PLA 열 또는 행의 최적 겹침쌍을 찾기위한 3 단계 휴리스틱 알고리즘

양 영 일°, 경 종 민
전기 및 전자공학과, 한국과학기술원

A Three-Step Heuristic Algorithm For Optimal PLA Column and/or Row Folding

Yeong-Yil Yang and Chong-Min Kyung
Department of Electrical Engineering, KAIST

Abstract

A three-step heuristic algorithm for PLA column folding and row folding of column-folded PLA is presented, which is significantly faster than the earlier works and provides nearly optimal results. The three steps are i) min-cut partition of vertices in the column (or row) intersection graph, ii) determination of products' order using Fiduccia's min-net cut algorithm, and iii) head-tail pairing for column folding, while some heuristics are proposed for deciding row folding pairs. The time complexity of this algorithm is $O(n^2 \log n)$ compared to the $O(n^3) - O(n^4)$ of the earlier works.[2][3][9] For a test PLA with 23 inputs, 19 outputs and 52 products, the number of column folding pairs obtained using this algorithm is 20 which is optimal, as compared to 17 in a previous work.[3]

1. Introduction

In the PLA folding, we allow inputs and/or outputs to arrive at either top or bottom of the PLA, so a physical column can be shared between a pair of inputs or outputs but not between an input and output. The PLA folding problem has been proven to be NP-complete,[1] and several heuristic algorithms[2][3] have been developed which find folding pairs one by one. The PLA folding results thus obtained are only locally optimal and dependent on the selection order of the folding pairs. On the other hand, branch and bound algorithm[4][5] can find optimal solutions theoretically, but the algorithm's time complexity cannot be predicted because of the backtracking.

In this paper, we propose a fast and near-optimal heuristic algorithm for minimizing the area of the PLA by means of simple column folding[2] and row folding of the column-folded PLA.[2][5] This algorithm can escape from the local minima by simultaneously determining the globally optimal positions of each row (or column) in PLA column (row) folding using min-cut partition[6] and min-net cut partition[7] algorithms. An algorithm called head-tail pairing is finally used for finding column folding pairs among many candidate pairs, which can be easily proven to be optimal in terms of the number of folding pairs generated. Moreover, the proposed algorithm has $O(n^2 \log n)$-time complexity due to the use of Kernighan-Lin algorithm,[6] which is lower than the $O(n^3)$[2][3] $-$ $O(n^4)$[9]-time complexity of the earlier algorithms.

2. Simple Column Folding

2.1 Min-Cut Bi-Partition of Input/Output Columns

Two columns c_i and c_j are disjoint if there are no rows which have connection with both c_i and c_j in the personality matrix. Let $R(c_i)$ be the set of rows which have connection with column c_i. Column intersection graph $G = (V, E)$ has been proposed,[2] where the vertices are the one-to-one correspondence with the columns of the personality matrix of the PLA. The set E is defined as $E = \{e = (v_i, v_j) | R(c_i) \cap R(c_j) \neq \phi\}$. In column-folded PLA, two inputs (and/or outputs) that share a column arrive from different directions, top or bottom. Inputs (and/or outputs) can be classified into two groups, one arriving from the top (upper group) and the other arriving from the bottom(lower group).

Fig. 1(a) shows the personality matrix of an example PLA whose column intersection graph is shown in Fig. 1(b). The vertices on the column intersection graph are partitioned into two groups of the same size, upper group and lower group using the min-cut algorithm.[6] Fig. 1(c) shows a bipartite column intersection graph, where shown are the edges between two groups, $\{I_1, I_3, I_5, O_1, O_3\}$ and $\{I_2, I_4, I_6, O_2, O_4, O_5\}$.

2.2 Product Order Determination Using Min-Net Cut Algorithm

The PLA personality matrix can be modeled as a network by noting that each column c_i corresponds to a net n_i and each row r_j corresponds to a cell m_j. In this network model, a net n_i has connection with cells which represent rows in $R(c_i)$. Dummy cells, m_{top} and m_{bottom}, are introduced to the network model such that

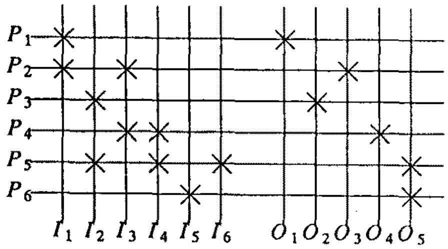


Fig.1(a) Personality matrix of the PLA

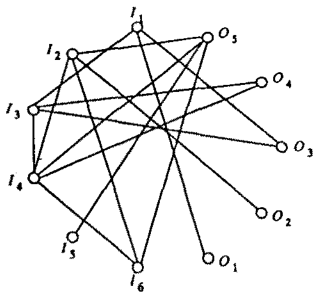


Fig.1(b) Column intersection graph

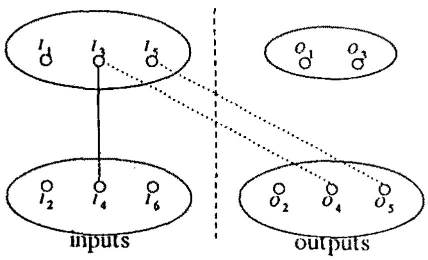


Fig. 1(c) Bipartite column intersection graph

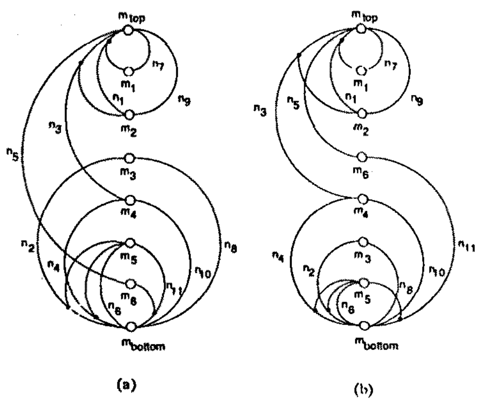


Fig. 2 A network model

nets which represent columns of the upper group have additional connection with m_{top} and nets which represent columns of the lower group have additional connection with m_{bottom}. A network model corresponding to the product line arrangement of Fig. 1(a) with the column bi-partition of Fig. 1(c) is shown in Fig. 2(a).

Cells from m_1 to m_5 of the above models are linearly ordered by the min-net cut algorithm[7] which minimizes the total number of net cuts at all cut lines between each consecutive cells. In ordering the products, the positions of m_{top} and m_{bottom} are fixed, and other cells are successively partitioned into two subsets of ≤ 1 size difference until the size of each subset is 1. Final result of product ordering is shown in Fig. 2(b).

2.3 Head-Tail Pairing

With the positions of each product line determined, we define the l-value and u-value as follows:

Definition) The lower extreme y-coordinate value of an input or output column in the upper group among the y-coordinate values of all cross-points of the column in PLA personality matrix is defined as l-value. Similarly, u-value of a column in the lower group denotes such upper extreme y-coordinate value.

The l-values of inputs (outputs) in the upper group and the u-values of inputs (outputs) in the lower group are sorted separately in non-increasing order and inserted into A-list and B-list, respectively. The so-called head-tail pairing, which can be easily shown to be optimal in terms of the number of folding pairs generated, performs pairing the elements at the front of the A- and B-list if l-value of the element in the A-list is greater than u-value of the element in the B-list. If paired, they are popped out from the A- and B-list. Otherwise, only the element in the B-list is popped out, and the similar pairing procedure is performed until either A- or B-list becomes empty. Final result of the PLA column folding is shown in Fig. 3.

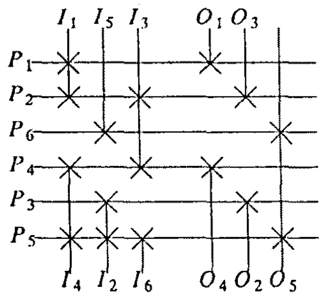


Fig. 3 Personality matrix of column-folded PLA

3. Row Folding of Column Folded PLA

3.1 Products Bipartition and Column Order Determination

The row folding of column-folded PLA is similar to the simple column folding mentioned in section 2 up to and including the second step if some modifications are made. Min-cut and min-net cut algorithms are similarly applied to determine the order of the folded or unfolded columns from left to right. The third step of deciding folding pairs in the row folding is different from that of column folding due to the positional constraint between the product lines set by the column folding.

3.2 A Heuristic for Finding Row Folding Pairs

In the column folding, if two columns c_i and c_j are an ordered column folding pair, the rows in $R(c_i)$ should be above the rows in $R(c_j)$. Row constraint graph,[2] $D(V,A)$ represents the constraints on the relative positions of rows due to the column folding. In the row constraint graph, $D(V,A)$, V represents the set of vertices (rows); and A represents a set of directed edges (constraints on the relative positions of rows) $A = \{(v_i, v_j) \mid v_i \in R(c_i), v_j \in R(c_j), (c_i, c_j) \in F\}$, where F is the set of column folding pairs. Row constraint graph is a dag (directed acyclic graph). A salient feature of the head-tail pairing used for column folding is that the number of directed edges in the row constraint graph is minimized, because it is likely that the column having many crossings with the product lines becomes a folding pair with the column with having few crossings with the product lines, and vice versa.

Continuing with the example of Fig. 3, products p_1, p_2, p_4 come from the left and products p_3, p_5, p_6 come from the right as the result of min-cut partition. Fig. 4(a) shows the row contraint graph having directed edges to represent the column folding in Fig. 3. Row constraint graph can be partitioned into two separate graphs by deleting the edges connecting two groups. Reduced row constraint graph is obtained as in Fig. 4(b), where the constraints on the relative vertical positions among the product lines in each group are separately shown as directed edges. Finally, the row folding graph shown in Fig. 4(c) is formed by connecting all foldable pairs by undirected edges. Following is an algorithm for finding row folding pairs using the row folding graph.

Step 1) Set ACT-VERTEX, ACT-EDGE and FOLD-LIST empty. Insert vertices whose in-degree $d_D^-(v)$ is zero into ACT-VERTEX.

Step 2) Insert edges whose ends are in ACT-VERTEX into ACT-EDGE.

Step 3) If ACT-EDGE is empty, select vertex v_i such that the number of vertices in ACT-VERTEX having a directed path to vertex v_i is minimal; insert v_i into ACT-VERTEX and go to step 2.

Step 4) Select an edge from the edges in ACT-EDGE according to the edge selection criteria and move it into FOLD-LIST. Insert the child vertices of ends of the selected edge into ACT-VERTEX. If ACT-VERTEX is empty, then STOP, else go to step 2.

Following are the edge selection criteria in the order of priority.

i) Select an edge such that the number of vertices in ACT-VERTEX which have a directed path to the ends of the edge is minimum.

ii) Select an edge joining two vertices (rows) with minimal difference of extreme values. Edges corresponding to the row folding pairs (v_j, v_k) and (v_i, v_l) are selected as shown in Fig. 5(a).

iii) Select an edge which connects to a vertex of degree 1. Edge a is selected before b as shown in Fig. 5(b).

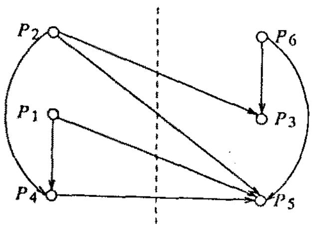


Fig. 4(a) Row constraint graph

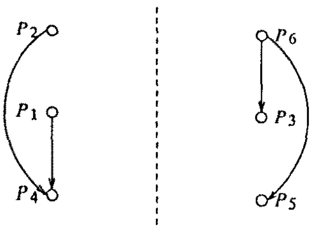


Fig 4(b) Reduced row constraint graph

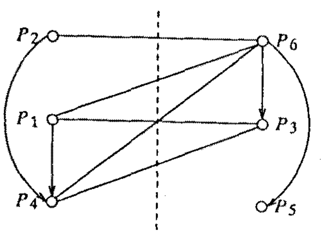
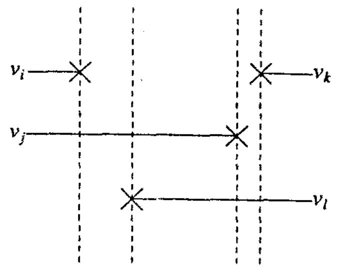
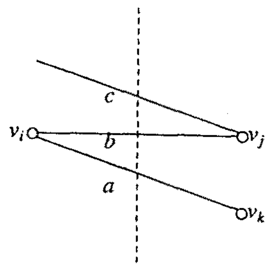


Fig. 4(c) Row folding graph



(a)



(b)

Fig. 5 Edge selection sequence

Fig. 6 shows the result of PLA row folding of column folded PLA according to this procedure.

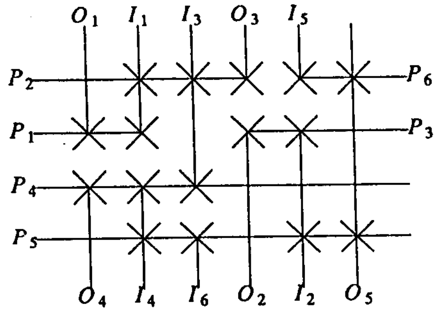


Fig. 6 Result of row folding of column-folded PLA

4. Experimental Results and Conclusions

The proposed algorithm for PLA column folding and row folding of column-folded PLA was implemented in C on MV10000 running on DG/UX. Table 1 shows the results of simple column folding. In ref. [3], the number of column pairs was 17 for the PLA #5 in the Table 1, compared to 11+9=20 folding pairs in our result, which is optimal for this PLA. 16 or 17 folding pairs was obtained in ref. [9] using $O(n^4)$-time complexity algorithm for the PLA #4 in Table 1, whereas our algorithm with $O(n^2 \log n)$-time complexity obtained 16 folding pairs in 1.9 sec CPU time. The results of row folding of column-folded PLA are shown in Table 2. Fig. 7 shows the result of PLA #2 in Table 2. In conclusion, we proposed a fast heuristic PLA folding algorithm applicable for column folding and row folding of column-folded PLA, which yields nearly optimal results for almost all examples tried.

	PLA Size			Results		
no	In/Out	Product	Sparsity	IFP	OFP	CPU(sec)
1	10/10	14	18.6	5	5	0.9
2	20/20	16	18.0	10	8	1.6
3	23/17	21	23.6	9	8	2.1
4	23/15	21	20.4	11	5	1.9
5	23/19	52	12.0	11	9	5.0
6	49/25	30	19.1	24	11	4.6
7	64/50	80	17.2	32	25	25.9

Table 1. Examples of simple column folding.
Sparsity : percentage of X's among whole grid points in the PLA matrix
IFP : number of input folding pairs
OFP : number of output folding pairs

	PLA Size			Results			
no	In/Out	Product	Sparsity	IFP	OFP	PFP	CPU(sec)
1	11/11	15	15.8	5	5	6	1.1
2	22/19	50	10.2	11	9	14	8.7
3	21/18	52	10.5	10	9	15	9.2
4	27/23	60	9.1	13	11	15	11.3

Table 2. Examples of row folding of column-folded PLA
PFP : number of product line folding pairs

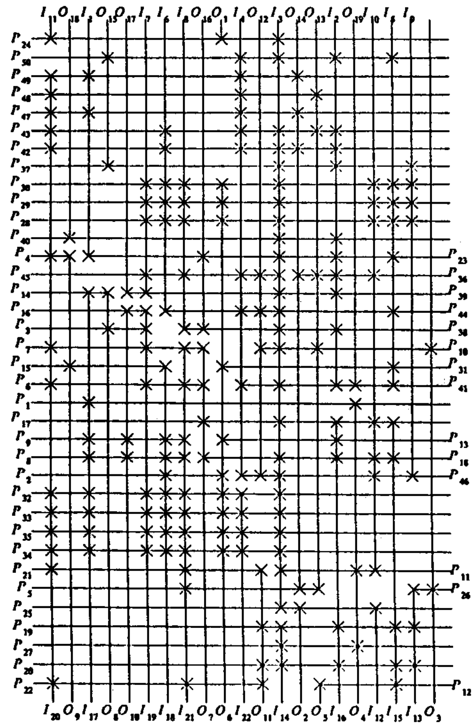


Fig. 7 Row folding of column-folded PLA

References

[1] G. D. Hachtel, A. R. Newton, and A. L. Sangiovanni-Vincentelli, "Some results in optimal PLA folding," *Proc. Int. Circuits and Comput. Conf.* pp. 1023–1027, Oct. 1980

[2] G. D. Hachtel, A. R. Newton, and A. L. Sangiovanni-Vincentelli, "Techniques for programmable logic array folding," *Proc. 19th Design Automation Conference*, pp. 147–155, 1982.

[3] G. D. Hachtel, A. R. Newton, and A. L. Sangiovanni-Vincentelli, "An algorithm for optimal PLA folding," *IEEE Trans. Computer-Aided Design* **CAD-1**, pp. 63–76, Jan. 1982.

[4] W. Glass, "A depth-first branch and bound algorithm for optimal PLA folding," *Proc. 19th Design Automation Conference*, pp. 133–140, 1982.

[5] J. L. Lewandowski and C. L. Liu, "A branch and bound algorithm for optimal PLA folding," *Proc. 21st Design Automation Conference*, pp. 426–433, 1984.

[6] B. W. Kernighan, S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal* vol.49, pp. 291–307, Feb. 1970.

[7] C. M. Fiduccia and R. M. Mattheys, "A Linear-Time Heuristic for improving Network Partitions," *Proc. 19th Design Automation Conference*, pp. 175–181, 1982.

[8] Qingjian Yu and Omar Wing, "INTERVAL-GRAPH-BASED PLA FOLDING," *Proc. Int'l Symposium on Circuits And Systems*, pp. 1463–1466, 1985.

[9] S. Y. Hwang, R. W. Dutton, T. Blank, "A Best First Search Algorithm for Optimal PLA Folding," *IEEE Trans. Computer-Aided Design* **CAD-5**, pp. 433–442, Jan. 1986.