I/O Pad Assignment based on the Circuit Structure*

Massoud Pedram, Kamal Chaudhary, Ernest S. Kuh

Electronics Research Laboratory
University of California, Berkeley, CA 94720

Abstract

We present an algorithm for assigning off-chip I/O pads for a logic circuit. The technique which is based on the analysis of the circuit structure and path delay constraints, uses linear placement, goal-programming, linear-sum assignment and I/O pad clustering to assign locations to I/O pads. The I/O pad assignment is then used by placement tools. Experimental data shows that as a result of using our I/O pad assignment procedure, the total interconnection length and circuit delay (after placement and routing) are reduced by 8-15% and 3-4% respectively.

1 Introduction

Most CAD systems do the design of electronic systems in a hierarchical top-down fashion where logic synthesis is followed by physical design. Decisions during the logic synthesis (including partitioning logic circuits into an interconnection of combinational logic components and registers, logic optimization, binding optimized logic equations into gates in a target library) are difficult because their effects on the final layout are hard to predict and may not become apparent until much later in the design process. Once any parameter at the layout design stage fails to satisfy a constraint imposed on it, the logic synthesis must be modified (even repeated) so as to accommodate the constraint. (This may become necessary because physical design is too far down in the design pipeline to solve performance issues that were not considered earlier.) The new change may cause some other constraint to be violated and the process must be repeated. In order to reduce the number of design iterations, explore larger portions of the design space, and find better quality solutions in shorter CPU time, physical design must be integrated with logic synthesis.

The key to incorporating layout aspects into logic synthesis is the generation of a placement of the multi-level Boolean network and using this placement to guide the decomposition and mapping processes [5, 6]. However, when using force-directed and mathematical programming approaches to solve the placement problem, positions of the off-chip I/O pads must be known prior to placing the gates [1, 3, 7]. This is because in the absence of off-chip I/O pads, the gates collapse to the center of chip. At the same time, different I/O pad assignments give rise to placements with different qualities. In particular, significant improvements in area and wire length can be obtained by doing a good I/O pad assignment.

One approach is to treat I/Os as floating gates and use a force-directed approach to assign positions to all gates. I/Os are later assigned to fixed pads [9]. The problem with this approach is that during the placement phase, I/Os are allowed to float (and hence assume infeasible positions) and, therefore, when they are moved to fixed pad positions, the quality of placement solution becomes questionable. This problem is more severe if all pads are constrained to be at the chip boundary. Another common approach is to use an arbitrary I/O pad assignment prior to placement and then improve the pad locations based on the detailed placement result. The two phases may be iterated until an acceptable placement solution is generated. This approach is also undesirable since even if convergence is achieved, the final solution is heavily influenced by the initial pad assignment which was arbitrary. In addition, the iteration process is costly and time-consuming.

The I/O pad assignment becomes even more important if there are path-delay constraints from primary inputs to primary outputs. In that case, the initial location of the I/O pads will greatly influence the quality of timing driven placement obtained. In particular, a poor pad assignment may result in an infeasible placement solution.

In this paper, we present an I/O pad assignment procedure which is based on the analysis of circuit structure (in logic equation or directed net list forms) and path-delay constraints. This procedure can be invoked prior to logic optimization and/or final gate placement. In either case, the pad assignment result is used as input to placement tools. The resulting placement can then be used either to guide logic decomposition/restructuring and technology mapping procedures or is followed by routing to generate the final layout.

Our I/O pad assignment technique can be summarized as follows. Initially, we order the primary outputs in order to maximize the proximity between their transitive fanin cones by formulating and solving a linear placement problem. We then distribute the output pads on the chip boundary and calculate *goal distances*

*This work was supported by the National Science Foundation, under Grant No. MIP 88-03711 and by the Semiconductor Research Corporation under Grant No. 91-DC-008.

for each primary input - primary output pair based on the analysis of circuit structure and / or path-delay timing constraints. Next, we introduce a number of slots on the chip periphery and assign primary inputs to slots such that the sum over all primary input - primary output pairs of violations of the goal distances is minimized. We use linear-sum assignment technique to solve this problem efficiently and concurrently. In order to assure that primary inputs which are connected to the same gates are assigned positions near each other, we cluster these inputs together and assign input clusters to slots.

The paper is organized as follows. Section 2 gives some definitions and and describes our timing model. Sections 3 and 4 present detailed descriptions of the basic approach followed by timing-driven extensions. Sections 5 and 6 contain our experimental results and conclusions.

2 Terminology

We assume that the circuit is specified in the form of a directed acyclic graph (DAG), that is, a *Boolean network* prior to logic optimization or a *directed net list* prior to gate placement. We therefore give some terminology and definitions relevant to directed graphs followed by statement of our assumptions and timing model.

A path in a directed graph is an open walk with no repeated vertices which follows the edge orientations. A (u, v) path is a path with u and v as endvertices. A (u, v) *bidirected* path consists of exactly two sub-paths (either $((u, z) \wedge (v, z))$ or $((z, u) \wedge (z, v)))$ for some vertex z. Distance $d(u, v)$ is the minimum number of edges in a (u, v) path. Bidirected distance $b(u, v)$ is the minimum number of edges in a (u, v) bidirected path. In both cases, if there is no such path, distance is set to ∞.

A node u is a *fanin* of a node v if there is a directed edge e_{uv} from u to v and a *fanout* if there is a directed edge evu. A node u is a *transitive fanin* of a node v if there is a (directed) path from u to v and a *transitive fanout* if there is a (directed) path from v to u. *Primary inputs* are inputs of the directed graph and *primary outputs* are its outputs. *Internal nodes* are nodes of the directed graph with at least one fanin and one fanout. The *primary input support* of a node u is the set of primary inputs that are transitive fanins of u.

We assume that each internal node has an exact (in case of gates in a net list) or estimated (in case of unmapped nodes of a Boolean network) area. The average dimensions of an internal node can therefore be calculated.

Consider a node u and let v be its fanout node. The delay through u for a signal transition at one of its inputs is given as

$$\tau + R\,(C_{gate} + C_{wire})$$

where τ is the intrinsic delay through u, R is the output resistance of u, C_{gate} is the input capacitance of v and C_{wire} is the propagation delay through edge e_{uv} and is given by

$$C_{wire} = C_H\ |x_u - x_v| + C_V\ |y_u - y_v|.$$

C_H and C_V denote the horizontal and vertical capacitance per unit length of the horizontal and vertical interconnect wires respectively and (x_u, y_u) and (x_v, y_v) denote the positions of nodes u and v

If the node is a gate in the library, its intrinsic delay, input pin capacitance, and output resistance are known. Otherwise, they are estimated as in [8]. That paper presents a simple model for estimating the delay of a multi-level combinational logic description prior to technology-dependent mapping. The model proposes that delay through a node varies logarithmically with both the complexity and the fanout of the node's logic equation. The input pin capacitance for a node is taken to be equal to that of a 2-input NAND gate in the target library and the drive capability for the output pin is equal to that of an inverter.

3 The Basic Approach

For each primary output po_i, we traverse the circuit in a depth-first order and identify its primary input support. We then derive a linear ordering on the primary outputs maximizing proximity among their primary input supports as explained below. Corresponding to each primary output po_i, we create a block B_i. Corresponding to each primary input pi_j in the primary input support of po_i, we create a pin p_j and attach it to block B_i. (Primary inputs which belong to the primary input support of exactly one primary output are ignored since they do not affect the proximity metric for primary outputs.) Consequently, we generate a net list representing primary outputs and their primary input supports. We then obtain a linear placement of blocks B_i which minimizes the total net span. This solution corresponds to a linear ordering on the primary outputs which maximizes proximity among their primary input supports. (See Figure 1.)

Next, we calculate the bidirected distance between each consecutive pair in the ordered list of primary outputs by performing depth first search from outputs toward the inputs. The total bidirected distance from the leftmost to the rightmost primary output in the ordered list is normalized to chip boundary length and the primary outputs are distributed on the chip periphery accordingly. The idea is that if the bidirected distance between a pair of outputs is small, the two outputs should be placed near one another. If the bidirected distance between a pair of outputs is ∞, then the two outputs can be placed anywhere with respect to one another. In particular, we place them near one another.

After assigning initial positions to the output pads, we proceed to assign the input pads. Let M denote the number of primary inputs in the circuit. We transform the pad placement problem into a linear-sum assignment problem as follows. We put S (θ times the number of floating I/Os where $\theta \geq 1.0$) slots on the circuit boundary and construct an $M \times S$ linear assignment cost matrix C. Entry (i, k) in matrix C represents the cost of assigning primary input pi_i to slot s_k. This

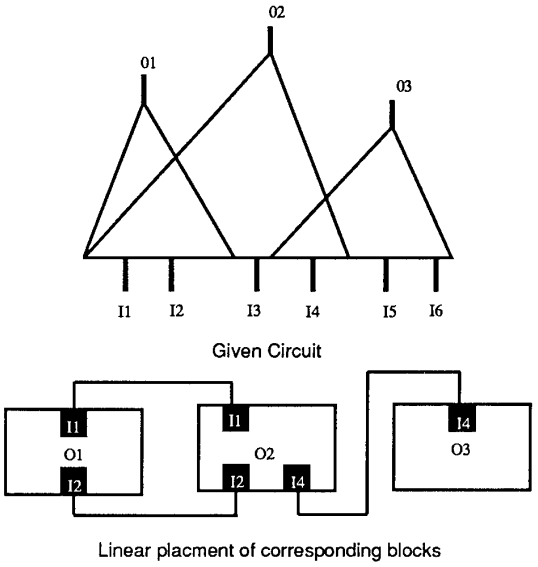

Figure 1: Output pad ordering based on linear placement

cost is calculated as follows:

$$c(i,k) = \sum_{j \in po's} \left(1 - \frac{h(j,k)}{d(i,j)}\right)^2$$

where $h(j,k)$ is the half perimeter length of the bounding box enclosing primary output po_j and slot s_k. $d(i,j)$ is equal to the distance from pi_i to po_j. It has been converted into Manhattan length using the average dimensions for an internal node.

After running the linear assignment solver on matrix C, we obtain a minimum sum-cost solution to the input pad assignment problem, i.e., a subset X of entries c_{pq} of matrix C is chosen such that the following holds:

$$\begin{aligned} &\forall i \ \exists j^\star : \ c^{ij^\star} \in X, \\ &if \ \ i_1 \neq i_2 \ \ then \ \ j_1^\star \neq j_2^\star, \\ &\sum_i c^{ij^\star} \ \ is \ \ minimum. \end{aligned}$$

Since rows in the cost matrix C correspond to floating input pads and columns correspond to the slots, the linear assignment determines input pad assignment with the minimum cost.

In order to insure that primary inputs which are connected to the same gates are assigned positions near each other, we cluster these inputs together and, during the linear assignment phase, assign the input clusters to slots. In particular, primary inputs whose pairwise bidirected distances are less than or equal to l are clustered together. (l is set to be a small fraction of L, the number of levels in the DAG.) For example, in Figure 2, inputs A through E will be clustered together for $l \geq 5$.

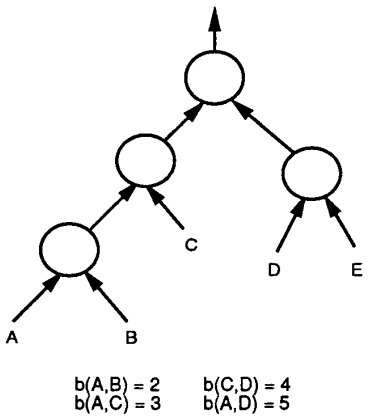

Figure 2: Input pad clustering based on bidirected distances

4 The Timing Driven Approach

We assume that the required times at the primary outputs and the arrival times at the primary inputs of the functional block are given. The timing slack on each path is used to estimate the wire length that can be accommodated on that path as outlined below.

Let $a(i,j)$ be the difference between the required time at po_j and the signal arrival time at primary input pi_i (i.e., the allowed path delay), and, $g(i,j)$ be the longest path delay from pi_i to po_j calculated recursively as in [4]. $g(i,j)$ does not include the wiring delay contribution, that is, $C_{wire} = 0$ in the delay equation. Now, let $w(i,j) = a(i,j) - g(i,j)$ represent the maximum delay that can be allocated to signal propagation through wires connecting gates (that lie on paths from pi_i to po_j) without violating the timing constraints. It is translated to a Manhattan length by using the values of C_H and C_V from the technology file.

Let $h(j,k)$ be the half perimeter length of the bounding box enclosing primary output po_j and slot s_k, and $d(i,j)$ be the distance from input pi_i to output po_j. We convert $d(i,j)$ to units of Manhattan length as in Section 3. Then, the cost function is defined as (Figure 3):

$$c(i,k) = \sum_{j \in po's} t(i,j,k)$$

$$t(i,j,k) = \begin{cases} d(i,j) - h(j,k) \\ \quad \text{if } h(j,k) \leq d(i,j) \\ 0 \\ \quad \text{else if } d(i,j) < h(j,k) \\ \quad \leq d(i,j) + w(i,j) \\ (h(j,k) - d(i,j))^2 \\ \quad \text{otherwise} \end{cases}$$

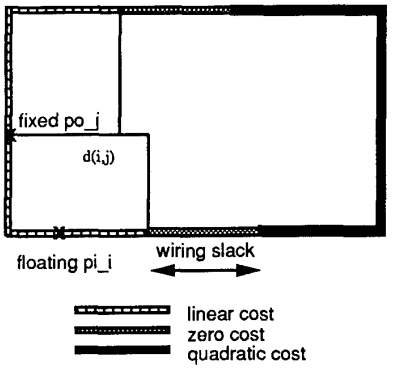

Figure 3: Cost function for timing-driven I/O pad placement

The same initial input clustering and output pad distribution followed by linear assignment can be used to solve the timing-driven pad placement.

5 Experimental Results

The techniques described here have been implemented in a computer program PACT (Pad Assignment based on Circuit sTructure). PACT is written in C and has been incorporated into LILY [5, 6]. We have run PACT on several MCNC logic circuit benchmarks [10] (after logic optimization and technology mapping). Table 1 shows some characteristics of the benchmarks. We compared PACT results with those of *random* and *clockwise* I/O pad assignment procedures provided in Octtools release 5.1. (We could not compare PACT with other I/O pad assignment procedures due to lack of access to the tools.) The first procedure randomly assigns a side and position along the side to each pad. It was repeated 100 times for each example with different seeds and the average wire lengths are tabulated. The second procedure selects an output pad and assigns that output and all primary inputs in its support set to the pads around the chip boundary in a clockwise fashion. It then processes the next output and so on. The procedure is very sensitive to the order in which outputs are picked. We ran the clockwise procedure 20 times with different output orderings and report the average.

The tabulated data are collected after pad assignment, detailed placement by Gordian placement package [3], and global and detailed routing tools of Octtools. The area, wire length and worst case path delay are based on the placed and routed circuits and include the delay through interconnecting wires. The circuits were optimized and mapped using MIS-II [2]. We used a library similar to MCNC standard cell library with modified timing parameters so that these parameters match those of a realistic 1 micron library. (We set $C_H = C_V = 2.5pF/cm$.) Table 2 shows the total interconnection of the benchmark circuits. We obtained an average reduction of 15.1% and 8.4% over random and clockwise procedures respectively.

Results showing effect of the I/O pad assignment on the circuit speed are tabulated in Table 3. We obtained an average improvement of 4.1% and 3.1% over random and clockwise procedures respectively. Note that the Gordian placement package does not have a timing-driven capability, therefore, Table 3 does not truly reflect the impact of I/O pad assignment on the circuit delay. Furthermore, the drive capability of a pad is much higher than that of a gate, and a good placement tool can reduce the longest path delay by allowing the pads to drive longer wires on the critical paths.

PACT run times are short (e.g., 8 seconds for output ordering, 33 seconds for linear assignment on C880 benchmark on a DEC3100 Workstation).

6 Conclusions

We have presented an I/O pad assignment technique for assigning I/O pads based on analysis of the circuit structure and path delay constraints. Both of these considerations are transformed into proximity relationships among the off-chip I/Os. A cost function which penalizes violations of these proximities is defined and linear assignment technique is used to simultaneously assign I/O pads to slots. This technique is general and can handle I/O pad assignment prior to logic synthesis or detailed placement procedures.

References

[1] A. Srinivasan, K. Chauhary and E. S. Kuh, "RITUAL: An algorithm for performance-driven placement of cell-based ICs," *Proc. Third Physical Design Workshop*, May 1991.

[2] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli and A. Wang, "MIS: Multiple-Level Logic Optimization System," *IEEE Trans. on CAD*, Nov. 1987, pp. 1062-1081.

[3] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. on CAD*, vol 10, no. 3, pp. 356-365, March 1991.

[4] R. B. Hitchcock, G. L. Smith, and D. D. Cheng, "Timing analysis of computer hardware", *IBM J. Res. Develop.*, Vol. 26, No. 1, January 1982, pp. 100-105.

[5] M. Pedram and N. Bhat, "Layout driven technology mapping," *Proc. 28th Design Automation Conference*, 1991, pp.

[6] M. Pedram and N. Bhat, "Layout driven logic restructuring / decomposition," *To appear in Proc. Int. Conf. CAD (ICCAD-91)*, 1991.

[7] R. S. Tsay, E. S. Kuh, and C. P. Hsu, "PROUD: A sea-of-gates placement algorithm," *Proc. Int. Conf. CAD (ICCAD-88)*, pp. 318-323.

[8] D. E. Wallace, M. S. Chandrasekhar, "High-Level delay estimation for technology independent logic equations," *Proc. Int. Conf. CAD (ICCAD-90)*, 1990, pp. 188-191.

[9] G. J. Wipfler, and D. A. Mlynski, "An automatic placement procedure for integrated circuits," *Proc. Int. Symp. on Circuits and Systems*, 1985, pp. 13-16.

[10] "Logic synthesis and optimization benchmarks - user guide," *Microelectronics Research Center of North Carolina*, 1988.

Example	# gates	# inputs	# outputs
C1355	210	41	32
C1908	244	33	25
C3540	589	50	22
C432	127	36	7
C5315	588	178	123
C880	221	60	26
bw	93	5	28
duke2	235	22	29
e64	185	65	65
misex2	60	25	18
misex3	300	14	14
rd84	87	8	4

Table 1: Examples (# gates after mapping)

Example	Total Net Length			% Improvement over	
	random	clockwise	PACT	random	clockwise
C1355	210.6	203.8	184.5	12.4	9.8
C1908	248.2	238.1	224.0	9.8	5.9
C3540	1215.2	1135.4	1047.0	13.8	7.8
C432	112.0	100.8	88.3	21.2	11.6
C5315	1823.2	1716.3	1423.3	21.9	17.0
C880	252.3	227.1	192.8	23.6	15.1
bw	71.2	66.1	63.5	10.8	3.9
duke2	339.5	326.0	300.4	11.4	7.9
e64	152.7	139.2	121.2	20.6	12.9
misex2	41.9	37.6	35.1	16.2	6.7
misex3	459.4	433.0	415.7	9.5	4.0
rd84	52.7	51.2	49.3	6.5	2.0

Table 2: Wiring Results for I/O PAD Assignment (wire lengths in millimeters)

Example	Delay without wiring	Delay with wiring			% Improvement over	
		random	clockwise	PACT	random	clockwise
C1355	13.6	17.5	17.2	15.9	9.7	7.6
C1908	20.3	25.6	25.4	24.5	4.3	3.5
C3540	29.9	39.1	38.7	38.1	2.6	1.5
C432	21.6	25.8	25.3	25.2	2.4	0.4
C5315	20.1	27.1	26.3	26.2	3.3	0.4
C880	23.5	28.2	27.8	26.2	7.1	5.8
bw	23.9	27.4	27.8	27.1	1.0	2.5
duke2	18.4	24.2	24.0	23.3	3.7	2.9
e64	41.8	46.1	46.4	45.0	2.4	3.0
misex2	7.5	8.2	8.2	8.1	1.2	1.2
misex3	16.4	22.9	22.8	21.1	7.9	7.5
rd84	11.1	12.6	12.3	12.2	3.2	0.8

Table 3: Timing Results for I/O PAD Assignment (delays in nano-seconds)