An I/O Pad Assignment Algorithm for IC Design

Somchai Prasitjutrakul

Department of Computer Engineering, Chulalongkorn University
Phayatai Rd. Bangkok 10330, THAILAND

ABSTRACT

This paper presents an algorithm for I/O pad assignment used to assign off-chip I/O's to pads (at the chip periphery) prior to the module placement process. Both timing and geometrical constraints are transformed into connection strengths which represent proximity relationships for the off-chip I/O's of the network. An objective function for the I/O pad assignment problem is proposed and experimentally shown to be almost monotonically increasing with the cost of the module placement. An algorithm for minimizing the objective function using a heuristic constructive assignment followed by iterative improvements was experimentally shown to produce solutions comparable to solutions using the simulated annealing technique, but requiring far less computation time.

1. INTRODUCTION

When using a physical analogy (e.g. force, energy, resistive network) or mathematical programming methods to model and solve the module placement problem, fixed assignment of off-chip I/O's to pads is usually required prior to placing the modules [1,2,3,4,5,6,7,8,9]. This is so because well-defined relative positions of the modules are obtained when the off-chip I/O positions are fixed, whereas all modules collapse to the center of the chip (assuming no repulsive force is used) when they are not. It is obvious that different I/O pad assignments will usually lead to different placement results. In other words, the quality of the module placement result depends heavily on the preassignment of I/O pads [2] as shown for the example in Figure 1.

In this paper, an I/O pad assignment algorithm is presented. The algorithm transforms the entire network into a new network consisting only of off-chip I/O's and their relationships. The relationships (weights) among the off-chip I/O's indicate the proximities of the I/O's, and are derived from the path-delay constraints and the structure (which considers both module sizes and their interconnections) of the original network. Then an objective

function representing the I/O pad assignment problem is minimized. The objective function (similar to a quadratic assignment problem) is formulated by using the weights among the I/O's and the distances between pads (distributed at the chip periphery) in such a way that it will lead to the minimum cost for the subsequent placement process. These steps are discussed in the following sections.

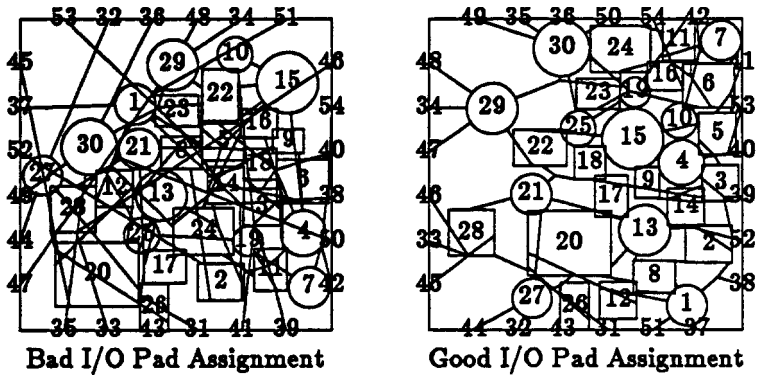


Figure 1. Module placement results with different I/O pad assignments

2. OFF-CHIP I/O PROXIMITY RELATIONSHIPS

The main objective of the I/O pad assignment process is to find an assignment which produces a minimum cost placement when the actual module placement is done. In the initial module placement [1], the objective function to be minimized is the total normalized wire delay of all paths, which can be transformed to a total weighted wire delay as shown below. (Let d_w be delay of wire w and $Dmax_p$ be maximum allowable wire delay of path p which is obtained from the difference between the given maximum delay of path p and total delay of modules on path p.)

$$F = \sum_{\forall \text{ path } p} \left[\frac{1}{Dmax_p} \cdot \sum_{\text{wire } w \in p} d_w \right]$$

$$= \sum_{\forall \text{ wire } w} \left[d_w \cdot \sum_{\substack{\text{path } p \\ \text{passing thru } w}} \frac{1}{Dmax_p} \right]$$

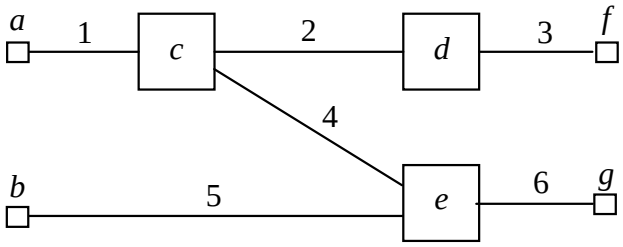
$$= \sum_{\forall \text{ wire } w} \left(d_w \cdot c_w \right)$$

$$\text{where } c_w = \sum_{\substack{\text{path } p \\ \text{passing thru } w}} \frac{1}{Dmax_p}$$

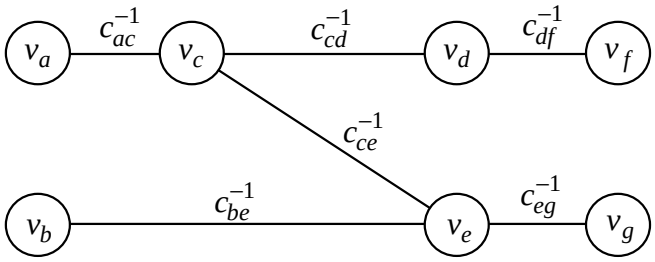
Let n be the number of modules including off-chip I/O's (for simplicity, off-chip I/O's are assumed to be modules) of the network, and M_{io} be the set of off-chip I/O's. Let C and S be $n \times n$ matrices where c_{ij} of C and s_{ij} of S represent the proximity relationships between modules i and j based on the timing constraints and geometrical constraints, respectively. For any connected modules i and j, $c_{ij} = \Sigma\, c_w$ for all the wires w connecting modules i and j. For any non-connected modules i and j, c_{ij} can be computed from the following assumptions. Let u_{ij} be the expected distance between modules i and j where u_{ij} is inversely proportional to c_{ij} (i.e., the stronger the connection strength, the shorter the distance between modules). From the triangular inequality property, for any modules i, j, k $u_{ij} \le u_{ik} + u_{kj}$ (i.e., $c_{ij}^{-1} \le c_{ik}^{-1} + c_{kj}^{-1}$). Setting $c_{ij}^{-1} = \min_{\forall k}\left\{c_{ik}^{-1} + c_{kj}^{-1}\right\}$ satisfies the above distance property so that it can be used for computing the connection strengths of non-connected modules based on timing information.

s_{ij} is a value indicating the maximum allowable distance between modules i and j so that any two modules located on any path from i to j can be placed adjacent to one another. Let L_k be the estimated width of module k (a module is modeled by using a circle whose area is the same as estimated module area, therefore L_k is assumed to be equal to the diameter of the corresponding circle [1]). For any connected modules i and j, $s_{ij} = 0$. For any non-connected modules i and j, $s_{ij} = \min_{\forall k}\left\{s_{ik} + L_k + s_{kj}\right\}$ i.e., when i and j are to be placed, it is preferred that the distance between both modules be as close to s_{ij} as possible.

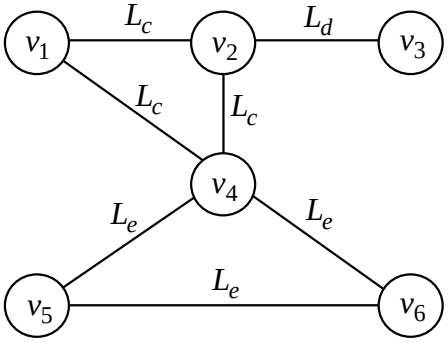
Since only the relationships between off-chip I/O's are needed in the objective function (described later), the matrices C and S, whose elements s_{ij} and c_{ij}, $i \in M_{io}$, $j \in M_{io}$, can be determined by using the shortest path algorithm [10]. For matrix C, let graph $G_c = (V_c, E_c)$ (see Figure 2b). V_c is a set of vertices representing modules. There is an edge between vertices v_i and v_j if there is a signal net connecting modules i and j. Associated with each edge is the value c_{ij}^{-1} where $c_{ij} = \Sigma\, c_w$ for all the wires w connecting modules i and j. Let Vc_{io} be the set of vertices representing all the off-chip I/O's. The problem is to determine the shortest path among all the vertices of Vc_{io}. Therefore, c_{ij} is equal to the reciprocal of the distance of the shortest path from v_i and v_j where i and j are off-chip I/O's.



(a) A module network



(b) Graph G_c



(c) Graph G_s

Figure 2. Examples of graphs G_c and G_s .

For matrix S, let graph $G_s = (V_s, E_s)$ (see Figure 2c). V_s is a set of vertices where v_p is a vertex representing signal p. There is an edge between vertices v_p and v_q if both signals p and q are connected to the same module. Associated with each edge is the width of the corresponding module. Let Vs_{io} be the set of vertices representing signal nets which are connected to I/O's. The problem is to determine the shortest path among all the vertices of Vs_{io}. Therefore, s_{ij} is equal to the distance of the shortest path from v_p to v_q where I/O i connectes to signal p and I/O j connects to signal q.

3. OBJECTIVE FUNCTION

Let P be a set of pads, and U be a $|P| \times |P|$ matrix where an element u_{kl} of U is the Euclidean distance between pads k and l. Given a possible assignment of off-chip I/O's to pads, p, where I/O i is assigned to pad $p(i)$, an objective function, Q, for the I/O pad assignment problem is defined as follows :

$$Q = \sum_{\substack{\forall i,j: i \neq j \\ i,j \in M_{io}}} \left[c_{ij} \cdot W(s_{ij}, u_{p(i)\,p(j)}) \right].$$

Function *W* defined below represents the cost for assignment I/O *i* to pad *p*(*i*) and I/O *j* to pad *p*(*j*) based on the geometrical constraints.

$$W(s_{ij}, u_{p(i)p(j)}) = \left[\frac{u_{p(i)p(j)}}{s_{ij}} \right]^2$$

The above function *W* suggests that if $u_{p(i)p(j)}$ is greater than s_{ij}, it means that some connected modules on a path between off-chip I/O's *i* and *j* cannot be placed adjacent to one another because the distance between the I/O's *i* and *j*, $u_{p(i)p(j)}$ exceeds the maximum allowable distance specified by s_{ij}. Therefore there is a penalty when $u_{p(i)p(j)}$ is greater than s_{ij} (by using the square function).

In order to show that the objective function *Q* of the I/O pad assignment problem has the characteristics close to the ideal objective function, a set of one hundred different I/O pad assignments were fed into the initial module placement process presented in our previous work [1] to obtain module placements based on each of the given I/O pad assignments. Then the costs of the obtained module placements, *F* (each of which is the summation of the normalized total wire delays), were plotted against the costs of the I/O pad assignments, *Q*. Five different examples were used for the experiment. The plots of the *F* and *Q* relationships for all the examples are shown in Figure 3.

An ideal I/O pad assignment objective function should have a monotonically increasing relationship to the cost of subsequent module placement. All of the plots in Figure 3 exhibit an almost ideal relationship. In other words, the I/O pad assignment having the minimum value of *Q* is likely to yield the minimum cost for the subsequent module placement.

4. OPTIMIZATION TECHNIQUE

Given the matrices *S* and *C* along with distance matrix *U*, the I/O pad assignment problem is to find an assignment with minimum cost function *Q*.

$$\underset{\forall i,\ p(i)}{\text{minimize}} \sum_{\substack{\forall i,j: i \neq j \\ i,j \in M_{io}}} \left[c'_{ij} \cdot u^2_{p(i)p(j)} \right]$$

$$\text{where} \quad c'_{ij} = \frac{c_{ij}}{s^2_{ij}}$$

This optimization problem is similar to the quadratic assignment problem. A simple constructive assignment similar to the pair-linking method [11], followed by an iterative improvement method, is used to determine the minimum-cost assignment. There are two operations used during the iterative improvement phase. The first operation is to move an I/O to be adjacent to another I/O when both I/O's connect to the same module. The other operation is to swap the pad positions of two selected I/O's.

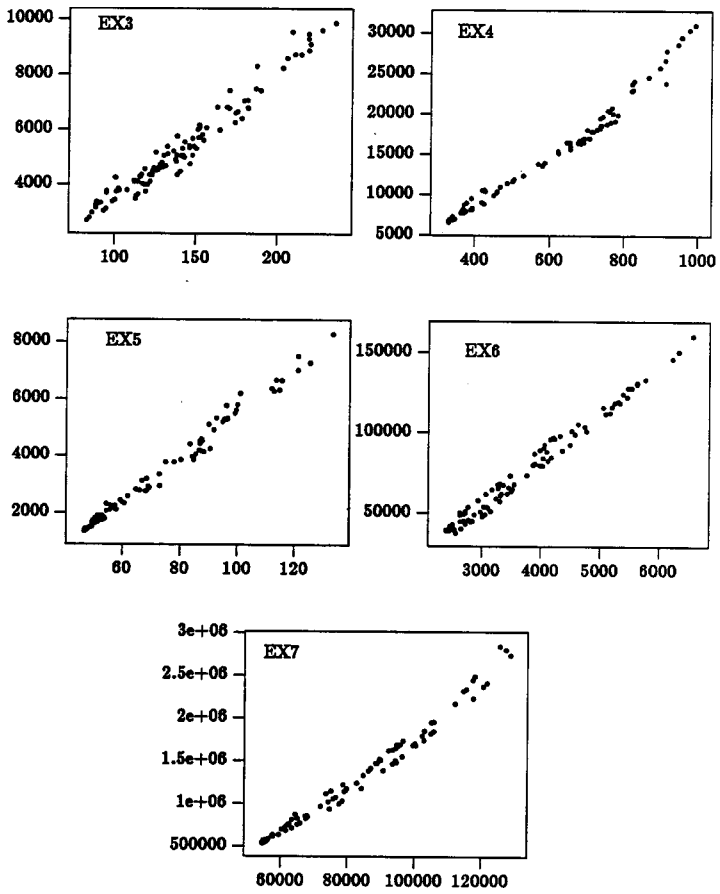


Figure 3. The costs of I/O pad assignment (x-axis) versus
the costs of module placements (y-axis)

The algorithm starts by choosing an I/O which has the maximum c'_{ij} and placing it on a pad if there is no pre-placed pad. Then the next I/O is selected and placed one I/O at a time. The I/O which has the largest number of already placed I/O's connected to the same module is chosen first since we prefer to have I/O's which connect to the same module plcaed close together. In case of a tie, the I/O which has the maximum c'_{ij} is chosen. The selected I/O is placed closed to a placed I/O where their c'_{ij} is maximum in order to minimize the cost function. When all the I/O's are placed, additional iterative improvement is applied to the current assignment by trying to move two I/O's which connect to the same module so that they are placed adjacent to each other. The other iterative improvement operation is to swap any two I/O's. The new assignment is accepted only when its cost is better than the cost of the previous assignment.

```
IO2PadAssignement
{
    if ( there is no pre-placed I/O ) {
        Choose an I/O i having the maximum c'ij
        Assign I/O i to a pad.
```

```
    }
    do {
        Choose I/O i having the maximum number of already placed I/O's
          which connect to the same module as i does.
        If there is a tie, choose i having the maximum c'ij where j is a placed I/O.
        Placed i as close to j as possible where j is a placed I/O and c'ij is max.
    } until ( all the I/O's are placed )
    Iterative improvement by moving operation.
    Iterative improvement by swapping operation.
}
```

5. EXPERIMENTAL RESULTS

In order to show the effectiveness of the optimization procedure described in the last section, a simulated annealing [12] procedure (SA - shown below), optimizing the cost of the I/O pad assignment problem, was implemented. There are two operations used for obtaining the new solution during the SA algorithm. First operation is swapping two I/O's. The other is moving one I/O to be adjacent to another I/O. (This operation involves shifting the location of I/O's located between the two selected I/O's.) Notes that the starting temperature is obtained by swapping random pairs of I/O's for one hundred times and setting the starting temperature to be the average of the cost increase.

```
SA
{
    set the initial random assignment
    Temp ← GetInitialTemp()
    NumTries ← 100 × (# of IO's)
    NumLimit ← 10 × (# of IO's)
    Cost ← ComputerCost()
    do {
        do {
            randomly select two I/O's i and j
            if ( RandomNo() ≤ 0.5 ) {
                if ( RandomNo() ≤ 0.5 )
                    Move i to the right of j
                else
                    Move i to the left of j
            } else {
                Swap positions of  i and j
            }
            NewCost ← ComputeCost()
            ΔE ← NewCost - Cost
```

```
            if ( ΔE < 0 OR RandomNo() < e^(-ΔE/Temp) )
                Cost ← NewCost
            else
                Restore previous assignment
        } until ( #new assignment > NumLimit OR #Loops > NumTries )
        Temp ← 0.97 × Temp
    } until ( no new assignment OR freezing point is reached )
}
```

The same five examples used in the previous section were used here to compare the results obtained using the simulated annealing technique with those obtained using the simple heuristic procedure presented in the last section. The results are shown in Table 1. With far less computation time, the results of the heuristic method are only slightly worse than the results from SA. It was observed that all of the results from SA have all of the I/O's which connect to the same modules placed close to each other, which corresponds to the heuristic used in the proposed procedure.

Ex	#mods	#I/Os	#Nets	Assignement cost (Q)			Time (seconds)	
				SA	Heuristics	%difference	SA	Heuristics
ex3	20	8	29	82.23	82.23	0.00	559.0	0.87
ex4	30	24	63	270.18	270.63	+0.17	7696.5	6.68
ex5	15	8	30	46.34	47.66	+2.86	183.7	0.82
ex6	25	20	53	2390.74	2570.08	+7.50	4671.8	5.00
ex7	20	24	82	54766.52	57378.40	+4.77	7882.1	18.37

Table 1.

6. CONCLUSION

In this paper, a procedure for I/O pad assignment has been presented which is used to assign off-chip I/O's to pads prior to the module placement process. Both timing and geometrical constraints are transformed into connection strengths which represent proximity relationships for the off-chip I/O's of the network. An objective function for the I/O pad assignment problem is proposed and experimentally shown to be almost monotonically increasing with the cost of the module placement. An algorithm for minimizing the objective function using a heuristic constructive assignment followed by iterative improvements was experimentally shown to produce solutions comparable to solutions using the simulated annealing technique, but requiring far less computation time.

7. REFERENCES

1. Prasitjutrakul, S. and Kubitz, W.J., "Path-Delay Constrained Floorplanning: A Mathematical Programming Approach for Initial Placement", *Proc. 26th Design Automation Conf.*, pp.364-369, 1989.

2. Tsay, R., Kuh, E., and Hsu, C., "PROUD: A Fast Sea-of-Gates Placement Algorithm", *Proc. International Conf. on Computer-Aided Design*, pp.318-323, 1988.

3. Wipfler, G., Wiesel, M., and Mlynski, D., "A Combined Force and Cur Algorithm for Hierarchical VLSI Layout", *Proc. 19th Design Automation Conf.*, pp.671-676, 1982.

4. Checng, C. and Khu, E., "Module Placement Based on Resistive Network Optimization", *IEEE Trans. on Computer-Aided Deisgn*, Vol.3, No.3, pp.218-225, 1988.

5. Sha, L. and Dutton, R., "An Analytical Algorithm for Placement of Arbitrarily Sized Rectangular Blocks", *Proc. 22nd Design Automation Conf.*, pp.602-608, 1985.

6. Chi, M., "An Automatic Rectilinear Partitioning Procedure for Standard Cells", *Proc. 24th Design Automaion Conf.*, pp.50-55, 1987 .

7. Kleinhans, J., Sigl, G., and Johannes, M., "GORDIAN: A New Global Optimization / Rectangle Dissection Method for Cell Placement", *Proc. International Conf. on Computer-Aided Design*, pp.506-509, 1988.

8. Ohmura, M., Isumoto, H., and et at., "A New Floorplanning Method with Global Routing Based on Functional Partitioning", *Proc. International Symp. on Circuits and Systems*, pp. 1697-1700, 1988.

9. Pillage, L.T. and Rohrer, R.A., "A Quadratic Metric with a Simple Solution Scheme for Initial Placement", *Proc. 25th Design Automation Conf.*, pp.324-329, 1988.

10. Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs", *Numerische Math*, Vol.1, pp.269-271, 1957.

11. Hanan, M. and Kurtzberg, M., "Placement Techniques", *Design Automation of Digital Systems*, Vol.1, ed. M.A. Breuer, Prentice Hall Inc., pp.324-342, 1972.

12. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing", *Science*, Vol.220, No.4598, pp.671-680, May 1983.