Adaptive Cut Line Selection in Min-cut Placement for Large Scale Sea-of-gates Arrays

K. Takahashi[†] K. Nakajima[‡] M. Terai[†] K. Sato[†]

[†]System LSI Laboratory
Mitsubishi Electric Corp.
Itami, Hyogo 664

[‡]Electrical Department
Meryland Univ.
College Park, Meryland 20742

Abstract

We present a new min-cut based placement algorithm for large scale sea-of-gates arrays. In the past all such algorithms used a fixed cut line sequence that is determined before min-cut partitioning is performed. In our approach, we adaptively select a next partitioning pattern based on the current parameter value; we then perform the corresponding min-cut partitionings and measure a new parameter value. We repeat this process until all cut lines are processed. As a parameter, we introduce a new global objective function based on wire congestions on cut lines. We establish a close relation between this function and cut line sequences. This relation is used to develop an innovative method of adaptively determining a cut line sequence so as to minimize this global function. With this adaptive selection of cut lines along with a new cluster-based min-cut partitioning technique, our algorithm can produce, in a short time and at a low cost, final placement results that achieve the 100% completion of wiring on chips of fixed sizes. This has led to its successful production use, having generated more than 400 CMOS sea-of-gates array chips.

1 Introduction

In the *gate array* approach final products are made at a low cost by customizing only metal layers of preprocessed gate array chips, called *masterslices.* Min-cut based placement has most often been used in existing layout systems for gate arrays (*e.g.* [2, 4, 7]). This is because (1) its primary goal is the reduction of excessive congestions of wires [7] and (2) it produces in a short time a placement that achieves the 100% completion of wiring [6].

As the sizes of gate arrays get larger, the wiring completion problem becomes more difficult to handle in the traditional framework of min-cut placement. This is mainly due to (i) its reliance on the single *local* optimization function, called the *sequential cut line* objective function [1] and (ii) the use of a fixed cut line sequence that is determined at the beginning of min-cut placement. In this paper we introduce a global function based on the congestion of wires on each cut line. We then propose a method of adaptively determining next patterns of cut lines so as to minimize this function.

In our masterslice model illustrated in Fig. 1, the entire chip area consists of an internal area containing rows of *internal slots*, and a peripheral area containing *external slots.* A circuit is hierarchically specified with sets of *circuit elements* and *signal nets* that connect circuit elements. It is composed of *functional modules* such as ALUs, comparators, and counters, and each module in turn consists of lower level functional modules such as flip-flops and logic gates. The circuit elements are categorized as *external elements* and *internal elements* that are to be allocated to external and internal slots, respectively. Since the positions of all external elements are usually predetermined, in this paper we deal only with the internal elements and slots, which we refer to simply as *elements* and *slots*, respectively.

We define the *size of an area* as the number of slots that reside in it. We define the *size of an element* as the number of slots required to accommodate it. We call a set of one or more elements a *cluster* and define its *size* as the sum of sizes of the elements that constitute the cluster. Similarly, we define the *size of a functional module.* We define our *placement problem* as that of allocating all elements of a given circuit to slots of a masterslice in such a way that no two elements share the same slot.

A grid of horizontal and vertical lines is defined on a masterslice and all wiring for signal nets between elements is done on grid lines. In the routing areas that exist between adjacent rows of slots, both horizontal and vertical grid lines can accommodate wire segments as long as they do not overlap. In the rows of slots, only those vertical grid lines that are not obstructed by the wiring patterns of elements can be used as *feed-throughs* for wiring.

In *min-cut placement* [1], we define *cut lines* on a masterslice in both the horizontal and the vertical directions. In our masterslice model, we set a *horizontal cut line* between each pair of adjacent rows of slots and a *vertical cut line* on the boundary between each pair of adjacent slots. At each step of min-cut placement, we first select a cut line and divide each area of concern into two subareas of as equal sizes as possible along the line. We then reassign one subset of the elements to one subarea and the other subset to the remaining subarea. In this new assignment, if the elements connected to the same signal net are allocated to different subareas, the net is said to be *cut* by the cut line. The number of signal nets to be cut by a cut line is called a *cut value* of the line. We perform the above area division and element assignment in such a way that (1) the corresponding cut value is minimized and (2) the sum of sizes of the elements to be assigned to each new subarea does not exceed the size of the subarea.

We call such a division and assignment a *min-cut partitioning*, or simply *partitioning.* We repeatedly apply this partitioning to each of the subareas until all cut lines are processed. We call a partitioning along a horizontal and vertical cut line simply a *horizontal* and *vertical partitioning*, respectively. The sequence of selected cut lines along which partitionings are performed is called a *cut line sequence.*

In this paper we introduce a new global objective function, called the *max-congestion.* We define the *cut line congestion* as the ratio of its corresponding cut value and the number of grid lines usable for wiring that intersect it. In our definition the cut value of each cut line is the number of signal nets rather than wires that are cut by the cut line. Thus, the cut value gives a lower bound on the number of actual wire segments that cross over the cut line. Therefore, as the cut line congestion approaches 1.0, detoured wires and/or unroutable wires may start to appear. We call the largest values among the congestions on all horizontal and all vertical cut lines the *maximum horizontal* and *vertical cut line congestions*, respectively. We call the largest of the two congestions the *maximum cut line congestion.* Under our new objective function of max-congestion, we minimize the maximum cut line congestion.

We first empirically demonstrate the existence of a close relation between the max-congestion function and cut line sequences. Using this relation, we develop an innovative method of adaptive selection of cut lines for the minimization of the max-congestion objective function. Unlike previous approaches [1, 7], we do not select at the beginning of min-cut placement, an entire cut line sequence among those sequences that are predetermined. Instead, we adaptively select next patterns of cut lines to be processed and eventually generate the entire cut line sequence at the end of this process. We use as a guide the relative ratio of wiring capacities in the horizontal and vertical directions that are *a priori* determined for each combination of a masterslice and a circuit.

In Section 2, we show our experimental results and establish a relation between cut line sequences and the max-congestion objective function. We then develop a method of adaptive selection of cut lines to minimize this function. In Section 3, we first briefly explain our cluster-based min-cut partitioning approach and some other implementation details. We then present the entire placement algorithm. Section 4 provides its evaluations and further discussions.

2 Cut Line Sequence and Congestions

In this section, we examine extensively the effect of cut line sequences on placement results for large scale sea-of-gates arrays. We establish a relation between cut line sequences and the maximum congestions on its corresponding horizontal and vertical cut lines. We use this relation to develop a method of adaptive selection of cut lines to minimize the new global objective function of *max-congestion*.

Before we proceed, we define the important notion of *levels* for (i)the areas that are generated by partitionings, (ii)their corresponding cut lines, and (iii)the partitionings done along those cut lines. We assign level 1 to the initial entire area on a masterslice. For $i \geq 1$, we give level $i+1$ to those new areas that are produced by dividing the areas of level i. We assign level i to a cut line that divides an area of level i. We also give level i to a partitioning performed along a cut line of level i.

In this paper we assume that (1)cut lines of the same level are in the same direction. We also assume that (2)we perform all partitionings of level $i+1$ only after we complete all partitionings of level i. These propositions imply the following:The process of partitioning in any ordering of cut lines of the same level would yield the same placement result. This leads us to redefine a cut line sequence to be a permutation of the directions in which the cut lines of each level are processed.

1. Relation between Cut Line Sequences and Maximum Congestions

Let h_M and v_M denote the largest values among the cut values of all horizontal and all vertical cut lines, respectively, in a cut line sequence. Let t_h and t_v be the numbers of usable grid lines that intersect such horizontal and vertical cut lines, respectively. The maximum horizontal and vertical cut line congestions are expressed as h_M/t_h and v_M/t_v, respectively. In our masterslice model, t_v represents the total number of horizontal grid lines in the routing areas between adjacent rows of slots. Although it may be defined in several ways, as an approximation we set t_h to be the average number of feed-throughs available in a row of slots. Note that the values of t_h and t_v are known before the execution of our placement program, and they are constants.

In order to observe how cut line sequences affect the values of h_M/v_M and $h_M/t_h + v_M/t_v$, we performed our min-cut partitionings on nine typical circuits to be realized on masterslices of 18.6K to 150K raw gates. Main features of the circuits are given in Table 1. The results are plotted in Figs. 2 and 3. In these experiments, we performed cluster-based partitioning at the first six levels and element-based partitioning at the remaining levels, as explained in Section 3. We used 64 cut line sequences enumerated in Table 2. The symbols *H* and *V* denote the horizontal and vertical directions, respectively, of cut lines of the same level. Each sequence contains six *H*s and six *V*s. In theory there are $924(={}_{12}\mathrm{C}_6)$ sequences. Due to the high cost of experiments, we have selected those 64 sequences that realize all possible combinations of *H*s and *V*s at the first six levels. This is because as demonstrated later, the partitionings at earlier levels are more likely to affect the values of h_M and v_M than those of later levels. The partitionings of levels 7 to 12 are performed in alternate directions as long as there remain both *H*s and *V*s.

Note that the numbering of sequences in Table 2 is done by decoding their first six characters as a 6-bit binary number with an *H* and a *V* representing 0 and 1, respectively. The larger (respectively, smaller) a sequence number is, the more significant the bit positions of *V*s (respectively, *H*s) are in the sequence. We say that the vertical direction is *more significant* in one partitioning pattern *P* than in another one *Q* if a *V* appears in *P* in the leftmost bit position in which *P* and *Q* contain different symbols. *P* is also said to be more *vertical-oriented.* Similarly, we say that the horizontal direction is more significant in *Q* than in *P* and that *Q* is more *horizontal-oriented.* If exactly two patterns *P* and *Q* are under consideration, *P* and *Q* are simply said to be *vertical-oriented* and *horizontal-oriented* patterns, respectively.

As shown in Fig. 2, as the number of the sequence applied increases, the value of h_M/v_M tends to increase on the whole with a few exceptions mentioned below. Thus the value of h_M/v_M clearly depends on the significance of the direction of partitionings. However, when the number increases from $4r-1$ to $4r$ for each $r = 1, 2, \cdots, 15$, the value of h_M/v_M decreases drastically. This exceptional phenomenon is caused by the reversal of partitioning patterns such as from *HVV* to *VHH* (i.e., in their binary representation, from 011 to 100).

As is seen from Fig. 3, the values of $h_M/t_h + v_M/t_v$ are rather large when such cut line sequences as Nos. 0-9, 15-17, 23, 24, 30-33, 39, 40, 46-48, 54-63 are applied. The main feature of these sequences is that partitionings of the same direction take place successively at more than two levels. For all the other sequences, the values may be considered to be a constant. In such a case, the minimization of the maximum values of h_M/t_h and v_M/t_v becomes equivalent to the equalization of h_M/t_h and v_M/t_v. Therefore, the selection of a cut line sequence that satisfies $h_M/v_M \simeq t_h/t_v$ would result in the minimization of the max-congestion objective function. This would naturally lead to the minimization of the maximum congestion of signal wires and hence reduce the possibility of generating detoured wires and/or unroutable wires.

2. Selection of Partitioning Patterns

In what follows, we discuss a method of equalizing h_M/v_M and t_h/t_v when the value of $h_M/t_h + v_M/t_v$ is kept a constant.

Let h_M^q and v_M^q denote the maximum values among the cut values of all the horizontal and all the vertical cut lines processed thus far, respectively, of levels less than or equal to q. We consider the application of a horizontal-oriented pattern P_h, and a vertical-oriented pattern P_v, of partitionings of levels $q+1$ to $q+x$ that include both *H*s and *V*s. Note that partitionings of both directions are needed to determine new values for h_M and v_M. This is because if partitionings of a single direction only, say H, are performed, the value for h_M always increases while that for v_M is unchanged, and hence the ratio of h_M/v_M monotonely increases although we want it to decrease as explained later. Judging from Fig. 2, it is expected that h_M^{q+x}/v_M^{q+x} becomes smaller(respectively, larger) in value than h_M^q/v_M^q in the case of P_h(respectively, P_v).

In order to verify the above conjecture, we compared the values of h_M^q/v_M^q and h_M^{q+x}/v_M^{q+x}, when cut line sequences Nos. 17, 18, 21, and 22 were applied to circuit C_6. The values of x and q each were set to 3. The results are shown in Fig. 4. Note that all sequences have the same initial partitioning pattern *HVH* and thus their h_M^3/v_M^3 values are all 0.59. The patterns of the next three levels for the sequences Nos. 17, 18, 21, and 22 are *HHV*, *HVH*, *VHV*, and *VVH*, respectively, and their h_M^6/v_M^6 values are 0.42, 0.51, 0.62, and 0.72, respectively. Clearly, the more significant horizontal (respectively, vertical) partitionings are, the more the ratio of h_M/v_M decreases (respectively, increases).

Using this property of the ratio, we adaptively select partitioning patterns of length x. In other words, at the completion of partitionings of level $i \cdot x$, where i is a positive integer, we measure the value of h_M^{ix}/v_M^{ix}. If $h_M^{ix}/v_M^{ix} \geq t_h/t_v$, we select the horizontal-oriented pattern; otherwise, we select the vertical-oriented pattern. The aim of this process is to make the value of $h_M^{(i+1)x}/v_M^{(i+1)x}$ closer to that of t_h/t_v than the value of h_M^{ix}/v_M^{ix}.

Figure 4 also shows that $\mid h_M^{(i+1)x}/v_M^{(i+1)x} - h_M^{ix}/v_M^{ix} \mid$ decreases as i increases in value. Thus, with the above method, the value of $h_M^{(i+1)x}/v_M^{(i+1)x}$ becomes closer to that of t_h/t_v than the value of h_M^{ix}/v_M^{ix}. We may conclude that the above process is able to make the final value of h_M/v_M closer to that of t_h/t_v.

In such a method, the smaller the value of x is, the finer tuning of the ratio we can make. When $x = 2$, only two partitioning patterns of *HV* and *VH* are possible. In this case, the most horizontal- and vertical-oriented patterns are *HVHV...HV* as in sequence No. 21 and *VHVH...VH* as in No. 42, respectively, of Table 2. Accordingly, the values of h_M/v_M vary in the range of 0.8 to 1.2 as shown in Fig. 2. On the other hand, the practical values of t_h/t_v vary from 0.5 to 1.3. This prohibits the selection of these two patterns.

When $x = 3$, the possible patterns that include both *H*s and *V*s are *HHV*, *HVH*, *HVV*, *VHH*, *VHV*, and *VVH*. However, the inclusion of patterns *HHV*, *HVV*, *VHH*, and/or *VVH* would generate cut line sequences in which partitioning of the same direction takes place successively at three or more levels. As mentioned before, such sequences may not keep the value of $h_M/t_h + v_M/t_v$ a constant, which would destroy the basis of our discussion. Therefore, we select the two remaining partitioning patterns, *HVH* and *VHV*; the former is a horizontal-oriented pattern and the latter a vertical-oriented pattern. In this case the values of h_M/v_M vary from 0.5 to 1.4, which covers the range of practical values of t_h/t_v. In conclusion, we decide to use the two patterns *HVH* and *VHV* in an attempt to equalize the values of h_M/v_M and t_h/t_v.

3 The Placement Algorithm and Implementation

Before we present the overall placement algorithm, we explain our cluster-based min-cut partitioning approach and clarify two other implementation issues.

1. Clustering

In the past it was reported that cluster-based min-cut partitioning approaches reduce the total processing time and the cut value [4, 5, 6]. In any such approach, the quality of clusters plays the most important role in the production of a final solution. The main features of our approach are: (i) logical hierarchy of a circuit is used in cluster generation, and (ii) total cut value is used to determine an upper bound on the sizes of clusters to be generated.

Our min-cut partitioning approach consists of three steps: (1) cluster generation, (2) partitioning based on clusters, and (3) partitioning based on elements. We use the algorithm of Shiraishi and Hirose [7] to perform min-cut partitionings on clusters at Step (2) and on elements at Step (3). We generate clusters at Step (1) as follows. The average size of the areas to be created at the end of partitionings of level j is calculated as $s(j) = \lceil A/(2^j) \rceil$, where A is the total number of slots defined on a masterslice. We use the value of $s(j)$ as an upper bound on the sizes of clusters to participate in all partitionings at Step (2). Among all functional modules that appear in the hierarchy of a circuit, we select as clusters as large modules as possible that do not exceed $s(j)$ in size. If the size of a single element exceeds $s(j)$, it makes a cluster by itself. After extensive experiments, we have decided to set the value for j to be six.

2. Other Issues

We describe two additional implementation details. The first issue is on the determination of an initial value of h_M/v_M. We may simply perform partitionings with two patterns *HV* and *VH* separately and select the pattern that yields a value of h_M^2/v_M^2 which is closer to that of t_h/t_v than the other. However, to be consistent with the discussion of the previous section and for simplicity of programming, we decided to apply the two chosen patters *HVH* and *VHV* in our implementation.

The second implementation issue is concerned with the situation in which all partitionings in one direction have been exhausted and partitionings with a particular pattern cannot be performed. In such a situation we ignore the pattern and perform partitionings in the remaining direction.

3. The Algorithm

We now present the overall placement algorithm. Let l^* show the level of the last partitioning.

Placement Algorithm

```
Step 1. (Clustering):
    Generate clusters by the method described above.
Step 2. (Partitionings of the first three levels):
    2.1. Perform partitionings with HVH and VHV.
    2.2. Select the partitioning result with which the resultant
         value of $h_M^3/v_M^3$ is closer to the value of $t_h/t_v$.
Step 3. (Initialization of the level $l$ of partitionings):
    Set $l \leftarrow 3$.
Step 4. (Selection of a pattern of partitionings of the next three levels):
    If $h_M^l/v_M^l \geq t_h/t_v$ , select the pattern of HVH; otherwise,
    select VHV.
Step 5. (Partitionings of levels higher than 3):
    5.1. Perform the following sequence of operations three times
         as long as $l \leq l^*$.
         (1) Set $l \leftarrow l + 1$.
         (2) If $l > l^*$, go to Step 6.
         (3) Perform partitionings of level $l$ on clusters if $l \leq j$ or
             on elements if $l > j$. If no cut line of a particular direction
             remains, perform partitionings of the other direction.
    5.2. Go to Step 4.
Step 6. (Elimination of overlappings of elements):
    If overlappings of elements are created, remove them and
    stop. □
```

4 Evaluations and Discussions

We coded our placement algorithm in FORTRAN and implemented it on our gate array layout system. The system is set up on an IBM mainframe computer Model 3090.

Using the circuits C_2, C_5, and C_6 of Table 1, we compared our new placement program and the internally developed program [3] that was previously in production use. The results are tabulated in Table 3.

From Table 3, the superiority of our new program over the old program is more than evident. Ours ran 11.3, 32.2, and 33.5 times faster than the old program for circuits C_2, C_5, and C_6, respectively. Furthermore, the old program failed to generate a placement that results in the 100% completion of wiring for circuit C_6. Finally, our results on total wire length were also always better than those of the old program.

Using circuit C_5, we also compared our layout system with the new placement program and two CAD vendors' gate array layout systems. The results are provided in Table 4. Since the vendors' systems run on workstations, their processing times are converted to their equivalent values on the IBM 3090 computer, based on the MIPS values of their respective computers. Our system produced a placement whose total wire length is shorter by 8% and 37% than that of the placements obtained by the first and second vendor's program, respectively. As for the total processing time for placement, we could not measure the time for each of the vendors' systems. Instead, we give the sum of total processing times for their placement and routing. Under this combined measure, our system ran 7.9 and 6.3 times faster than the first and second vendor's system, respectively.

5 Conclusions

As demonstrated above, our new placement program outperforms the two CAD vendors' programs as well as our old. Furthermore, it has successfully produced into the real world more than 400 CMOS sea-of-gates array chips of 1.5K to 150K raw gates. It should also be noted that our program requires no manual specification of parameters associated with min-cut partitioning. This enables any user who is not skilled at gate array layout and/or min-cut placement, to easily obtain a good result on the first run of the placement program.

References

[1] M. A. Breuer, "Min-cut placement," *Jour. Design and Fault-Tolerant Computing*, vol. 1, no. 4, pp. 343-362, Aug. 1977.

[2] M. Igusa, M. Beardsiee, and A. Sangiovanni-Vincentelli, "ORCA: A sea-of-gates place and route system," *Proc. 26th DAC*, June 1989, pp. 122-127.

[3] S. Murai, H. Tsuji, M. Kakinuma, K. Sakaguchi, and C. Tanaka, "A hierarchical placement procedure with a simple blocking scheme," *Proc. 16th DAC*, June 1979, pp. 18-23.

[4] C. Ng, S. Ashtaputre, E. Chambers, K. Do, S. Hui, R. Mody, and D. Wong, "A hierarchical floor-planning, placement, and routing tool for sea-of-gates design," *Proc. CICC*, May 1989, paper no. 3.3.

[5] T. -K. Ng, J. Oldfield, and V. Pitchumani, "Improvements of a mincut partition algorithm," *Proc. IEEE ICCAD*, Nov. 1987, pp. 470-473.

[6] T. Payne, R. Wells, and W. Gundel, "A study of automatic placement strategies for very large gate array designs," *Proc. IEEE ICCAD*, Nov. 1987, pp. 194-197.

[7] H. Shiraishi and F. Hirose, "Efficient placement and routing for master-slice LSI," *Proc. 17th DAC*, June 1980, pp. 458-464.