Enhanced heuristic for multichannel optimization in gate array layout

L Lin*, S Sahni and E Shragowitz

Enhancements to the multichannel optimization heuristic proposed by Aoshima and Kuh[1] for gate array layout are presented. These enhancements result from the introduction of spacing and windowing concepts.

computer-aided design, multichannel optimization, gate arrays

In the multichannel optimization problem[1], a chip in which the cells (or macros) are placed in uniformly separated rows is given (Figure 1). There is a horizontal routing channel between each pair of adjacent cell rows. Vertical routing is performed on the second layer. Each cell row has uniformly spaced terminals on either side. Terminals are labelled $-1, 0, 1, 2, \ldots, k$, where k is the number of nets. If a terminal is labelled i, $1 \leqslant i \leqslant k$, then it is a terminal of net i. A blocked terminal is labelled -1 and a feedthrough location is labelled 0. This model is the same as that used for standard cell and polycell LSI placement and routing[2]. However, in the case discussed here the placement has already been carried out.

The objective of the multichannel optimization problem is to decompose the nets in such a way that they can be routed using horizontal routing channels of least capacity (all channels are assumed to have the same capacity). Each net is decomposed into a set of subnets such that each subnet consists solely of terminals on either side of the same routing channel and the union of the subnets is the original net. Following this decomposition, each routing channel is routed independently of the others using a classical channel router. When a channel is being routed, all subnets with end points on that channel are considered.

Since the output of the multichannel optimization problem is the input for classical channel routing, and since channel routers are generally able to route using as many tracks as the channel density[3,4], it is reasonable to modify the objective of the multichannel optimization problem to the following: 'decompose the nets so that the maximum density in any channel is a minimum'.

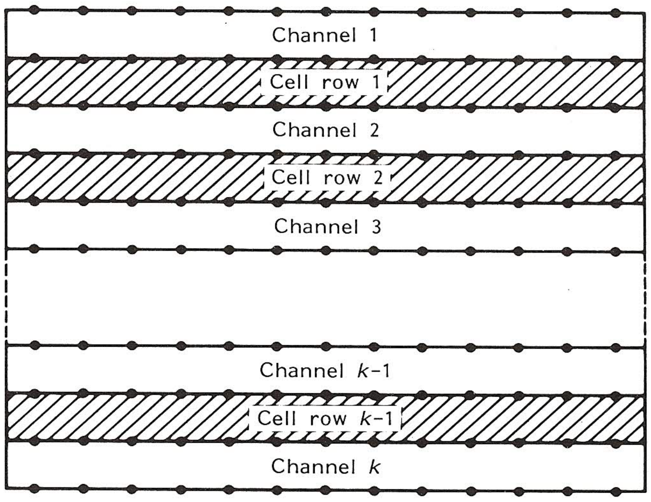

Figure 1. Chip with cells placed in uniformly separated rows

For this objective function, Aoshima and Kuh[1] have proposed two heuristics, namely, optimum decomposition and practical approach. In both of these, net decomposition is performed one net at a time. When a net is being decomposed, available information about the decomposition of all other nets is used. One complete iteration of each heuristic involves decomposing or modifying the decomposition of each of the nets. As many iterations of each heuristic as desired can be performed. The heuristic may be terminated either when an iteration does not improve the decomposition (i.e. does not reduce the maximum channel density) or when the available computation resources are exhausted.

HEURISTICS OF AOSHIMA AND KUH

Optimum decomposition

In the optimum decomposition heuristic of Aoshima and Kuh[1], a graph $G_n = (V_n, E_n)$ is constructed for the net n that is to be decomposed. The vertices V_n of G_n include all terminals of net n together with the available feedthroughs. (i, j) is an edge of E_n only if the terminals

Computer Science Department, University of Minnesota, 136 Lind Hall, 207 Church Street S.E., Minneapolis, MN 55455, USA

*Currently at AT&T Bell Laboratories, Murray Hill, NJ, USA

This research was supported, in part, by a grant from Control Data Corporation and by the National Science Foundation under grant MCS-83-05567

and/or feedthroughs represented by i and j are neighbours in the same horizontal channel (an example is given in Figure 2).

Each edge in E_n is assigned a weight. This is obtained by considering the decomposition of all nets so far decomposed. The weight of an edge is the maximum density of the horizontal channel interval spanned by the edge (note that each edge represents an interval in some channel).

In the optimum decomposition heuristic of Aoshima and Kuh[1], the decomposition of net n is obtained by finding a min–max Steiner tree (redundant edges are eliminated). A min–max Steiner tree is a Steiner tree in which the weight of the maximum weight edge is a minimum. As pointed out in Aoshima and Kuh[1], a min–max Steiner tree of G_n may be found quite easily using a modified version of Kruskal's spanning tree algorithm[5].

Practical approach

The optimum decomposition heuristic does not attempt to limit the usage of feedthroughs. Since the excessive use of feedthroughs in the decomposition of the first few nets could result in an inability to decompose the remaining nets (especially for chips with a limited number of feedthroughs), Aoshima and Kuh[1] also propose a heuristic that attempts to minimize the usage of feedthroughs. This heuristic is called the 'practical approach'. In this heuristic, a graph G_d is constructed in a manner similar to the construction of G_n. However, the vertex set of G_d includes only the terminals of net n (no feedthroughs are included).

Each connected component of G_d spans a consecutive set of horizontal channels and represents a subnet of n. The decomposition for each subnet is obtained by finding a min–max spanning tree for the corresponding components of G_d.

Next, the subnets are considered in pairs beginning at the top of the chip. Let n_1 and n_2 represent, respectively, the top subnet and the one immediately below it. A new graph is constructed in the same way that G_n was constructed for the optimum decomposition heuristic. However, all terminals in n_1 are represented by the single vertex S and those in n_2 by the single vertex T. The new graph is directed and edge weights are defined as for G_n. A min–max directed ST path is found in the constructed graph. This represents the interconnections to be used to connect subnets n_1 and n_2. An example is shown in Figure 3.

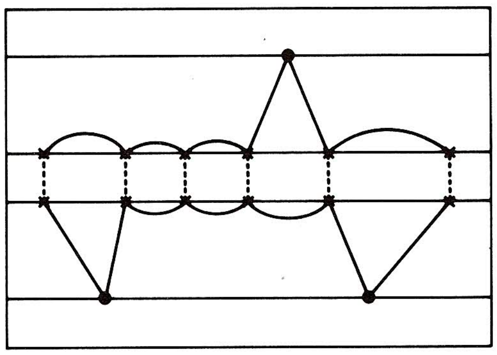

Figure 2. Example of graph of net for optimum decomposition

Overview

The practical approach heuristic was developed by Aoshima and Kuh[1] to overcome the tendency of their optimum decomposition heuristic to use an excessive number of feedthroughs. The excessive use of feedthroughs in the decomposition of a net is considered harmful as there may be no feedthroughs left to decompose remaining nets. However, by restricting the use of feedthroughs too much, as in the practical approach heuristic, the possibility of obtaining very good decompositions (i.e. those with truly low maximum channel density) may be eliminated. For example, consider the two-channel chip of Figure 4(a). For the net (x_1, x_2, x_3) no feedthroughs are permitted by the practical approach heuristic. Thus, the only decomposition possible is that of Figure 4(a). When use of a feedthrough is permitted, the decompositions of Figures 4(b)–4(d) become possible. Any of these could be more desirable than that of Figure 4(a).

In the next section, enhancements to the graph theoretic approaches of Aoshima and Kuh[1] are presented, and in the subsequent section experimental results are

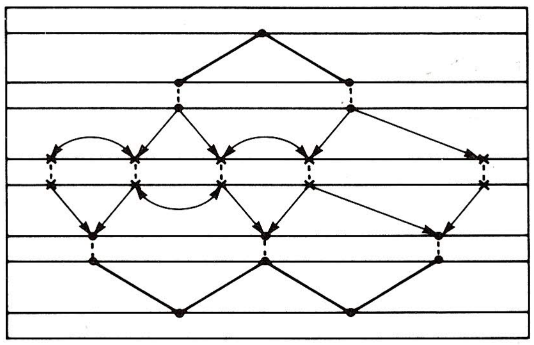

Figure 3. Example of directed graph

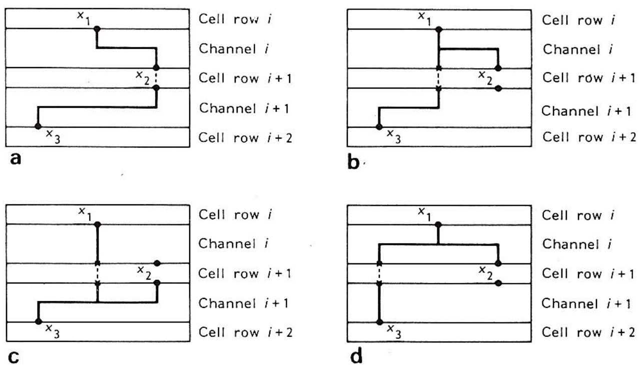

Figure 4. Decompositions of net (x_1, x_2, x_3)

given that indicate that the enhanced heuristic obtains better net decompositions than either of the heuristics proposed by Aoshima and Kuh[1].

ENHANCED HEURISTIC

The enhanced heuristic is an enhancement of the optimum decomposition heuristic of Aoshima and Kuh[1]. It uses two additional concepts: spacing and windowing.

Spacing

When constructing the graph $G_n = (V_n, E_n)$, a spacing factor S, $S \geqslant 1$, is used. Two terminals have a spacing of S only if they are on the same boundary (top or bottom) of a horizontal channel and there are exactly $(S-1)$ terminals between them (see Figure 5). Each channel boundary is scanned left to right. Only feedthrough terminals and terminals of net n are candidates (whether it is a feedthrough or a terminal of net n) for inclusion in V_n. If the next candidate is a terminal of net n it is included in V_m. If it is a feedthrough, it is included if it is spaced at least S from the last terminal included in V_n. The same criterion is used for the remaining candidates. When $S = 1$, all feedthroughs are included in V_n. E_n is defined as in the optimum decomposition heuristic of Aoshima and Kuh[1].

By varying the spacing factor S, the use of feedthroughs may be controlled. When S is small, more feedthroughs are used in a net decomposition and when S is large, less are used. Besides this, when a small spacing factor is used, each edge in E_n represents a smaller interval of a horizontal routing channel. Hence the weight assigned to it is a more accurate reflection of the density in the interval (as density may change in an interval). Consequently the graph G_n more accurately models the cost of different decompositions and it is possible to obtain better solutions. On the other hand, a small S implies a larger graph G_n, and hence requires more computational resources. A large S has exactly the opposite effects.

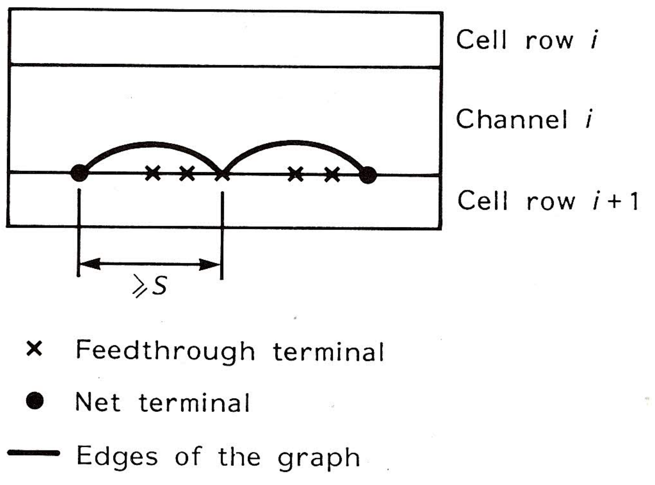

Figure 5. Illustration of spacing concept

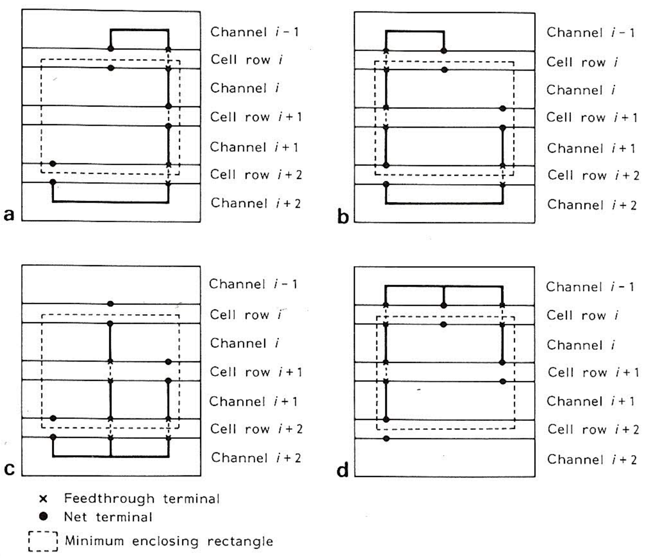

Figure 6. Decompositions using windowing concept

Windowing

The windowing concept is the same as that used for Lee's router[6]. A rectangular window is placed over the chip. The net decomposition is restricted to lie within this window. The window used must include the smallest bounding rectangle for the net. All of the decompositions of Figure 4 are possible when the decomposition window is exactly equal to the minimum bounding rectangle. However, when a larger window is used, the decompositions of Figure 6 become possible (of course, those of Figure 4 are still permissible). The heuristics of Aoshima and Kuh[1] use a window whose width is the chip width and whose height is the same as that of the minimum bounding rectangle.

Some of the important effects of using small or large windows are summarized below.

- Small windows
  Advantages
  - Net length is expected to be shorter
  - Interference in the decomposition of other nets is reduced
  - Decomposition time for the current net is reduced

  Disadvantages
  - Net decomposition may not be possible because of unavailability of feedthroughs in the specified window
  - If a decomposition is obtained, its cost may be higher than necessary
- Large windows
  Advantages
  - Greater possibility to decompose the net
  - Improved chances of obtaining a low cost decomposition

  Disadvantages
  - Net length is expected to be larger
  - Interference among nets is increased
  - More computational resources are needed to run the decomposition algorithm

Enhanced heuristic

The proposed enhanced heuristic consists of three phases. In the first phase, it begins with a 'reasonably' large spacing factor S and a window as wide as the minimum enclosing rectangle, and high enough to include the two horizontal channels that bound the minimum enclosing rectangle. In each iteration of the first phase, the nets are decomposed one at a time. When a net is being decomposed, the current decomposition of the net is obtained by constructing a to determine the edge weights in G_n. The new decomposition of the net is obtained by onstructing a min–max Steiner tree for G_n that includes the terminals of the net. The new decomposition is accepted only if the new min–max channel density is no worse than with the previous decomposition. In case a decomposition is not possible, the window size is increased for this net and the decomposition reattempted. Phase 1 is terminated either when 10 iterations have been performed or when one iteration results in no improvement over the previous iteration.

If the first phase used fewer than 10 iterations, phase 2 is entered. In this phase, S is reduced to a small value ($S = 1$ was used in the experiments described here). The window size for each net is held fixed in this phase and improvement iterations similar to those of phase 1 are performed. Phase 2 terminates when the total number of iterations of phase 1 and 2 become 10 or when no improvement is recorded in an iteration.

In phase 3, the window size is increased and S is held to the small value of phase 2. Improvement iterations are made until no improvement is recorded in an iteration or the total number of iterations of phases 1, 2, and 3 becomes 10. After each iteration the window size is increased (unless it is already the maximum). The improvement iterations of phase 3 attempt to redecompose only those nets that pass through the most congested channel areas.

EXPERIMENTAL RESULTS

The following heuristics were coded in C and run on an Apollo DN320 workstation.

- H1 Optimum decomposition heuristic of Aoshima and Kuh[1]
- H2 Practical approach heuristic of Aoshima and Kuh[1]
- H3 Phases 1 and 3 of the enhanced heuristic described in the last section
- H4 Enhanced three-phase heuristic described in the last section

19 test sets were generated in the following way. First, two practical circuit designs obtained from Control Data Corporation were used. The first of these had 290 macro cells, 684 internal nets and 112 I/O nets. The second circuit had 1850 macro cells, 1700 internal nets and 120 I/O nets. For the first circuit, nine different placements were generated by changing the number of cell rows available, the length of a cell row, and the fraction of a cell row occupied by macros. For the second circuit, ten different placements were generated in the same way. Thus, a total of 19 different tests sets were obtained.

Table 1. Number of cell rows and occupation density in test cases

Test number	Number of rows	Chip density (%)
1	18	93
2	17	91
3	16	90
4	15	89
5	14	90
6	17	74
7	16	78
8	16	78
9	15	84
10	15	84
11	15	95
12	8	97
13	15	52
14	14	56
15	13	60
16	12	65
17	11	71
18	10	78
19	9	87

Table 1 provides the number of cell rows and the occupation density of these rows for the 19 test cases. Table 2 provides the results obtained by each of the four heuristics H1–H4. On test instances 1, 7, 8, 9, and 10, the optimum decomposition heuristic H1 is unable to obtain a feasible decomposition. This is due to feedthroughs for some nets being unavailable. The remaining heuristics were able to obtain a feasible decomposition for all 19 test cases. The points to the virtue of using feedthroughs conservatively. The most conservative approach, i.e. the practical approach heuristic H2, is by far the fastest heuristic. However, the quality of the solutions (as measured by the maximum channel density) produced by this heuristic is quite inferior to that of the solutions produced by the other three heuristics. In fact, for over half the instances, H2 produced decompositions with a maximum channel density that was 80% larger than that of the best solutions obtained.

A comparison of the relative performance of the four heuristics shown in Table 2 leads to the conclusion that heuristic H4, the enhanced three-phase heuristic, performs best. It produced the best decomposition for all test cases except case 16. Its run time, while longer than that of H2, is quite acceptable and is also less than that of H1.

CONCLUSIONS

The multichannel optimization heuristic of Aoshima and Kuh[1] has been enhanced. The enhanced heuristic produces solutions with a smaller channel density and requires a computing time that is between that required by the two heuristics of Aoshima and Kuh[1]. This enhancement has been obtained by introducing the concepts of spacing and windowing.

Table 2. Comparison of performance of the four heuristics

Test	H1			H2			H3			H4		
	Number of iterations	Time (s)	Maximum channel density	Number of iterations	Time (s)	Maximum channel density	Number of iterations	Time (s)	Maximum channel density	Number of iterations	Time (s)	Maximum channel density
1	10	5754	–	5	1568	24	7	1971	21	7	1874	20
2	10	5878	28	6	1994	25	6	1626	23	10	3159	20
3	10	6100	28	3	951	29	7	1984	23	7	2468	22
4	10	5466	23	7	2250	29	7	1939	23	8	2700	22
5	10	5096	22	5	1397	27	7	2082	23	9	2493	21
6	10	12503	44	3	1175	75	9	5230	38	10	4884	37
7	10	14622	–	3	1109	72	7	4112	43	10	5407	40
8	10	15636	–	4	1523	70	8	4773	42	10	5482	41
9	10	14460	–	3	1022	87	6	3538	53	10	5555	49
10	10	19461	–	4	1398	89	10	6329	48	10	5245	48
11	10	1651	14	6	374	16	10	617	13	10	868	13
12	5	1036	17	4	183	18	10	636	14	10	937	14
13	6	2448	11	2	211	25	8	784	10	9	2202	10
14	5	1469	13	3	338	25	5	494	14	10	2623	13
15	7	1658	11	2	206	29	10	1479	11	9	2228	11
16	8	2317	12	2	177	28	7	719	13	7	1226	13
17	5	2055	13	2	157	28	6	576	13	10	1820	13
18	7	2650	15	2	148	26	6	530	16	9	1463	15
19	10	2950	17	2	145	33	4	321	17	8	1043	17

REFERENCES

1 **Aoshima, K and Kuh, E S** 'Multi-channel optimization in gate-array LSI layout' *Proc. ICCAD '83* (1983) pp 1005–1008

2 **Kambe, T, Chiba, T, Kimura, S, Inufushi, T, Okuda, N and Nishioka, I** 'A placement algorithm for polycell LSI and its evaluation' *19th IEEE Des. Automation Conf.* (1982) pp 655–662

3 **Deutsh, D N** 'A dogleg channel router' *Proc. 13th Des. Automation Conf.* (1976) pp 425–433

4 **Rivest, R L and Fiduccia, C M** 'A "greedy" channel router' *Proc. 19th Des. Automation Conf.* (1982) pp 418–424

5 **Horowitz, E and Sahni, S** *Fundamentals of computer algorithms* Computer Science Press, Potomac, MD, USA (1978)

6 **Aker, S** 'Routing' in **Breuer, M (ed.)** *Design automation of digital system: theory and techniques* Vol 1 Prentice-Hall, NJ, USA (1972)