A layout compaction algorithm with multiple grid constraints
Jin-fuw Lee
IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

As the chip density grows, wiring circuits on a VLSI chip becomes hard. It is then important to leave feed-through channels in the layouts of cells and macros. One strategy to achieve this goal is to keep wires on their respective wiring grids. This requirement presents a new constraint to the compaction problem of cells and macros. In this paper, a new efficient algorithm is proposed to solve such compaction problem on multiple grids.

I. Introduction

The symbolic layout approach [1] would be attractive, if many complex design requirements can be accommodated. Grid constraints, which require selected objects to be placed on specified grids, will be considered here for two reasons. (1)The pin positions of cells/macros are often required to be placed on the wiring grids, so that grid-based routers [2-3] can wire them up. Even though significant progress has been made on the gridless routing problem [4], most industrial wiring tools in practical use today are still grid-based. (2)Cells/macros need to be designed in such a way to support feed-through channels. As more circuits are packed into one chip, wiring them up becomes very hard, and it is necessary to feed global wires directly through cells/macros, when wiring channels are congested. The sea of gate layout style represents an extreme case where the wiring channels are completely eliminated, and all the wires go through the interior of cells/macros. Therefore, it is desired to keep as many feed-through channels inside cells/macros as possible.

In Figure 1, vertical wires A and C are located at $x = 0$ and 3 respectively, and wire B runs between A and C. Let the minimum spacing between wires be 1. If wire B is placed at the midpoint, $x = 1.5$ (Figure 1a), then no global wire is allowed to go through either between A and B, or between B and C, and the porosity is zero. If wire B is placed at $x = 1$ (Figure 1b), then one global wire can squeeze through between B and C, and the porosity is increased to one. So if the internal wires of a cell/macro can be all placed on grids, a larger porosity can be achieved. The compaction problem with grid constraints turns out to be a mixed integer problem. Even though a general mixed integer problem is NP-hard, we will show that this particular case admits a polynomial time algorithm. A compaction algorithm dealing with one single grid constant has been given in [5].

Figure 1: (a) porosity = 0 (b) porosity = 1.

Different mask layers in general have different sets of ground rules. So we may encounter the situations with several grid constants. For example, the number of wiring layers increase from traditional two layers to multi-layers [6] with the evolution of new technologies. If four layers are supported, two of them may be used for the horizontal routing, and two of them for the vertical routing. A new problem arises, if different layers turn out to be on different pitches. Then when compacting in one direction, say the x-direction, we have to place different types of wires on different grids. In this paper, we shall present a new efficient algorithm to solve the compaction problem on multiple grids.

II. Multiple grids

In the one-dimensional compaction approach, the layout of VLSI circuits is compacted sequentially in the x and y directions. For compaction in one direction, say x, ground rule constraints of VLSI circuits are typically specified in terms of a set of constraint equations $x_i - x_j \geq l_i$ (spacing) among the positions $x_i$ of circuit components which are usually represented by a constraint graph, $G = (V, E)$. The compaction is the problem of finding $\{x_i\}$ such that the cell span is minimized while all the constraints are satisfied.

In the case of multiple grids, some nodes' positions need to be integer multiples of grid constants. Such nodes are called grid nodes, to distinguish from the rest of nodes called non-grid nodes. Let the grid constant associated with grid node $V_i$ be $D_i$. We shall make the assumption...
that the least common multiple of grid constants exists:
\[ D = \text{LCM}(D_1, D_2, \ldots) \]  This is a reasonable assumption, since all the ground rule values can be expressed in terms of some basic mask resolution unit, \( r \), which is a common divisor of \( D_1, D_2, \ldots \). Results derived in this paper are valid for the general case that \( r \) does not exist, or is too small. Simplification with the special case of a finite \( r \) will also be discussed.

Now we shall formulate an effective longest path method. Consider a path \( p = U_0U_1U_2\ldots \). Let the edge spacing of \( (U_{j-1}, U_j) \) be \( l_j \), and the sub-path from \( U_0 \) to \( U_j \) be \( p_j \). The effective lengths of paths which start from node \( U_0 \) with an initial value \( u_0 \) will be defined recursively as follows:
\[
\text{EFFL}(p_0, u_0, U_0) = u_0 \\
\text{EFFL}(p_0, u_0, U_0) \equiv \text{ROUND}(\text{EFFL}(p_{j-1}, u_0, U_0) + l_j)
\]
where the round up function \( \text{ROUND}(x, U_0) \) is defined as:
\[ x, \text{if } U_j \text{ is not a grid node; } kD_j, \text{if } U_j \text{ is a grid node and } (k-1)D_j < x \leq kD_j. \]

The longest effective path length from the source node \( V_s \) to a node \( V_t \) is then the maximum of \( \text{EFFL}(p, 0, V_t) \) taken over every path \( p \) from \( V_s \) to \( V_t \).

**Theorem 1:** (Effectively longest path)
The minimum solution, \( \bar{x} \), is given by the longest effective path length from the source node to node \( V_t \) in \( G \), if there exists a solution to the constraint equations.

There may be infinitely many paths from the source node to \( V_t \), if cycles exist in the constraint graph. For such cases, the effectively longest path may or may not exist, depending on the contribution from the cycles. If a path \( p \) consists of two path segments \( p_1 \) and \( p_2 \), we shall say that \( p \) is their union: \( p = p_1 \cup p_2 \). Let \( c \) denote a cycle with \( U_0 \) as the start node and \( u_0 \) as the initial value. Let \( c' \) denote the union of \( j \) such cycles, and \( u_j = \text{EFFL}(c', u_0, U_0) \).

If \( c \) contains grid nodes, it can be shown that there exists positive integers \( m, j, r \) such that \( u_j = u_0 + mnD(c) \) for \( j \geq j_l \) and \( n \geq 0, 0, \) where \( D(c) \) is the least common multiplier of grid constants for grid nodes on \( c \). Therefore the effective cycle length for a cycle containing grid node is defined as
\[ \text{EFFC}(c) = mD(c)/j. \]

It can be shown that the following definition for the effective cycle length is equivalent:
\[ \text{EFFC}(c) = \lim_{j \to \infty} \frac{u_j}{j} = \lim_{j \to \infty} \frac{\text{EFFL}(c', x, U)}{j}. \]

The second definition also applies for cycles containing no grid node, since it is equal to the ordinary cycle length. In Figure 2, \( x_1x_2x_4x_3x_1 \) form a cycle with nodes \( x_2 \) and \( x_4 \) on grid of 4 units and nodes \( x_1 \) and \( x_3 \) on grid of 5 units. The values of \( \text{EFFL} \) in the first four cycles for a start node and value \( x_1 = 0 \) are shown in Figure 2, from which we obtain \( J = 3, m = 1 \), and \( \text{EFFC}(c) = 20/3 \) (a fractional number).

---

**Figure 2:** Triangle nodes are on grid of 5 units and square nodes are on grid of 4 units. \( D(c) = 20, J = 3, m = 1, \) and \( \text{EFFC} = 20/3. \)

It can be proved that \( \text{EFFC} \) is a property dependent solely on the cycle itself and is independent of the choice of the start node and value. So we shall say that a cycle, \( c \), is an effectively positive cycle, if and only if \( \text{EFFC}(c) > 0 \). Then

**Theorem 2:** (Existence theorem)

The necessary and sufficient condition for the existence of the effectively longest path from the source node to any node is that there is no effectively positive cycle in the constraints graph \( G \).

### III. Complexity analysis

Since in the worst case there may be exponentially many cycles in the graph \( G \), the search of effectively longest paths can be costly. Lemmas 1-2 help us to narrow down the search further over the path space, and lead us to a fast effectively longest path search algorithm.

**Lemma 1:**

If there is no effectively positive cycle in \( G \), then the effectively longest path lengths can be found among paths, which do not traverse (a) any grid node more than \( D/D_t \) times, and (b) any non-grid node more than \( \sum D/D_t \) times, where the summation is taken over all the grid nodes.

Now let us associate each node \( V_t \) a bound number, \( b_t = D/D_t \) for a grid node, and \( b_t = \sum D/D_t \) for a non-grid node. In the special case that \( r \) exists, the latter can be lowered to \( b = \min(D(r, \sum D/D_t)). \)

When nodes are sorted in a left-to-right order, the edges of the constraint graph \( G \) can be partitioned into two sets, \( E_r \) (right directed edges) and \( E_l \) (left directed edges), with \( |E_r| < |E_l| \). The set of all paths in subgraph \( G_r = (V, E_r) \) is finite, denoted as \( Q_r \). Also let \( Q_h \) denote the set of paths in \( G \) with exactly \( k \) or less than \( k \) left directed edges. Then
Lemma 2:
If there is no effectively positive cycle in the graph G, then the longest effective path lengths can be found among $Q(\mathbf{k})$ where $M = \sum\min(b_i, b_j)$.

IV. Compaction algorithm

Since $G$ is a directed acyclic graph, the effectively longest path lengths in $Q^{(0)}$ can be systematically done with a breadth first search. Next, it is easy to see that a path in $Q^{(k + 1)} - Q^{(k)}$ can be expressed as $p^{(k)} \cup e_i \cup p^{(0)}$ where $p^{(k)}$ is a path in $Q^{(k)}$, $e_i$ is a left-directed edge, and $p^{(0)}$ is a path in $Q^{(0)}$. So the longest effective path lengths in $Q^{(k + 1)}$ can be derived from the effectively longest paths in $Q^{(k)}$ by adding at most one left-directed edge followed by another breadth first search in $G$. Based on this, a compaction algorithm is constructed in Table 1.

**PROCEDURE Compaction(G )
1 Count := 0;
2 $x_i := 0$; for i := 1 to N do $x_i := -\infty$;
3 repeat
4 Flag := true;
5 Breadth-first-Search(G,); {RightPass} ;
6 For each left-directed edge $(V_i, V_j)$ do {LeftPass} ;
7 if $x_j < ROUND(x_i + I_p, V_j)$ then
8 begin $x_j := ROUND(x_i + I_p, V_j)$; Flag := false; end;
9 Count := Count + 1;
10 until Flag becomes true or Count reaches M + 1;
11 If Flag = false then report effectively positive cycles;
12 End {Compaction}

**PROCEDURE Breadth-first-Search (G,);
1 Create a queue and push $V_i$ into queue;
2 While queue is not empty do
3 pop top node $V_i$ from the queue;
4 For each right-directed edge $(V_i, V_j)$ in G, do
5 if $x_j < ROUND(x_i + I_p, V_j)$ then
6 $x_j := ROUND(x_i + I_p, V_j)$
7 if all in-coming edges of $V_i$ are visited, then
8 push $V_i$ into queue
9 end {for}
10 end {while}
11 End {Breadth-first-Search}

Table 1 : The compaction algorithm

**Theorem 3:
The algorithm in Table 1 will converge to the minimal solution $\{x_i\}$ within $M + 1$ iterations, if and only if there is no effectively positive cycle in the constraint graph, G.

As an example, let us consider the simple constraint graph shown in Figure 3. Non-grid nodes are marked by circles. There are two grid constants: 1.0 for triangle grid nodes, and 2.0 for square grid nodes. Then we have $D_1 = 1.0, D_2 = D_3 = D_4 = 2.0$, and $D = \text{LCM}(2,1) = 2$. It is not hard to verify that all the cycles in Figure 3 have

\[ EFFC = 0. \]

So the solution exists according to Theorem 2.

The bound numbers are $b_1 = b_2 = b_3 = 1, b_4 = 2, b_5 = 5$, and $M = \min(b_i, b_j) = 2$. So the algorithm will converge to the minimum solution within at most three iterations. It did find the solution in exactly three iterations, which is the worst case bound predicted by Theorem 3. The nodes' positions are also listed in Figure 3.

**Figure 3**: Triangle nodes are on grid of 1 unit and square nodes are on grid of 2 units. Underlines mark those x positions changed during that iteration.

V Results and Conclusions

As the first example of practical application, consider a micro processor chip for which a hierarchical design system is employed to handle the complexity of layout: (1) cells are first assembled into macros, and (2) macros are then placed and wired together on a chip. Suppose that chip-level wiring among the macros is done in two layers, say M1 in the x-direction, and M2 in the y-direction. It will be helpful by making macros porous to the metal layers. As a consequence, one design methodology requires that polysilicon wires are also used in the cell-to-macro assembly phase. Naturally, because of the high resistivity, poly layer can be used only for local wirings, namely, either inside cells or among cells separated by short distances. A cell library designed to fit this methodology needs to have two grids in the x-direction, poly grid and M2 grid, and one grid in the y-direction, M1 grid, if the poly grid size as determined by the gate-to-gate spacing is larger than M2 grid size. A standard cell example is shown in Figure 4. This is an and-or CMOS cell with four input nets and one output net. Poly and M2 pins are placed on the same M1 grid in the y-direction, but on different grids in the x-direction. To improve the chance for global wiring, many equivalent pins are provided in each input and output net.

As the second example, consider some ASIC chip designs in which the density is not as important as a fast design cycle, because of the narrow market window these products are facing. One strategy used in speeding up the design cycle time is to have porous cells so that the chip-
level physical design can be automatically done in a turnkey fashion. Assume that, for chip-level wiring, we use M1 layer in the x-direction, M2 layer in the y-direction, and no poly layer for simplicity. Then such porous cell library requires that the x-pin locations be placed on \( N_x \) times M2 pitches and the x-cell boundaries on \( N_{cell} \) times M2 pitches, where either \( N_x \) or \( N_{cell} \) is greater than 1. An \((N_x = 1, N_{cell} = 3)\) example is shown in Figure 5. This is an exclusive-or cell with three input nets and one output net. Two pins are generated for each net. During the x-compaction, the slack spaces imposed by \( N_{cell} = 3 \) are distributed among the diffusion breaks and the static power nodes to minimize the capacitance loading.

In conclusion, an efficient algorithm is developed to handle the compaction problem on multiple grids. The worst case time complexity of the algorithm is \( O((M + 1) (|P| + |E|)) \). The algorithm has been implemented to a compactor [7] and applied to the layout designs for both micro-processor chips and ASIC chips.

Acknowledgements

We would like to thank Dr. E. Hockenek, Dr. D.T. Tang, Dr. D. Ostapko, and Dr. Z. Barzilai of IBM Research Division for encouragement and support. We would like to thank design communities in IBM Advanced Workstation Division and General Technology Division for their interest and feedback.

References