Consistent Floorplanning with Super Hierarchical Constraints

Shigetoshi NAKATAKE, Yukiko KUBO and Yoji KAJITANI
The University of Kitakyushu
1-1 Hibikino, Wakamatsu-ku, the City of Kitakyushu
Fukuoka, 808-0135 Japan
{nakatake, kubo, kajitani}@env.kitakyu-u.ac.jp

ABSTRACT

Sequence-Pair based floorplanning has been revealed its limit of usefulness in VLSI physical design. Among reasons, the key issue is in its being non-hierarchical and indifferent to the preceding step of partitioning. This paper restructure the algorithm by the idea that the partition algorithm produces the constraint that is imposed on the sequence-pair data. The partition algorithm is the one based on the balanced-bipartition that works hierarchically. Thus a new floorplan algorithm that is consistent with the hierarchical partition algorithm is constructed. This is enhanced to include other algorithms that are based on the binary search. Here, a clock-tree synthesis by H-tree is shown to be consistent. Experiments are given to show better achievements in length and wire-density.

1. INTRODUCTION

Physical design of VLSI chips is becoming harder to converge timing issues and spatial embedding as the target VLSIs are becoming larger and more complex. Thus, physical design plan itself is now requested to be restructured.

The key steps of physical design related with are circuit-partition, block-placement, clock-tree-synthesis, and routing. Those steps are all controlled by floorplanning that defines relative positions on the chip to accept circuit elements. If we discuss the circuit design in block-level, a floorplan is the 2-dimensional arrangement of rooms, spaces, and line segments [1, 2, 4, 6, 7, 9]. The output of algorithms of these four steps, whatever they are, shall be explained consistently on the floorplan. Thus, in any physical design is understood to go in a fashion that if the result is not satisfiable, change the floorplan or the algorithm that is considered the bottleneck.

Floorplanning is to dissect the chip into rooms. Since blocks are the main contents assigned to the rooms in the following steps, block-placement or block-packing can be used in the same context as floorplanning is if some space is assumed on the peripherals of blocks. Efforts have long been devoted to improve algorithms and floorplanning. As for floorplanning (block-placement or block-packing), the invent of data structures of BSG [7], Sequence-Pair (SP) [6], and O-Tree (OT) [9] was a good leap. But few results have been reported of significant success in practical use compared with the conventional floorplanning that is based on the Slice-Line (SL) data structure [2].

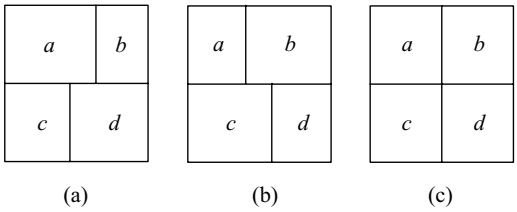

Figure 1: Possible placements by cluster size

Good in respect to the generality, those BSG, SP or OT based floorplan algorithms (BSGa, SPa or OTa, respectively) is lack of consistency with the preceding step, partitioning. For the circuit partition, though it seems so much variety of methods has been proposed, the basic idea in the back is the balanced-bipartition (BBP).

Those BSGa, SPa and OTa say that they do not mind because they list any possible floorplan for search. However, BBPa minds since it produces the set of clusters unaware of the clusters being embedded on the plane. This claim is natural since the preceding circuit design is connectivity oriented by nature, and so BBPa. Thus, we might say that BSGa, SPa and OTa are not consistent with BBPa, neither so SLa. Or more, any block placement algorithm so far proposed is not consistent with BBPa. The same observation, or worse, holds if we adopt the multi-way partition since its instant independent application would produce more detailed and less convincing informations.

Let us study what the above comments mean using a general example. Let a graph $G(V, E)$ denote a circuit where vertices and edges represent the blocks and nets, respectively, in a very conventional manner. Suppose that BBPa gets clustering $V = V_1 \cup V_2$. Any placement algorithm would not have no other idea than to place them above and below. BBPa successively partitions as $V_1 = V_{11} \cup V_{12}$ and $V_2 = V_{21} \cup V_{22}$. The obtained clusters will be placed left and right in each region.

Now they are going to be realized according to their optimality. Let them be denoted simply by labels a, b, c, and d, respectively, as depicted in Figure 1.

ISPD'01, April 1-4, 2001, Sonoma, California, USA.

So far we did not think of the block-sizes. According to them, the vertical line between a and b is either left-of, right-of, or on the same position of the line between c and d, each case corresponds to (a), (b), and (c), respectively, of the figure. This difference will be significant later as hierarchical BBPa proceeds. But there have been no idea to keep this ambiguity, maybe by the reason that people do not like to mix topological and physical features.

As long as BBPa goes hierarchically, there is no stage for the non-slicing structure, e.g. spiral or Order-5 to play a role. It seems that SLa is enough. But still, since BBPa is not perfect (examples where only 4-way partition gets an optimal placement in a spiral are easily constructed), restriction to SL would cause serious degradation.

Only the possible solution is: Given a circuit, start with partition by BBPa and general placement by SPa. Represent the ambiguity or remaining freedom somehow. Letting it as a constraint on the placement by SPa, look for the optimal one.

The main contribution of this paper is in coding of such mixed feature of topology and dimension in SP, by the name of "super-constraint on SP", such that any feasible SP bears the implication which the applied BBPa intended to give. It is an easy form for computation to use in a heuristic search by simulated annealing. Thus we get a floorplanning, which we call the Sequence-Pair with super-constraint, or SPa-super.

Further considerations will reveal that any algorithm as long as it follows the fashion of binary search can be embedded in this floorplanning consistently. Here we only refer to clock-tree synthesis by H-tree.

Finally we would like to mention about our contribution in the context of two historical discussions.

One is about the merit or demerit of treating physical dimensions of blocks and topological relations between blocks in separate form. BSGa and SPa consider topology first and take the dimensions into consideration in the last compaction stage. In this sense, they are called the topological packing. While, OTa is the one that uses physical dimension from the beginning. SLa claims that any time the dimension can be considered. This merit is by a trade-off to the structural generality. A new type floorplanning by Q-Sequence (QS) [10] or Corner-Block-List [11] is classified to the category of topological packing. However, what is to be stressed is that only the dimensions that affects the topology in the following stages are significant. To consider all (OTa) or to consider after (BSGa, SPa, QSa), or to restrict the structure (SLa) are not on the way to the solution. The Sequence-Pair with super-constraint proposed here is the answer.

The other is a blame: Partition algorithm BBPa works hierarchically while those BSGa, SPa, OTa and QSa all execute flat computation. So by definition, there will be no nice matching with BBPa. Another key contribution of this paper is to answer this. Any hierarchical condition induced by BBPa is nicely translated as a constraint on SP.

Because of the feature that the mixed and hierarchical data is coded in a single form, our proposing floorplan algorithm is named "Sequence-Pair Based Floorplan Algorithm with Super-constraint".

A prototype was implemented very faithfully without speeding up ideas, and experimented on a bench-mark. It showed expected performances.

2. PRELIMINARIES

Given a placement of n blocks, gridding is a procedure to determine a pair of sequences (α, β), called the Sequence-Pair (SP). It is defined such that blocks can be removed one after another along α (β) in the above-left (below-left) direction without touching the remaining blocks. (The original definition[6] is more complicated but this simplified definition does not loose generality.) The resultant is not unique in general. For the placements shown in Figure 1 (a), (b) and (c), they are

$$\begin{array}{lll} (a) & : & \alpha = (abcd) \textit{ or } (acbd),\ \beta = (cdab), \\ (b) & : & \alpha = (abcd),\ \beta = (cdab) \textit{ or } (cadb), \\ (c) & : & \alpha = (abcd) \textit{ or } (acbd),\ \beta = (cdab) \textit{ or } (cadb). \end{array} \quad (1)$$

Given an SP, it uniquely represents the relative relation (topology) of {*right-of, left-of, above, below* } for each pair according to the following reasoning. If a block x is taken before block y in the above-left direction, and also before in below-left direction, a reasonable conclusion is that x must be right-of y. Analogously, if x is taken before y in above-left direction and after y in below-left direction, it is concluded that x is above y. This observation is formalized as the decoding rule from the SP to the topology. It is well-known that a topology defines a unique placement after x- and y-direction 1-D compactions. Thus, with 1-D compaction, the sequence-pair is a data structure to store the placements.

The point of this paper is in the discovery that the difference of the placements as shown in Figure 1 can be mapped to the difference of the corresponding sequence-pairs as observed in Equation (1).

A logic is introduced to represent the ambiguity of sequences. The fact that α is possible to be $(abcd)$ *or* $(acbd)$ is represented by formula $a(b+c)d$. Here addition "$x+y$" denotes a commutative relation that order (xy) and (yx) are both feasible, while product "xy" non-commutative relation, that order (xy) is unique. Remember that this is the order constraint not adjacency constraint. Another example to show this is: formula $a(b+cd)$ means that all the possible sequences are $(abcd)$, $(acbd)$ and $(acdb)$.

With this notation, Equation (1) is mapped to the set of formula

$$\begin{array}{lll} (a) & : & \alpha = a(b+c)d,\ \beta = (cdab), \\ (b) & : & \alpha = (abcd),\ \beta = c(d+a)b, \\ (c) & : & \alpha = a(b+c)d,\ \beta = c(d+a)b. \end{array} \quad (2)$$

The binary relation of *implication* is defined on these formula. A formula f is defined to imply formula g if the former contains the latter as a case. For example, $a(b+c)d$ implies $abcd$. For the set of formula, formula h that implies every formula and there is no formula in the set that implies h is called the least upper bound (*lub*) of the set. It is not required that an *lub* is in the set.

Now let us come back to our subject to get a convenient form to represent all the possible sequence-pairs. Find an *lub* to cover all α's. In our case, it is $a(b+c)d$. For β, it is $c(a+d)b$. Then the sequence-pair $(c(a+d)b, c(a+d)b)$ is considered to represent all the possible sequence-pairs, and called SP with a super-constraint.

All the sequence pairs implied in this are feasible sequence-pairs that are worth to be considered because they are fit to the BBP.

The proposing placement system is a heuristic search that

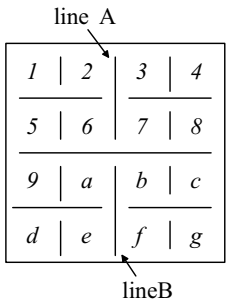

Figure 2: Matrix of placement with 16 clusters

generates feasible sequence-pairs and evaluate each, and choose the best by the strategy of simulated-annealing. In this example, all the feasible seq-pairs are obtained by expanding $(c(a+d)b, c(a+d)b)$: $(abcd, cadb)$, $(acbd, cadb)$, $(abcd, cdab)$, $(acbd, cdab)$. Each corresponds to a floorplan.

We started to list all the meaningful SP's from the point of BBP. But it is a constraint from the point of free SP, which is used in conventional SPa. In fact, SPa will generate all the combinations, so of $(n!)^2 = 576$ SP's $(n = 4)$.

This simplification comes from the feature of the partition algorithm. To consider all these 576 cases is the consequence of the old fashion that each step shall be optimized believing that the input is given (by god) and perfect. But we know the truth. It is simply an output of that may-not-be smart but diligent BBPa.

To make clear our proposing system to generate only the sequence-pairs consistent with BBPa, following a more enhanced example will suffice.

The placement shown in Figure 2 is the result of two-times application of pairs (horizontal partition, vertical partition) in BBPa. This matrix form is called the formal placement since the elements are block-labels without physical dimension and partition is regular. For a real instance, some of partition may be skipped and some of elements are empty. But the following explanation covers these cases.

In the following, a sub-matrix consisting of elements $x, y, ...$ is referred to as [xy...]-part. As for α, the super-constraint imposed on the [1256]-part is $1(2+5)6$. From [bcfg]-part, it is $b(c+f)g$. From [3478]- and [9ade]-parts, they are $3(4+7)6$, $9(a+d)e$, respectively. Furthermore, [1256]-part comes before [bcfg]-part. Thus, the super-constraint is

$$\begin{array}{rcl} \alpha & = & 1(2+5)6(9(a+d)e+3(4+7)8)b(c+f)g \\ \beta & = & d(9+e)a(5(1+6)2+f(b+g)c)7(3+8)4 \end{array} \quad (3)$$

This is the underlying constraint. Somehow on the way of placement, suppose we had an information from the library, for example, that the line A is left-of line B, we get additional constraint added to β as [1256]-part comes before [bcfg]-part. So β in Equation (1) is strengthened to

$$\begin{array}{rcl} \alpha & = & 1(2+5)6(9(a+d)e+3(4+7)8)b(c+f)g \\ \beta & = & d(9+e)a(5(1+6)2)(f(b+g)c)7(3+8)4 \end{array} \quad (4)$$

The packing algorithm SPa with super-constraint is strongly aware of balanced bipartition algorithm BBPa, by which the solution space is drastically reduced. Hence, if we apply this algorithm for minimizing the area by assigning physical dimensions to these 16 blocks, we may have some inferior result with respect to area. In fact, we will have to search

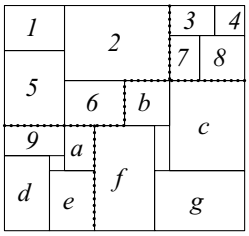

Figure 3: Placement satisfying Equation (2) whose SP is $(15269dae3478bfcg, de9a5f16bg2c7834)$

only the placements that contain block a at the above-left corner. Still we tried an example that searches the placements satisfying Equation (2). The result is shown in Figure 3 which somehow by God is not bad.

3. FLOORPLANNING BY SP WITH SUPER-CONSTRAINT

The basic idea of the consistent physical design was described with stress on the consistency between partition and packing. It was also mentioned that clock-tree synthesis by H-tree is merged consistently. However, before getting into that subject, a floorplanning with consistent partitioning algorithm will be fixed since steps that follow are affected how the preceding step behaves.

3.1 Balanced-Bipartitioning

The algorithm in [1] is adopted for balanced-bipartition algorithm BBPa. Among many candidates of partition algorithms, the reason for this is in its procedure: Along with circuit-partitioning, it partitions the chip area into rooms with current clusters assigned simultaneously. This feature helps us because otherwise we have to start with a formalization of the matrix as shown in Figure 2.

This BBPa is applied hierarchically to the circuit and area until small clusters are in regular small rooms. Therefore, the initial placement is fixed when the partition ends.

The procedure is illustrated in Figure 4 through Figure 6. The first figure shows an image of the input, blocks and nets. First, the chip is divided horizontally and then vertically as shown in the 2nd figure.

The objective of partitioning is set each time as to minimize the number of nets that have terminals in both subclusters. Two clusters born by BBPa are assigned to two divided subregions arbitrarily. This situation is depicted in the 2nd and 3rd figures.

For this matrix representation, get the SP with super-constraint, which is Equation (3). If the circuit designer gives a hint about the relative position, add it as additional constraint. (It is an interesting subject to discuss the way of addition, but omitted here for the space.)

3.2 Simulated Annealing

Optimization of block-placement by simulated annealing needs to make clear several issues.

Members of the solution space are not floorplans SP's. Therefore, we need to define a way to get the packing from each SP. We follow a conventional way as in [6].

The evaluation function of a floorplan is the product of total wiring length and area of the chip.

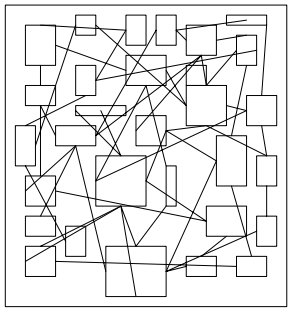

Figure 4: Input blocks and nets

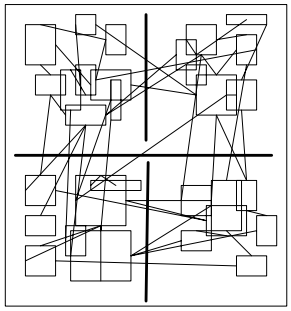

Figure 5: Clusters and rooms by one-time BBPa

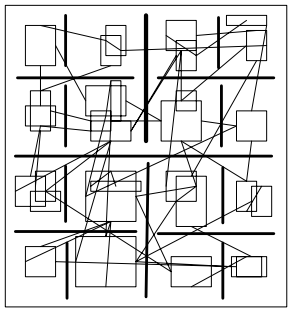

Figure 6: Clusters and rooms by two-times BBPa

Generation of solutions is by changing one SP to another. This operation is, as in [6], chosen from

1. (Full-exchange:) Take a pair of blocks such that they are not in any ordered relation in both sequences, and interchange them in both sequences.

2. (Half-exchange:) Take a pair of blocks such that they are not in any order relation in either of sequences, and interchange them in the focused sequence.

3. (Rotation:) Take a block and rotate it 90 degrees.

The parameters of the simulated annealing such as initial temperature, cooling rate, repetition, termination condition, are defined very conventionally.

4. CLOCK-TREE SYNTHESIS BY H-TREE

The input of a clock-tree synthesis is ff-terminals, which are on the flip-flops, and one source-terminal, from which the clock is supplied, that are distributed on the chip. The purpose is to construct a tree that connects all those terminals. Optimization is to make the skew minimum, where the skew is the difference of the maximum and minimum of the path length from the source-terminal to ff-terminals.

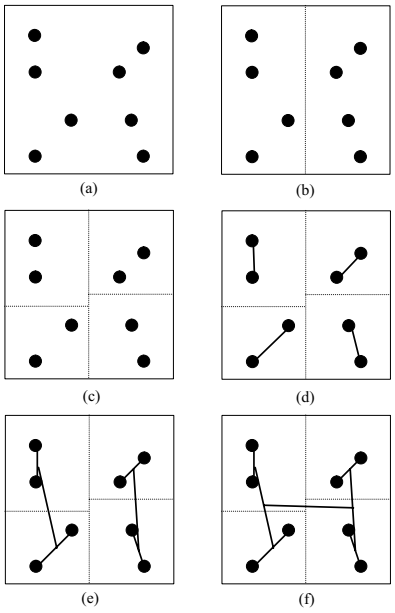

Figure 7: Behavior of Clock-Tree Synthesis MMMa

Since the problem is essential in timing plan, many algorithms have been proposed so far but we adopt the one by [3], which is called the Method-of-Means-and-Medians (MMMa). It will be shown that this is consistent with our packing algorithm, which we name as SPa-super because it is based on SP with super-constraint.

MMMa follows a strategy very similar to the H-tree algorithm. It is simply described as follows when the number of ff-terminals is even. The recursive step is: Partition the concerned region into two by a slice line such that the center of the mass lies on the line, where the mass is in effect the number of terminals and the line is horizontal or vertical so that it is orthogonal to the previous drawn line. Continue the step until there is only one terminal in each rectangular sub-region.

Then, the desired tree is constructed backwards by connecting the center of mass. A further detail will be understood by an example shown in Figure 7.

The behavior of MMMa is very similar to SPa-super. Only the difference is that SPa-super partitions the area into two of equal area while MMMa depends on the distribution of terminals.

To make the partitioning consistent with a clock-tree, we introduce a ratio-cut technique [5] to the partitioning. It removes the restriction on the area being constant. Consider to partition the set of blocks by a line into two clusters C_i and C_j. Let Π be the number of cut-nets between C_i and C_j, and H_i and H_j the numbers of ff-terminals included in respective area. Then, the optimization is set to minimize the ratio:

$$R = \frac{\Pi}{|C_i||C_j||H_i||H_j|}.$$

Thus, partitioning gives a critical balance between C_i and

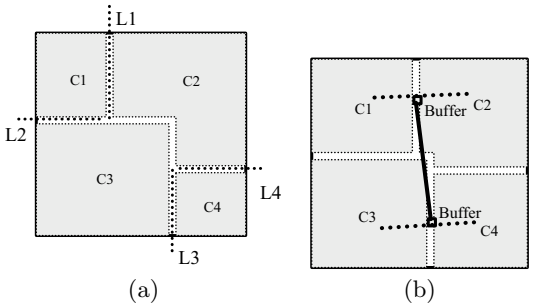

Figure 8: (a)Spaces between clusters and dividing lines, (b)Buffer area and clock-tree

C_j according to the number of flip-flops and area. Two divided regions are determined by the size of C_i and C_j, respectively.

Furthermore, we must consider about the location of buffers in a clock-tree algorithm. Buffers are inserted into a clock tree to meet zero-skew and phase-delay constraints. Buffers cannot be overlapped with any block (if hard-block) , so they must be placed between blocks.

In a step of SPa-super, two clusters by the first partitioning be c' and c''. Buffers may be inserted at the center of c' and c'' by MMMa. Applying partitioning next, let the created four clusters be c_1, c_2, c_3, and c_4. Assume that $c' = c_1 \cup c_2$ and $c'' = c_3 \cup c_4$. Accordingly, the center of c' is between c_1 and c_2, a buffer can be inserted without overlapping.

Furthermore, we can get a clock-tree close to an H-tree, even if buffers are inserted. If the super-constraint is $\alpha : (c_1(c_2 + c_3)c_4)$ and $\beta : (c_3(c_1 + c_4)c_2)$, the placement is obtained as shown in Figure 8(a). We adopt a ratio-cut technique for the balance between clusters according to the area and number of ff-terminals. So it is expected that $s(c_1) = s(c_2) = s(c_3) = s(c_4)$ and $f(c_1) = f(c_2) = f(c_3) = f(c_4)$, where $s(c)$ and f(c) are the size of cluster c and the number of flip-flips in c.

If $s(c_1) = s(c_2) = s(c_3) = s(c_4)$, then the placement is obtained as shown in Figure 8(b). Due to the super-constraint, a clock-tree is close to an H-tree. This fact is obtained by observing a relationship between cluster size and dividing lines of clusters. (In Figure 8(a), the lines are L_1, L_2, L_3, L_4.) Details are omitted here for the space.

5. EXPERIMENTS

We implemented SPa-super consisting of partitioning and floorplanning, but not yet clock-tree synthesis. Experiments were to demonstrate a basic performance for MCNC benchmark circuits. Its framework of the algorithm has been explained already.

The data of the developing environment and input circuits are shown in Table 1 and Table 2.

As the competent algorithm, we choose SPa, the Sequence-Pair based one[6]. A cost function and annealing schedule are same in both. The evaluation of total length is based on the minimum spanning tree (MST) for each net. It is the sum of Manhattan distances between pairs of terminals taken in order from the shortest pair.

SPa searches the space of size $(n!)^2$. While, for our SPa-super, the super-constraint is imposed on the one-time 2×2 partition as shown in Figure 8. Four clusters c_1, c_2, c_3 and c_4

Comp. Environment	Linux on Celeron 433MHz
Prog. Language	C++

Table 1: Computing environment

circuits	no. of blocks	no. of nets
apte	9	97
xerox	10	203
ami33	33	123
ami49	49	408

Table 2: Circuit data (MCNC)

are the ones obtained by BBPa. Hence, the super-constraint is $\alpha = (c_1(c_2 + c_3)c_4)$, $\beta = (c_3(c_1 + c_4)c_2)$. If a cluster consists of k blocks, the space our SPa-super searches is of size $(k!(2k)!k!)^2$, far smaller than $(n!)^2$ since $n = 4k$.

The results are shown numerically in Table 3. As for ami33, the resultant layouts are demonstrated in Figure 9 and Figure 10. Furthermore, the distribution maps of wire-density are shown in Figure 11 and 12.

It is observed that our method achieved superior results to SPa both with respect to length and wire-density.

The fact encourage us an important future work. The experiment in this paper was done, for simplicity, using wire-length aware BBPa. However, if we adopt timing-driven partitioning, the super-constraint will convey useful information so that we can control the block-level timing.

6. CONCLUDING REMARKS

We introduced "consistent floorplanning" on the Sequence-Pair. The main frame is based on the understanding that circuit partitioning is always connectivity-oriented bipartition and the Sequence-Pair is a data-structure of general floorplans. The idea is to convey the former feature into the latter as a constraint. By this idea, the solution space is drastically reduced, and experiments showed the effect.

This brings us to an expectation that we can control the block-level timing if we adopt timing-driven partitioning.

Future works are to implement consistent clock-tree algorithm, to get more efficient way of representing the constraint, and to show that timing-driven circuit-partitioning enables us to control block-level timing.

7. REFERENCES

[1] Breuer M. A., "A class of min-cut placement algorithms," Proc. 14th Design Automation Conference pp.284-290, 1977.

[2] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design," IEEE Trans. on Computer-Aided Design of IC's and Systems, pp.483-491, Vol. 10, No. 4, 1991.

[3] Jackson, M.A.B., A. Srinivasan, and E.S. Kuh, "Clock Routing for High-Performance ICs," Proc. 27th Design Automation Conference, pp.573-579, 1990.

[4] J. M. Kleinhans, G. Sigl, F. M. Johannes, K. J., "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," IEEE Trans. on Computer Aided Design of IC's and Systems, pp.356-365, Vol.10, No.3, 1991.

[5] Yen-Chuen Wei and Chung-Kuan Cheng, "Ratio cut partitioning for hierarchical design," IEEE Trans. on

data	apte		xerox		ami33		ami49	
super-constraint	SPa	SPa-super	SPa	SPa-super	SPa	SPa-super	SPa	SPa-super
MST(μm)	604,814	579,071	644,889	593,803	104,832	97,538	776,482	722,526
calc. time (sec.)	68.73	67.76	265.26	168.08	278.47	215.82	423.42	433.91

Table 3: Experimental results

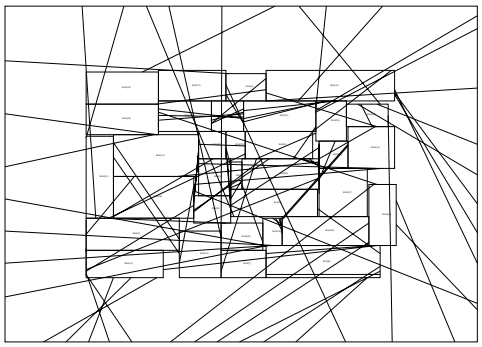

Figure 9: Result of ami33 by SPa

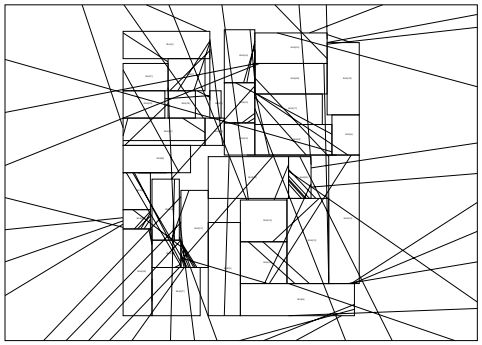

Figure 10: Result of ami33 with 2×2 super-constraint

Figure 11: Distribution map of wire-density of Figure 9

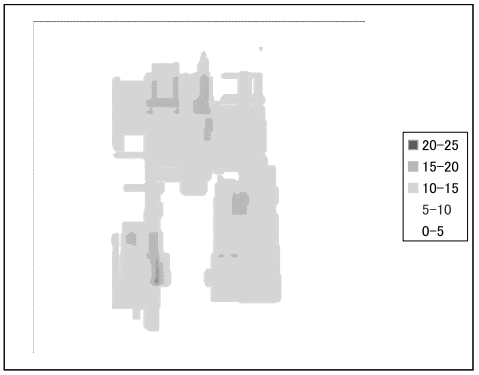

Figure 12: Distribution map of wire-density of Figure 10

Computer-Aided Design of IC's and Systems, Vol.10, No.7, pp.911-921, 1991.

[6] H.Murata, K.Fujiyoshi, S.Nakatake and Y.Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," IEEE Trans. on Computer-Aided Design of IC's and Systems, Vol.15 No.12 pp.1518-1524, 1996.

[7] S.Nakatake, H.Murata, K.Fujiyoshi, Y.Kajitani, "Module Packing Based on the BSG-Structure and IC Layout Applications," IEEE Trans. on Computer Aided Design of IC's and Systems, Vol.17, No 6, pp.519-530, 1998.

[8] H.Murata, K.Fujiyoshi and M.Kaneko, "VLSI/PCB Placement with Obstacles Based on Sequence Pair," IEEE Trans. on Computer-Aided-Design of IC's and Systems, Vol.17, No.1, pp.60-68, 1998.

[9] P.Guo, C.Cheng and T.Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications," Proc. 36th Design Automation Conference, pp.268-273, 1999.

[10] K. Sakanushi and Y. Kajitani, "The Quarter-State Sequence (Q-Sequence) to Represent the Floorplan and Applications to Layout Optimization," Proc. of IEEE Asia Pacific Conference on Circuits and Systems 2000, pp.829-832, 2000.

[11] Y.Ma, S.Dong, X.Hong, Y.Cai, C.K.Cheng and J.Gu, "VLSI Floorplanning with Boundary Constraints Based on Corner Block List," Proc. of Asia South Pacific Design Automation Conference 2001.