrease in driving strength increases the propagation delay of signals through the wires connected to the devices. Consequently, interconnection delay becomes a major contributing factor (Donath *et al.*, 1990; Sutanthavibul and Shragowitz, 1990) to circuit performance. It was reported (Donath *et al.*, 1990; Sutanthavibul and Shragowitz, 1990) that interconnection delays can be more than 50% of the circuit delay in some circuits. Hence, interconnection delays can no longer be ignored, and the performance driven placement problem has become an important research topic in CAD for VLSI (Donath *et al.*, 1990; Dunlop *et al.*, 1984; Hauge *et al.*, 1987; Jackson and Kuh, 1989; Jackson *et al.*, 1990; Lin and Du, 1990; Sutanthavibul and Shragowitz, 1990; Wu *et al.*, 1992).

The performance of a circuit is bounded by the clock cycle applied to the circuit (Bakoglu, 1990). In non-pipelined circuits, the clock period should not be shorter than the delay of the longest path in the circuit. Consequently, the delay of the longest path bounds the circuit performance, thus reducing the delay of the longest path improves circuit performance.

Due to the immense complexity of the VLSI layout problem, it is normally divided into subproblems like partitioning, floorplanning, placement, global routing, detailed routing and layout compaction. Unfortunately, all these subproblems are still NP-hard.

In this paper, we consider the problem of performance driven placement with global routing. The placement problem is to determine the positions and orientations of the modules on a plane based on information of the modules and their interconnections. The global (also known as loose) routing problem is to decide approximately which regions the interconnections should be routed through. Placement algorithms that incorporate interconnection delay information into consideration are known as performance driven placement algorithms. Placement is often considered to be the most important stage in performance driven layout as it determines the geometric locations of modules, thus, indirectly determining lower bounds of the interconnections. However, placement algorithms that do not consider global routing will overestimate the performance of the circuit as numerous detours of interconnections usually take place due to congestion. Worst of all, it is possible that such a placement cannot be realized at all as prior planning for routing is not done during placement. The objective of our algorithm is to obtain a placement of modules of a circuit together with the consideration of global routing so that we can route the circuit with the predicted good performance during detailed routing.

Two major difficulties to include path delays into placement algorithms are:

- Interconnection delays are unknown at the time of placement.
- There are huge number of paths in the circuit.

Delay estimation is inevitable to cope with the problem of unknown module positions and unknown interconnection delays. Many timing driven placement algorithms have been proposed. They can be roughly categorized as either net-oriented (Dunlop *et al.*, 1984; Hauge et al., 1987; Jackson and Kuh, 1989; Jackson, 1990) or path-oriented (Donath *et al.*, 1990; Lin and Du, 1990; Sutanthavibul and Shragowitz, 1990).

The net-oriented approach avoids the problem of a large number of paths by considering delays at the net level. Such an approach usually does not have a global view of the circuit and does not capture the path delay accurately. The common scheme for the net oriented approach is to assign weights to nets depending on how critical they are and use the weights to guide the placement process (Dunlop, *et al.*, 1984; Hauge *et al.*, 1987).

Path-oriented (Donath *et al.*, 1990; Lin and Du, 1990) approaches estimate the delays of paths in the circuit. This estimation is used in the placement process. Donath *et al.* (1990) use timing analysis of complete paths to guide the placement. Lin and Du (1990) use a window-based approach and consider delays of the K most critical paths, where K is a constant. Since there can be enormous number of paths in a circuit, the path oriented approach is quite time consuming.

General optimization methods, like mathematical programming techniques and simulated annealing methods have also been attempted (Jackson and Kuh, 1989; Jackson *et al.*, 1990; Sechen and Sangiovanni-Vincentelli, 1986). However, such methods are usually very time consuming.

In this paper, we propose a hierarchical, divide and conquer, quad-partitioning approach. The quad-partitioning is done using the *Tabu Search* meta-heuristic. We use the concept of proximity of regions to approximate interconnection delays during the placement process. Our delay approximation will progressively be more accurate as the regions that are being examined become smaller. In addition, our algorithm can handle modules whose locations are fixed, or restricted to a particular subregion in the layout frame. This flexibility allows the algorithm to handle placements with a mixture of standard cells and macro cells. The terminology and problem formulation will be presented in Section 2. In Section 3, our algorithm will be presented. Experimental results will be reported in Section 4. Our conclusions and future work will be discussed in Section 5.

2. Terminology and problem formulation. Fig. 1 clarifies some definitions and terms we shall use below. The layout frame is rectangular in shape. It is the region where modules, except in the case of I/O pins, are placed. Surrounding the layout frame are the I/O pads. Only I/O pins can be assigned to I/O pads. Modules are rectangular in shape and associated with each of them is a delay value called module delay.

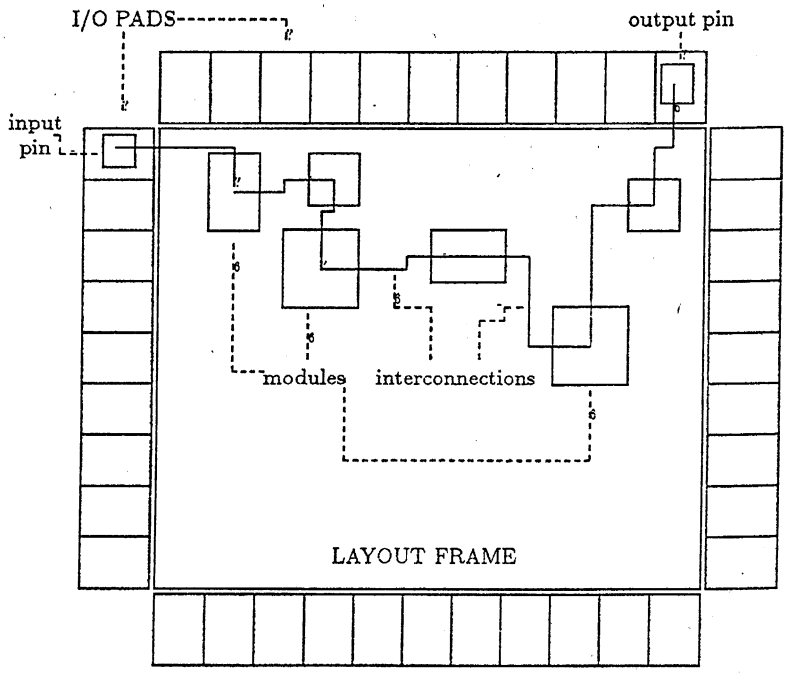

Fig. 1. Layout.

2.1. Circuit topology and modeling. Net list is a list containing the interconnections between modules. We assume that the modules and interconnections can be modelled by a directed acyclic graph (DAG) $G = (V, E)$. We associate a node with each module. If a module is connected to another, there will be a directed edge between them. The direction of the edge depends on the direction of the signal flow. So, all input pins have in-degree of zero and output pins have out-degree of zero. Other modules, which are not I/O pins, have non-zero in-degree and out-degree nodes. A path will always start and end with I/O pins. This can easily be extended to the situation where there are synchronizing elements, like latches or flip-flops, in the circuit. In that situation, each synchronizing element is split into 2 modules such that all the incoming connections of the synchronizing element will be the incoming connections of one module with no outgoing connection, and the outgoing connections will be the outgoing connections of the other module with no incoming connection.

With the modification, a path begins with an input pin or an output pin of a synchronizing element and ends with an output pin or an input to a synchronizing element. Path delays can then be computed accordingly.

2.2. Delay model. A path starts with an input pin and ends with an output pin. The modules in Fig. 1 form a path. Path delay is the sum of all module delays and interconnection delays on the path.

The delay of a path $P =< M_{i_1} \rightarrow M_{i_2} \rightarrow \ldots \rightarrow M_{i_k} >$ is estimated by:

$$\text{Delay}(P) = \sum_{1 \leqslant j \leqslant k} MD(M_{i_j}) + \sum_{1 \leqslant j < k} ID(M_{i_j}, M_{i_{j+1}}), \tag{1}$$

where $MD()$ is the module delay (given for each module) and $ID()$ is the interconnection delay. Note that M_{i_1} and M_{i_k} are I/O pins.

Interconnection delay (ID) is:

$$ID(M_i, M_j) = K_{RC} \times \text{Dist}(M_i, M_j)^2, \tag{2}$$

where $\text{Dist}(M_i, M_j)$ is the Manhattan distance between module M_i and module M_j. K_{RC} is a constant determined by the average resistance and capacitance per unit length of the interconnection over all routing layers.

2.3. Region constraints. Some modules may be placed only in certain regions. Constraints on the possible locations of modules are to handle the following situations:

- Movable I/O pins. Some I/O pin positions may be fixed, others may be assigned to any of the available I/O pads.
- Locations of some modules may be already fixed. In some cases, where the layout consists of macro cell modules and standard cell modules or digital and analog modules, the regions in which they are supposed to be placed may have already been decided. So, such a flexibility will enable us to perform placements in such situations.

2.4. Formulation for performance driven placement with global routing. In our performance driven placement with global routing problem, we are given:

- a set of modules with fixed rectangular geometries and delays (modules may be I/O pins);
- a layout frame where the modules are to be placed;

- constraints on the possible positions of some modules (if any);
- a net list specifying the interconnections between modules;
- resistance and capacitance attributes of the interconnections.

The objective of our algorithm is to determine the positions of modules in and on the layout frame such that

1) global routing is performed;
2. the delay of the longest path is minimized;
3. there is no overlap between modules.

3. The algorithm. The outline of our algorithm is given in Fig. 2. Details of the individual steps are given in the subsections below.

Step 1: Put the entire placement region into queue Q.
Step 2: If Q is empty goto Step 7, else do
- Get region R from Q.
- Divide R into 4 subregions R_1, R_2, R_3, R_4.

Step 3: Generate an initial solution.
Step 4: Calculate path delays.
Step 5: Perform Quad-Partitioning.
Step 6: If R_i, $i = 1, 2, 3, 4$, has k or more modules in it, add R_i into Q, else add to branch and bound queue, BBQ. Goto Step 2.
Step 7: For each region, R, in BBQ perform a branch and bound search for the best solution for that region.
Step 8: Shift and adjust to remove overlaps.

Fig. 2. Outline of our algorithm.

3.1. Division. Given a region R, let the 2 corners of the region be $(X_{\min}, Y_{\min})$ and $(X_{\max}, Y_{\max})$. We divide the region R into 4 subregions R_1, R_2, R_3, R_4 by cutting R horizontally and vertically through its center point, $(\frac{X_{\min}+X_{\max}}{2}, \frac{Y_{\min}+Y_{\max}}{2})$ (see Fig. 3).

3.2. Initial solution. The initial solution is obtained by randomly assigning modules with region constraints first into the four subregions, and then followed by all other modules. The assignment is done in such a way that the total area of all modules in each subregion is almost equal.

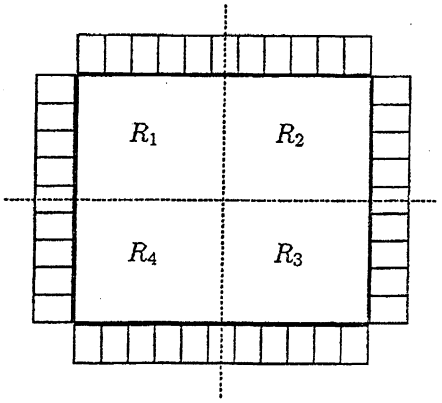

Fig. 3. Division of a region.

3.3. Path delay estimation. First, we need to estimate the length of all interconnections. We define X_c^R to be the center of region R. If the edge connects 2 modules in the same region, then its length is estimated by:

$$\frac{(X_{\max}^R - X_{\min}^R) + (Y_{\max}^R - Y_{\min}^R)}{3.5}.$$

If the modules are in different regions, R_1 and R_2, then the length is:

$$\left|X_c^{R_1} - X_c^{R_2}\right| + \left|Y_c^{R_1} - Y_c^{R_2}\right|.$$

After the length of each edge is computed, its delay can be estimated using Eq. 2 in Section 2.2. Finally, for each edge e, we compute:

1) longest path length passing through it, $L_e^{\max}$;
2) number of paths through it, N_e;
3) average lengths of all incoming paths and outgoing paths.

Note that this step can be done by a forward and backward topological sweep. It takes no more then $O(|E| + |V|)$ time, where $G = (V, E)$ is the directed acyclic graph modeling the circuit topology (see Section 3.1).

3.4. Quad-partitioning using tabu search. This the most important part of the algorithm. The quality of the final solution depends on the quality of solution obtained at each level of the partitioning. Quad-partitioning is chosen instead of doing 2 bi-partitioning because the placement problem is a 2-dimensional problem and bi-partitioning is essentially one dimensional.

Our partitioning algorithm is based on the *tabu search* approach. We define the cost function to be:

$$\alpha F_A + \beta F_B + \gamma F_C + \delta F_D + \epsilon F_E,$$

where $\alpha, \beta, \gamma, \delta$ and ϵ are constants that are experimentally tuned.

F_A measures the imbalance of area usage of the 4 subregions, R_1, R_2, R_3 and R_4 created. It is given by the expression:

$$\frac{\max_{i=1}^{4}\{\sum_{\text{module}\in R_i}\text{Area(module)}\} - \min_{i=1}^{4}\{\sum_{\text{module}\in R_i}\text{Area(module)}\}}{\text{Area}(R_1) + \text{Area}(R_2) + \text{Area}(R_3) + \text{Area}(R_4)}.$$

F_B is the total number of edges that pass through all the four boundaries. An edge that connects two modules, one in R_1 and the other in R_3 (or one in R_2 and the other in R_4), contributes $\frac{1}{2}$ an edge to all the four boundaries.

F_C measures the imbalance of edges through the boundaries. Formulation is analogous to F_A. It is given as the maximum difference between the number of edges across any of the four boundaries.

F_D is the square of the summation of $L_e^{\max}$ where e is an edge that spans across one of the four boundaries. If an edge connects one module in R_1 and the other in R_3 or one module in R_2 and other in R_4, then the $L^{\max}$ of the edge is added twice to the total.

F_E measures the criticality of the edges that span across one of the four boundaries. By criticality, we mean the number of long paths that pass through this edge and their average length.

We use the general framework of the tabu search technique as outlined in Fig. 1. The neighborhood configurations of the current configuration are those configurations that are reachable from the current configuration by the two types of moves given below:

- Move one module from one subregion to another.
- Exchange two modules in different subregions.

The neighborhood configurations of the current configuration are obtained by performing one of the above moves. Among all possible moves, we choose the best one that does not satisfy the tabu conditions or if the aspiration function overwrites the tabu conditions. There can be at most $O(n^2)$ moves at each level of the subdivision.

A move will only affect the incoming and outgoing edge lengths of the module/modules being moved. Note that recomputing everything is not necessary to determine the cost of the new configuration. All that is needed is to recompute those edge lengths and path lengths affected by the move. In practice, this takes $O(1)$ time (in the worst case $O(|V|+|E|)$ time is required).

The *tabu list* stores the last 10 modules being moved. The length of the tabu list is chosen to be 10 as that seems to give consistently good partitions with the least time required to check for tabu status, although any length between 10 to 15 will suffice. The *aspiration function* stores the cost of the configuration when the module was last in that subregion. The aspiration function will overrule the tabu condition if the current move decreases the aspiration function value for the module in that subregion.

3.5. Branch and bound search. We fixed the constant k to be 10. When the number of modules in any region is less than 10, we perform a branch and bound search for the best possible arrangement of the modules in that region. This is done by generating non-overlapping constraints and using a tool like CPLEX/AMPL to solve the layout problem.

3.6. Shifting and adjustment. Minor overlaps may happen after the first seven steps. We will adjust the placement without changing the relative positions of the modules to eliminate the overlaps. This is done by starting at a corner of the layout frame and considering module packing along the direction of the diagonal from that corner. After all overlaps are removed, minor adjustments are made to further improve the layout.

4. Experimental results. We implemented a preliminary version of our algorithm in C and ran our experiments on a SUN SPARC station. We use the test circuits in Lin and Du (1990). Table 1 provides the characteristics of the test data. Table 2 tabulates our results. Our algorithm is faster then the algorithm proposed in Lin and Du (1990). This is obvious as in Lin and Du (1990), the performance of their algorithm depends on the number of paths in the circuit, and the number of paths can grow exponentially in $|V|+|E|$. The length of our longest paths are only slightly superior than those in Lin and Du (1990). This is because in our algorithm, global routing is already taken care of. In Lin's placement, the only objective is to minimize the longest path length without considering routability So, if global routing is performed on their layout, the

Table 1. Characteristics of test cases

	# of modules	# of nets	module density	# of paths
ckt 1	10	26	0.3	18
ckt 2	15	32	0.3	23
ckt 3	20	43	0.4	51
ckt 4	20	53	0.4	81
ckt 5	30	75	0.4	305
ckt 6	40	104	0.5	1314
ckt 7	40	104	0.6	1314
ckt 8	40	104	0.7	1314
ckt 9	60	153	0.75	1274
ckt 10	100	201	0.5	7438

Table 2. Comparisons with Lin's algorithm

		Lin's Algorithm		Our Algorith	
	lower bound	circut delay	run time (sec)	circuit delay	run time (sec)
ckt 1	239.66	326.73	2.12	320.6	1.01
ckt 2	296.02	345.33	11.45	349.2	3.21
ckt 3	424.62	594.06	27.79	593.8	7.66
ckt 4	424.62	611.62	31.70	615.1	8.89
ckt 5	1241.47	1581.8	161.09	1500	41.6
ckt 6	1266.96	1570.8	841.25	1550	215
ckt 7	1266.96	1703.3	849.19	1679	256
ckt 8	1266.96	1684.5	846.65	1654	289
ckt 9	1507.67	2216.9	1637.4	2210	643
ckt 10	2469.03	3012.6	16832	3000	2412

solution could be much worse.

5. Conclusion and future work. In this paper, we proposed an effective performance driven placement algorithm that takes global routing into consideration. Our experimental results indicate its usefulness. Our algorithm uses a hierarchical, divide and conquer, quad-partitioning concept, incorporat-

ing global routing and performance issues into the quad-partitioning routine. Our algorithm also uses a novel approximation scheme to estimate net delays. Our approximation becomes progressively more accurate as the regions become smaller. In addition, our algorithm allows region restrictions for modules to handle fixed and movable I/O pins and possibly analog and digital placement if the analog and digital regions of the layout frame are already pre-defined. In the quad-partitioning routine, we applied the concept of *Tabu Search* to obtain good results. The results in this paper indicate the potential of applying the tabu search technique to other VLSI CAD problems. We are currently extending this work to handle pin to pin delay computations. In addition, we hope to use the tabu search framework to develop partitioning, floorplanning, detailed routing and layout compaction subsystems and integrate these subsystems into a complete layout tool.

REFERENCES

Bakoglu, H.B. (1990). *Interconnections, and Packaging for VLSI.* Addison-Wesley, Menlo Park, California.

Donath, W. *et al.* (1990). Timing driven placement using complete path delay. In *21st Design Automation Conference.* pp. 84–89.

Dunlop, A.E., V.D. Agrawal *et al.* (1984). Chip layout optimization using critical path weighting. In *21st Design Automation Conference.* pp. 133–136.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Comput. and Ops. Res.,* **13**(5), 533–549.

Glover, F. (1989). Tabu search – Part I. *ORSA Journal on Computing,* **1**(3), 190–206.

Glover, F., and H.J. Greenberg (1989). New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European Journal of Operational Research,* **39**, 119–130.

Glover, F. (1989). Tabu search – Part II. *Technical Report CAAI-TR-89-05.* University of Colorado at Boulder, Center For Applied Artificial Intelligence.

Hauge, P.S., R.Nair and E.J. Yoffa (1987). Circuit placement for predictable performance. In *ICCAD–87.* pp. 88–91.

Jackson, M.A.B., and E.S. Kuh (1989). Performance-driven placement of cell based IC's. In *26th Design Automation Conference.* pp. 370–375.

Jackson, M.A.B., A. Srinivasan and E.S. Kuh (1990). A fast algorithm for performance-driven placement. In *International Conference on Computer-Aided Design.* pp. 328–331.

Lin, I., and D. Du (1990). Performance-driven constructive placement. In *27th Design Automation Conference.* pp. 103–106.

Sechen, C., and A. Sangiovanni-Vincentelli (1986). Timberwolf3.2: A new standard cell Placement and Global Routing Package. In *ACM/IEEE Design Automation Conference.* pp. 432–439.

Sutanthavibul, S., and E. Shragowitz (1990). An adaptive timing-driven layout for high speed VLSI. In *27th Design Automation Conference.* pp. 90–95.

Wu, C.T., A. Lim and D. Du (1992). An effective timing-driven placement algorithm for macro cells. In *5th International Conference in VLSI Designs.* pp. 31–35.

Received April 1996

A. Lim received his B.Sc. (High Distinction), M.Sc. and Ph.D. degrees from the University of Minnesota (Twin Cities) in 1987, 1990, and 1992 respectively. At present he is a consultant (decision support systems) at the Information Technology Institute and an adjunct lecturer at the Department of Information Science and Computer Science. Dr. Lim's research interests and expertise include operations research/management in transportation and logistics, and combinatorial optimization.

PRALAIDUMO VALDOMAS IŠDĖSTYMAS, BESIREMIANTIS TABU PAIEŠKA

Andrew LIM

Straipsnyje pateikiamas efektyvus makro elementų išdėstymo metodas, tuo pačiu tenkinantis ir globalaus trasavimo reikalavimus. Siūlomas algoritmas remiasi hierarchiniu skaidymo į keturias dalis metodu: „dalink ir valdyk". Skaidymo į keturias dalis procedūra vartoja Tabu paieškos metodą. Jungčių užlaikymams aproksimuoti išdėstymo procese siūloma sričių artumo koncepcija. Be to, pagal siūlomą algoritmą galima išdėstyti modulius, kurių pozicijos yra fiksuotos arba apribotos tam tikromis sritimis projektuojamojo objekto šablone. Eksperimentiniai duomenys gauti mūsų metodu, lyginant juos su paskelbtais Lin, yra pranašesnis sprendinio kokybės ir vykdymo laiko atžvilgiu.