

has $2^{n/2}$ elements. Assume that there are p variables corresponding to the a_i 's and q variables corresponding to the b_j 's in L . We know that $0 \leq p \leq n$, $0 \leq q \leq n$, and $p + q = n$. Further, we have q a_i variables in R and p b_j variables in R .

We will show that for any partition (L, R) , we will always be able to pick a g_z such that $\geq n/2$ terms in g_z satisfy the property that the a_i variable in the term belongs to L , and the b_j variable to R , or vice versa. That is, these terms will be split across the partition. The number of terms in all the g_k 's, where any term $a_i \cdot b_j$ has $a_i \in L$ and $b_j \in L$ is pq . Similarly the number of terms in all the g_k 's $a_i \cdot b_j$ such that $a_i \in R$ and $b_j \in R$ is pq . We thus have $2pq$ terms that are not split across the partition. Given that $p + q = n$, the maximum of $2pq$ occurs at $p = q = n/2$. Since the total number of terms in f is n^2 , the number of terms split across the partition is $\geq n^2 - 2(n/2)(n/2) = n^2/2$. In general for different selections of L and R , these terms will be contained in different g_k 's. Given there are ng_k 's we will always have at least one g_z which will have at least $(1/n)(n^2/2)$ terms split across the partition.¹

Now that we have a selected g_z , we use the settings for the mux_i variables that make $f = g_z$. We will now show that we have a fooling set $A_{\text{OBDD}}(L, R)$ for f that is of cardinality $2^{n/2}$. The fooling set corresponds to all possible settings of the $n/2$ a_i 's or b_j 's in L that are contained in terms of g_z split across the partition. The a_i and b_j variables in L not in the $n/2$ terms that are split across the partition are set to constant 0 values for all possible settings of the variables that are in the split terms.

Assume we have two settings of the variables in L , namely l and l' . We pick a_m such that $l(a_m) \neq l'(a_m)$. The term $a_m \cdot b_s$ belongs to g_z , such that $b_s \in R$. We set the values of all a_i and b_j variables in R to 0, except for $b_s = 1$. Call this assignment (along with the mux_i variable assignment) r . Because $b_s = 1$ and all terms in g_z except $a_m \cdot b_s$ have been set to 0, $g_z(l \cdot r) \neq g_z(l' \cdot r)$. Therefore, $f(l \cdot r) \neq f(l' \cdot r)$. We can pick a b_m instead of a_m above and use similar arguments to show that two settings l and l' that differ in b_m can be differentiated. Thus, we can differentiate any two members of the constructed fooling set.

Given that we have a fooling set $A_{\text{OBDD}}(L, R)$ with $2^{n/2}$ elements invoking the lemma above gives us the desired result. \square

IV. SUMMARY

Families of logic functions having OBDD representations that grow linearly with the number of inputs (under particular orderings), but sum-of-product representations that grow exponentially with the number of inputs, have been known for a long time. We have provided an example of a function that has a quadratic-sized sum-of-product representation, but an exponential-sized OBDD representation, under any possible variable ordering. A larger class of functions similar to the one above, where f is a multiplexor composition of n/k functions (k is a constant) that have nk product terms and $\Omega(2^{n/2k})$ vertices in any OBDD representation, can be derived.

V. ACKNOWLEDGMENT

Discussions with Pranav Ashar and Abhijit Ghosh that led to the discovery of the function analyzed here are acknowledged. Thanks to Randy Bryant for suggesting the use of the theory developed in [3].

¹In fact, the minimum corresponds to the case where all the g_k 's have exactly $n/2$ terms split across the partition.

REFERENCES

- [1] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Norwell, MA: Kluwer Academic, 1984.
- [2] R. Bryant, "Graph-based algorithms for Boolean function manipulation," in *IEEE Trans. Computers*, vol. C-35, pp. 677-691, Aug. 1986.
- [3] R. Bryant, "On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication," in *IEEE Trans. Computers*, vol. 40, pp. 205-213, Feb. 1991.
- [4] R. Lipton and R. Sedgewick, "Lower bounds for VLSI," in *Proc. ACM Symp. Theory of Computation*, pp. 300-307, June 1981.

Physical Models and Efficient Algorithms for Over-the-Cell Routing in Standard Cell Design

Jason Cong, Bryan Preas, and C. L. Liu

Abstract—When an over-the-cell routing layer is available for standard cell layout, efficient utilization of that routing space over the cells can significantly reduce layout area. In this paper, we present three physical models to utilize the area over the cells for routing in standard cell designs. We also present efficient algorithms to choose and to route a planar subset of nets over the cells so that the resulting channel density is reduced as much as possible. For each of the physical models, we show how to arrange inter-cell routing, over-the-cell routing, and power/ground buses to achieve valid routing solutions. Each algorithm exploits the particular arrangement in the corresponding physical model and produces provably good results in polynomial time. We tested our algorithms on two industrial standard cell designs. In these tests, this method reduces total channel density by as much as 21%.

I. INTRODUCTION

Standard cells are widely used in the design of VLSI circuits. After the cells are placed in rows and necessary feedthroughs inserted, a channel router completes the interconnections in the channels among the cells (Fig. 1). Conventional channel routers are restricted to utilizing two routing layers in the channels for interconnections. The conventional channel routing problem has been extensively studied, and there are several channel routers which can produce solutions using at most one or two tracks more than channel density for most practical problems (for example, see [7], [23], [20], [1], [19]). To further reduce the channel routing area, some channel routers use the extra routing area over the cells for interconnections [9], [15], [21], [14], [3], [5]. These routers are

Manuscript received December 11, 1990; revised December 2, 1991. This work was partially supported by the National Science Foundation under Grants MIP 87-03273 and MIP 91-10511, by the Semiconductor Research Corporation under Contract 87-DP-109, and by a grant from the Alexander von Humboldt Foundation. This paper was recommended by Associate Editor M. Marek-Sadowska.

J. Cong is with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90024.

B. Preas was with the University of Paderborn, Germany, when the work was performed. He is currently with the Xerox Palo Alto Research Center, Palo Alto, CA 94304.

C. L. Liu is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

IEEE Log Number 9200915.

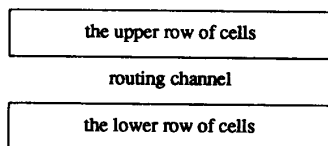


Fig. 1. Channel routing between cell rows.

called *over-the-cell channel routers*. [9] presents an algorithm that produces single-layer planar routing over the cells for I^2L and LST^2L logic arrays. A gate array router presented in [15] utilizes horizontal and vertical over-the-cell routing channels to increase cell density. In [21], an over-the-cell channel router called a permeation router was presented. [14] studied the problem of choosing net segments to be connected in the channel after over-the-cell routing. In [3] and [5], a symbolic model for over-the-cell channel routing was presented together with the algorithms for each stage of the entire routing process. These routers can typically complete the connections using fewer tracks than the channel density. In standard cell designs, because a large portion of VLSI circuit area is used for channel routing, the savings in area obtained by over-the-cell routing can be quite significant. However, existing over-the-cell routers can produce only symbolic routing solutions or solutions for a particular design technology. General physical models for over-the-cell routing in standard cell design have not been carefully studied.

In this paper, we present three physical models for over-the-cell routing in standard cell designs. These models are general enough to be applied to most cell families. In each of these models, we show how to arrange inter-cell routing, over-the-cell routing, feedthroughs, and power/ground buses to achieve valid routing solutions. We discuss the advantages and limitations of each arrangement. We also present three different algorithms (one for each of the three models) to choose and to route some of the connections over the cells. These three algorithms are all based on the idea of finding a maximum-weighted planar subset of nets to be routed over the cells. The effect is that the resulting channel density is reduced as much as possible. One algorithm uses a dynamic programming method. The other two algorithms are based on the maximum-weighted h -family computation in a partially ordered set. We tested our algorithms on several industrial circuits, including the benchmark Primary1 from the Physical Design Workshop [18]. The algorithms reduced the total channel density as much as 21% for the examples in our test suite.

The remainder of this paper is organized as follows. In Section II, we discuss three physical models for over-the-cell routing in detail. In Section III, we present the algorithms to solve the over-the-cell routing problem for each model. In Section IV, we present a polynomial time algorithm for computing a maximum weighted h -family in a partially ordered set. Experimental results are presented in Section V. An extended abstract of this paper was presented in the proceedings of the 1990 Design Automation Conference [6].

II. THREE PHYSICAL MODELS FOR OVER-THE-CELL ROUTING

In the standard cell design style, cells are placed in rows and channels are formed between adjacent cell rows. There are three routing layers: one layer of polysilicon and two layers of metal. In conventional layout design, intra-cell routing is carried out using all routing layers within the cells, and inter-cell routing is carried out using the routing layers in the channels. It has been observed that intra-cell routing can be completed using one layer of polysil-

icon and one layer of metal.¹ Therefore, it is possible to use the other layer of metal over the cells for inter-cell routing to reduce channel routing area. In this section, we present three general physical models. Each shows a different way of realizing over-the-cell routing according to the physical design rules. In these models, intra-cell routing, feedthroughs, power/ground buses, and over-the-cell routing are arranged carefully so that the final routing solutions are valid and compact.

For convenience of discussion, horizontal wires in the channels are referred to as *trunks*, and vertical wires in the channels are referred to as *branches*. The channel above a row of cells is called the *upper channel*, and the channel below these cells is called the *lower channel*. Moreover, the terminals on the upper edge of the cells are the *upper terminals*, and the terminals on the lower edge are the *lower terminals*. P, M1, and M2 denote the polysilicon layer, the metal-1 layer, and the metal-2 layer, respectively. In all three models, we assume that intra-cell connections are completed on the P and M1 layers, and over-the-cell are completed on the M2 layer.

While there are many ways that standard cell families can be designed for over-the-cell-routing, these three layout models represent the most important variations, given modern integrated circuit fabrication technology. The following three layout models were selected for study because other models 1) are not appropriate for use with over-the-cell-routing (for example, interfere with the M2 routing above the cells); 2) have obvious deficiencies compared to these models; or 3) can be represented as variations of these models.

2.1. The Horizontally Connected, Vertically Divided (HCVD) Model

In this model, we place cell terminals on the M2 layer and feedthroughs on the M1 layer. In the channels, we route trunks on the M1 layer and branches on the M2 layer. Power/ground buses are routed on the M2 layer at the middle of the cell row. Over-the-cell connections are on the M2 layer. Clearly, the over-the-cell routing region for each row of cells spans horizontally over the entire row of cells. However, it is divided into two parts vertically by the power and ground buses at the middle of the cells.² We call this over-the-cell routing model the horizontally connected, vertically divided (HCVD) model since the over-the-cell routing region is connected in the horizontal direction and disconnected in the vertical direction. Fig. 2 shows a valid over-the-cell routing solution in the HCVD model. As we can see, in the HCVD model, the over-the-cell connections for the upper and lower terminals are separated by power and ground buses in the middle of the cell rows on the M2 layer. The HCVD model has two advantages. First, cell terminals and the over-the-cell connections are on the same layer (M2). Consequently, extra (or larger) vias are not needed to bring the terminals to the over-the-cell routing layer. Second, the over-the-cell routing region horizontally spans the entire row of cells. This long span is advantageous when compared with another model in which the over-the-cell routing region is divided horizontally into smaller regions. This model also has shortcomings. When the power or ground buses on M2 are connected to the diffusions, extra vias are required. Also, extra vias are required to bring the terminals of feedthroughs up to the M2 layer.

¹Intra-cell connections can also be completed in the channels (by adding these connections to the netlist) if the cell height is too small for the required routing.

²The heights of the two parts may be different because the sizes of P-transistors and N-transistors may not be the same.

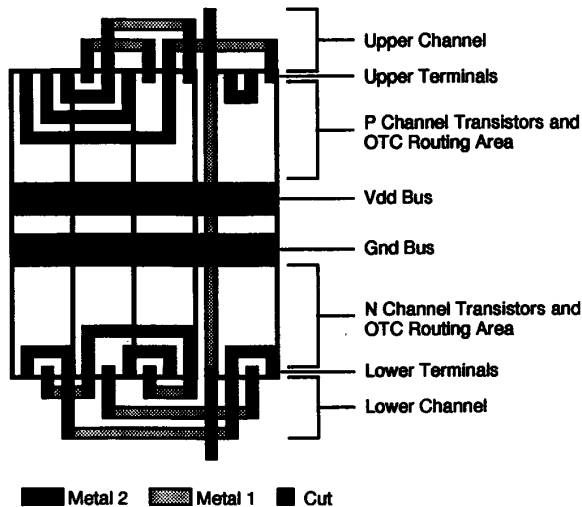


Fig. 2. A valid over-the-cell routing solution in the HCVD model.

2.2. The Horizontally Connected, Vertically Connected (HCVC) Model

In this model, cell terminals and feedthroughs are both on the M1 layer. In the channels, trunks are routed on the M2 layer and branches are routed on the M1 layer. The power bus is on the M2 layer in the upper channel just above the upper terminals, and the ground bus is on the M2 layer in the lower channel just below the lower terminals. Over-the-cell routing is carried out on the M2 layer. Clearly, the entire M2 layer over the cells is available for over-the-cell routing. Therefore, we call this over-the-cell routing model the *horizontally connected, vertically connected* (HCVC) model. Fig. 3 shows a valid over-the-cell routing solution in the HCVC model. This model has several advantages. First, we have the entire M2 layer available for over-the-cell routing, which maximizes opportunity for over-the-cell connections. Second, over-the-cell routing for upper and lower terminals can share the M2 layer, which makes better utilization of the over-the-cell routing region. Third, terminals of the cells and feedthroughs are all on the M1 layer, so we can connect them to branches without using extra vias. However, this model has three shortcomings. First, cell terminals and over-the-cell connections are on different layers; thus, we need extra vias to bring the terminals up to M2. Second, power and ground buses (which are usually wider than signal routing) take up two tracks or more per channel, which will partially offset the reduction in channel height achieved by over-the-cell routing. Furthermore, cells will probably be wider than the cells of the other models because of the need to bring power and ground signals into the cells.

2.3. The Horizontally Divided, Vertically Connected (HDVC) Model

In this model, both cell terminals and feedthroughs are on M2. In the channels, trunks are on the M1 layer and branches are on the M2 layer. Power and ground buses are on the M1 layer at the upper and lower edges of the cells. Over-the-cell routing is carried out on the M2 layer between adjacent feedthroughs. Clearly, the over-the-cell routing region for each row of cells is divided horizontally (by feedthroughs) into many smaller regions. There is no vertical separation within regions. We call this over-the-cell routing model the *horizontally divided, vertically connected* (HDVC)

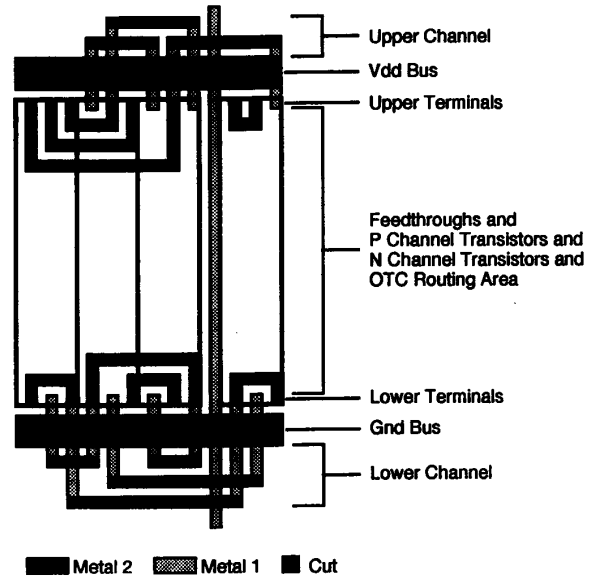


Fig. 3. A valid over-the-cell routing example in the HCVC model.

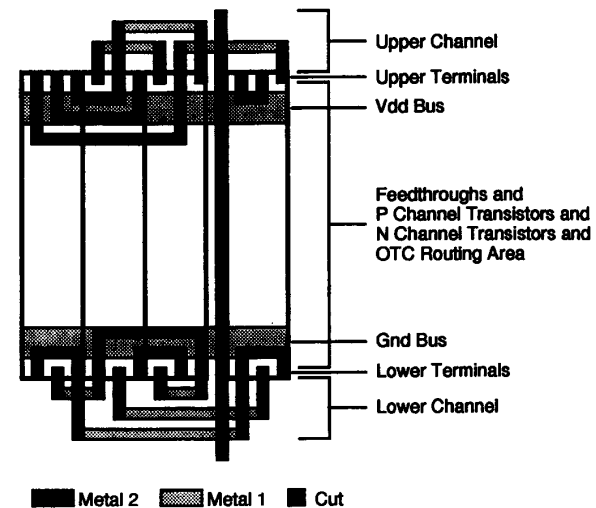


Fig. 4. A valid over-the-cell routing example in the HDVC model.

model since the over-the-cell routing region is disconnected in the horizontal direction and connected in the vertical direction. Fig. 4 shows a valid over-the-cell routing solution in the HDVC model. This model has two advantages. First, since the cell terminals, feedthroughs, branches, and over-the-cell connections are on the same layer, no extra vias are needed to connect them. Second, since the power and ground buses are on the M1 layer, they can connect to diffusions easily. However, in this model, routing can only be carried out between two adjacent feedthroughs. Consequently, a large number of feedthroughs makes it difficult to utilize the over-the-cell routing area. Also, some M1 routing area within the cells is lost, since power and ground buses are routed in the M1 layer within the cells.

We summarize the features of the three physical models for over-the-cell routing in Table I.

TABLE I
THREE MODELS FOR OVER-THE-CELL ROUTING

Model	Terminals	Feedthroughs	Power/Ground Buses	Over-the-Cell Routing
HCVD	M2	M1	M2, over cells	M2
HCVC	M1	M1	M2, in channels	M2
HDVC	M2	M2	M1, within cells	M2

Note that we assume in our models that all the standard cells are of the same height. This implies that the channel routing region and the over-the-cell routing region are rectangular in shape. A more general model would allow cells of different heights so that the channel routing region and the over-the-cell routing region could be more-complex rectilinear polygons (in this case, channel routing and over-the-cell routing can be carried out by a general-area router, such as a maze router). Also, we assume in our models that all the pins for over-the-cell routing are available at the boundary of cells. A more general model would allow poly-metal contacts at appropriate places inside the cells for over-the-cell connections. This may lead to better utilization of the over-the-cell routing area. However, in this case, the over-the-cell routing problem becomes a general-area planar routing problem (as compared to the one-row or two-row planar routing problems formulated in the next section) and it is computationally difficult to compute an optimal or near-optimal over-the-cell routing solution (see the NP-completeness result in [17]). Again, a maze router can be used in this case.

III. ALGORITHMS FOR OVER-THE-CELL ROUTING

3.1. Algorithm Overview

Since the over-the-cell channel routing problem is NP-hard [14], a common approach is to partition the problem into several sub-problems and solve each one separately. Cong and Liu [5] proposed one such partition, which worked very well in their symbolic over-the-cell channel routing model. The problem was solved in three stages. The first stage was to route over the cells. In this stage, the terminals on each side of a channel were connected using the over-the-cell routing region on that side of the channel. The same procedure was carried out for the upper and lower sides of the channel independently. Fig. 5 shows a routing solution for one side of the channel after the first stage. Note that the routing solution generated in the first stage is always a planar routing. Each maximal set of terminals connected together by the planar routing is called a *hyperterminal*. For example, {5.4, 5.6, 5.11} is a hyperterminal, where $a \cdot c$ denotes the terminal of net a at column n . The objective in the first stage is to minimize the number of resulting hyperterminals. Intuitively, this procedure is equivalent to maximizing the number of connected terminals in the planar routing. The problem thus formulated was reduced to the problem of finding a maximum independent set of a circle graph [10], which can be solved in quadratic time. The second stage was to choose which net segments were to be connected in the channel. A *net segment* is a set of two terminals of the same net that belong to two different hyperterminals. For example, Fig. 6 shows the four possible net segments that can be used to connect the two hyperterminals of net 1. Since all the terminals in each hyperterminal are connected in the over-the-cell routing region already, the problem in the second stage was to choose a subset of net segments to connect all the hyperterminals of each net such that the resulting channel density would be minimum. It was shown that the general net segment selection problem is NP-complete, and an efficient heu-

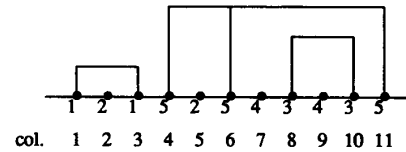


Fig. 5. Planar routing over cells.

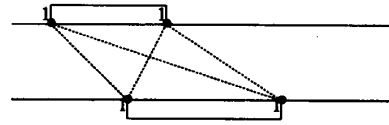


Fig. 6. Possible net segments connecting two hyperterminals.

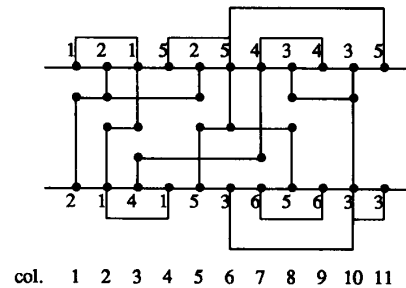


Fig. 7. Example of symbolic over-the-cell channel routing solution.

ristic algorithm was presented based on finding a minimum-density spanning forest. The third stage was to connect the terminals corresponding to the selected net segments. Clearly, the problem in this stage is equivalent to the conventional two-layer channel routing problem. Therefore, it can be solved by well-known two-layer channel routers. Fig. 7 shows a symbolic over-the-cell channel routing solution.

Because the symbolic model in [5] captures the essential issues in over-the-cell routing, we will follow the same three-stage approach in this paper.³ In fact, the algorithms for the second and third stages can be applied directly to all three physical models described in the preceding section. However, their algorithm for the over-the-cell routing stage (the first stage) is too simple to be applied in any of the three physical models described in the preceding section. (They did not account for layout design details, such as the arrangement of feedthroughs and power/ground buses. Moreover, they assumed that there are an infinite number of tracks in the over-the-cell routing area.) In the remainder of this section, we present three algorithms for the over-the-cell routing stage for the HCVD, HCVC, and HDVC models. Each of these algorithms

³Ideally, we would like to combine over-the-cell routing and net segment selection in one step to minimize the resulting channel density. However, the computational complexity of the combined problem is very high. In general, each net segment can be routed in three possible regions—over the lower cells, in the channel, and over the upper cells. In the lower or upper over-the-cell routing region, we may have 4 to 5 tracks and each track may route several segments, so that we may easily have over 20 net segments. In this case, the height of search tree is at least 40. Therefore, the total routing configurations is at least $3^{40} \approx 1.2 \times 10^{19}$, which is too large for exhaustive search. Although branch-and-bound or other search techniques may prune a large number of non-optimal routing configurations, in most cases, it would be still too expensive to compute an optimal solution of the combined problem. Therefore, we choose to use the divide-and-conquer technique to construct an approximate routing solution in three stages.

exploits the particular arrangement in the corresponding physical models and produces provably good results in polynomial time.

3.2. Connection Weight Computation

The objective of the first stage in [5] is to minimize the number of resulting hyperterminals, or, equivalently, to maximize the number of terminals connected over the cells. Although the number of hyperterminals over the cells is related to the resulting channel density, different choices of hyperterminals may result in different channel densities. In general, an over-the-cell connection between a pair of terminals that spans the densest columns in the channel is more likely to reduce channel density. Therefore, a more accurate objective of the first stage would be based on the density distribution in the channel [8].

Let R be the row of terminals of the lower (or upper) side of a channel C . Let $a \cdot c$ denote the terminal of net a at column c . We define the *weight* of a pair of terminals $a \cdot x_1$ and $a \cdot x_2$ of the same net to be

$$w(a \cdot x_1, a \cdot x_2) = \frac{d(x_1, x_2)}{D}, \quad x_1 < x_2$$

where $d(x_1, x_2)$ denotes the maximum of the local densities in the channel between column x_1 and column x_2 , and D denotes the channel density (before over-the-cell routing). Clearly, the weight of a pair of terminals measures the degree of congestion in the channel between these two terminals. Intuitively, connecting a pair of terminals with larger weight in over-the-cell routing provides a better chance of reducing the resulting channel density. Given a planar routing solution S , a *connected pair* in S is a pair of terminals *adjacent* in a hyperterminal in S . For example, in Fig. 5, the terminals $5 \cdot 4$ and $5 \cdot 6$ are a connected pair, but the terminals $5 \cdot 4$ and $5 \cdot 11$ are not. Clearly, a hyperterminal consisting of k terminals contains $k - 1$ connected pairs. We define the weight of a hyperterminal to be the sum of the weights of the connected pairs contained in the hyperterminal. In other words, let $H = \{a \cdot x_1, a \cdot x_2, \dots, a \cdot x_l\}$ be a hyperterminal in $S(x_1 \leq x_2 \leq \dots \leq x_l)$. The *weight* of H is defined to be $w(H) = \sum_{k=1}^{l-1} w(a \cdot x_k, a \cdot x_{k+1})$. Finally, we define the weight of the solution S , denoted $w(S)$, to be the sum of the weights of the hyperterminals in S . The objective of our algorithms in this paper is to maximize the weight of the planar routing solution. Clearly, the effect is that the resulting channel density is reduced as much as possible.

3.3. The Algorithm for the HCVD Model

In the HCVD model, for each row of cells R , the over-the-cell routing region is divided vertically into two sub-regions by the power and ground buses. Therefore, over-the-cell routing for the lower terminals of R and the upper terminals of R can be carried out independently in the lower and the upper sub-regions, respectively. Moreover, the height of each sub-region is limited by the half height of cells. Thus, the over-the-cell routing problem in the HCVD model is to find a planar routing S to connect a subset of the nets on a row of terminals, using a fixed number of tracks on one side of the terminals such that the weight of S is maximum. We call this problem the *one-row fixed-height planar routing (OFPR) problem*. Fig. 8 shows a valid routing solution to an OFPR problem.

First, we transform the multiterminal net OFPR problem into the two-terminal net OFPR problem, because we shall see later on in this section that the two-terminal OFPR problem can be solved efficiently using the dynamic programming technique. Our transfor-

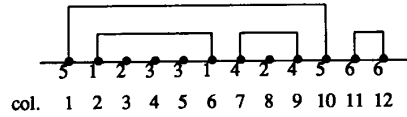


Fig. 8. A valid solution to OFPR (height is 2).

mation procedure is similar to the one in [5] and it shall be explained in the proof of the following theorem:

Theorem 1: For any instance I of the multiterminal net OFPR problem, I can be transformed to an instance I' of the two-terminal net OFPR problem in $O(n^2)$ time such that I' contains $O(k \cdot n)$ terminals, where n is the number of terminals in I and k is the maximum size of a net in I' .

Proof: Given an instance I of the multiterminal net OFPR problem, we construct an instance I' of the two-terminal net OFPR problem as follows: Let a be a net with k terminals ($k \geq 3$). Let p_1, p_2, \dots, p_k denote the k terminals. We shall replace net a by $(k(k-1))/2$ two-terminal nets which will be denoted $a - i - j$, ($1 \leq i < j \leq k$). Specifically, we split p_l , ($1 \leq l \leq k$), into $k - 1$ terminals $p_{l,1}, p_{l,2}, \dots, p_{l,k-1}$. (These terminals are placed next to each other as a group.) $p_{l,i}$ is assigned to net $a - i - l$ for $1 \leq i \leq l - 1$, and $p_{l,j}$ is assigned to net $a - l - (j + 1)$ for $1 \leq j \leq k - l$. Moreover, we assign the weight of net $a - i - j$ in I' to be $\sum_{k=1}^{l-1} w(p_l, p_{l+1})$ in I . Clearly, I' thus constructed is an instance of the two-terminal net OFPR problem. Furthermore, using an argument similar to the one in the proof of theorem 1 in [5], we can show that I' contains $O(k \cdot n)$ terminals and that the problem of finding a maximum weighted planar routing in I is equivalent to the one of finding a maximum weighted planar subset of I' . \square

According to this theorem, we need only concentrate on the two-terminal net OFPR problem. In the rest of this subsection, we will describe a dynamic programming approach to the two-terminal net OFPR problem. For an instance I of the two-terminal net OFPR problem, let n be the number of terminals in I and t be the number of tracks allowed. Let $I(i, j, s)$ denote the instance of the two-terminal net OFPR problem resulting from restricting I to the interval $[i, j]$ and allowing s tracks for routing. Clearly, our goal is to find a maximum-weighted routing solution for the instance $I(1, n, t)$. Let $M(i, j, s)$ denote the maximum weight of any routing solution for $I(i, j, s)$. Assume that the net at column i is net a and that the other terminal of net a is at column i' . If i' is not in the interval $[i, j]$, we cannot connect net a in any of the solutions for $I(i, j, s)$. Thus a routing solution for $I(i, j, s)$ is also a routing solution for $I(i + 1, j, s)$. Therefore,

$$M(i, j, s) = M(i + 1, j, s) \quad \text{if } i' \notin [i, j]. \quad (3.1)$$

Suppose that i' is in the interval $[i, j]$. Let S be a maximum-weighted routing solution for $I(i, j, s)$. S may or may not connect net a . If net a is not connected in S , clearly, we still have $M(i, j, s) = M(i + 1, j, s)$. If net a is connected in S , then the connections for the terminals in $[i + 1, i' - 1]$ in S are separated from the connections for the terminals in $[i' + 1, j]$ in S because of the restriction of planar routing. Moreover, the connections in $[i + 1, i' - 1]$ use at most $s - 1$ tracks and the connections in $[i' + 1, j]$ use at most s tracks. Furthermore, it is not difficult to see that the connections in $[i + 1, i' - 1]$ in S form a maximum-weighted routing solution for $I(i + 1, i' - 1, s - 1)$ and the connections in $[i' + 1, j]$ in S form a maximum-weighted routing solution for $I(i' + 1, j, s)$. (Otherwise, the weight of S can be augmented by replacing these connections with their corresponding maximum-

weighted routing solutions.) Therefore, in the case that net a is connected in S , we have $M(i, j, s) = M(i + 1, i' - 1, s - 1) + M(i' + 1, j, s) + w(a \cdot i, a \cdot i')$. Thus we have the following equation:

$$M(i, j, s) = \max \{M(i + 1, j, s), M(i + 1, i' - 1, s - 1) + M(i' + 1, j, s) + w(a \cdot i, a \cdot i')\} \quad \text{if } i' \in [i, j]. \quad (3.2)$$

Based on (3.1) and (3.2), we can obtain the following result:

Theorem 2: The two-terminal net OFPR problem can be solved in $O(t \cdot n^2)$ time, where n is the number of terminals and t is the number of available tracks.

Proof: If n is the number of terminal and t is the number of available tracks in a two-terminal net OFPR problem, the maximum weight of solutions is $M(1, n, t)$. According to the two recursive equations, $M(1, n, t)$ can be computed using the dynamic programming method as follows:

For $s = 1$ to t do

 For $k = 1$ to $n - 1$ do

 For $i = 1$ to $n - k$ do

 Let $j = i + k$; Assume that net a has one terminal at column i and the other at column i' ;

 If $i' \notin [i, j]$ then $M(i, j, s) = M(i + 1, j, s)$

 Else $M(i, j, s) = \max \{M(i + 1, j, s), M(i + 1, i' - 1, s - 1) + M(i' + 1, j, s) + w(a \cdot i, a \cdot i')\}$.

It is easy to see that this procedure takes $O(t \cdot n^2)$ time. Moreover, by keeping proper information at each step (i.e., remembering if we have chosen net a in the connection or not), not only can we compute the value of $M(1, n, t)$, but also we can construct the solution which achieves $M(1, n, t)$ at the end of the procedure. \square

Corollary 1: The multiterminal net OFPR problem can be solved in $O(t \cdot k^2 \cdot n^2)$ time, where n is the number of terminals, k is the maximum size of a net, and t is the number of available tracks.

It is well known that for industrial circuits, the average number of terminals per net is between 2 and 4, and the maximum number of terminals per net is bounded by a constant (typically between 8 and 16). Moreover, t is bounded by a constant which is determined by the half height of cells (usually t is no more than 10). Therefore, for all practical examples, the multi-terminal net OFPR problem can be solved in $O(n^2)$ time.

Based on these results, our over-the-cell routing algorithm for the HCVD model works as follows. We route the upper and lower terminals of each row of cells independently. For each row of terminals, first we reduce the multi-terminal net OFPR problem to the two-terminal net OFPR problem. Then, we apply the dynamic programming approach to compute an optimal solution of the two-terminal net OFPR problem. According to Theorems 1 and 2, these steps can be carried out in $O(n^2)$ time in practice.

3.4. The Algorithm for the HCVC Model

In the HCVC model, for each row R of cells, the entire M2 layer over R can be used for over-the-cell routing. Moreover, both the lower terminals of R and the upper terminals of R share the over-the-cell routing region. Therefore, over-the-cell routing for both the lower and upper terminals of R has to be carried out simultaneously so that the common routing region will be used efficiently. Furthermore, the number of tracks in the over-the-cell routing re-

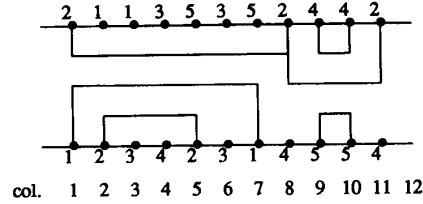


Fig. 9. A valid solution to TFPR (height is 3).

gion is limited by the height of cells. Thus the over-the-cell routing problem in the HCVC model can be formulated as follows. Given two rows of terminals, we want to find a planar routing S to connect a subset of nets on each row of terminals using a fixed number of tracks. This problem is called the *two-row fixed-height planar routing (TFPR) problem*. Fig. 9 shows a valid solution to a TFPR problem. Note that in the TFPR problem, terminals on the two different rows do not have to be connected although they may belong to the same net. This is because the two rows of terminals belong to two different channels, and connections for the same net in different channels have been accomplished by adding feedthroughs in the global routing phase (for example, see [2]). Therefore, a routing solution S to the TFPR problem can be partitioned into the union of two planar routing solutions S^l and S^u for the lower and upper rows of terminals. We define the weight of S to be sum of the weights of S^l and S^u (i.e., $w(S) = w(S^l) + w(S^u)$). An *optimal solution* to the TFPR problem is a solution whose weight is maximum. For a solution S to a TFPR problem, we use $d(S)$ to denote the density of S . Clearly, if $d(S)$ is no more than the number of tracks given in the problem, S is a valid solution to the TFPR problem. This is due to the fact that there is no vertical routing constraint in the TFPR problem since we do not have to connect terminals in the two different rows. When the number of available tracks is not limited in the TFPR problem, we call the resulting problem the *two-row unlimited-height planar routing (TUPR) problem*. Clearly, the weight of any optimal solution to a TFPR problem is no more than the weight of an optimal solution to the corresponding TUPR problem. Fig. 10 shows a valid solution to the corresponding TUPR problem of the example in Fig. 9 (we labeled each connected pair for later reference).

However, solving the TFPR problem optimally is very difficult. So far, we have not obtained any polynomial time algorithm to solve the TFPR problem optimally. The complexity of the TFPR problem is still unknown. Therefore, we solve the TFPR problem in two steps to obtain an approximate solution. For a given TFPR problem, let h denote the number of available tracks between the two rows. Let S^* denote an optimal solution and S denote the solution to be constructed by our algorithm. Our algorithm can be outlined as follows. In the first step, we compute an optimal solution \bar{S} to the corresponding TUPR problem (i.e., assuming that the height is unlimited). If $d(\bar{S}) \leq h$, we simply let S equal \bar{S} and output S . Clearly, in this case, S is an optimal solution to the given TFPR problem. If $d(\bar{S}) \geq h$, we choose S to be a subset of connected pairs from \bar{S} such that $d(S) = h$ and $w(S)$ are maximum. Clearly, S is a valid solution to the TFPR problem. We shall show later that such an S can be computed in polynomial time. Moreover, we shall show that the weight of the solution S thus constructed will not be too small. In fact, we shall show that $w(S)/w(S^*) \geq h/d(\bar{S})$. Therefore, in both cases, we have

$$\frac{w(S)}{w(S^*)} \geq \min(1, h/d(\bar{S})).$$

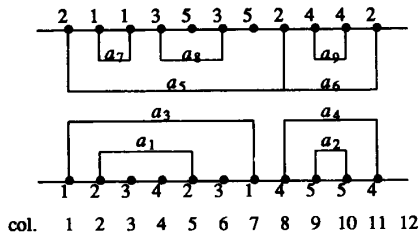


Fig. 10. A valid solution to the corresponding TUPR of the example in Fig. 9.

This bound ensures that the weight of our solution is very close to the weight of an optimal solution to the original TFPR problem because of the following observation. In most cases, the density $d(\bar{S})$ of the optimal solution \bar{S} to the corresponding TUPR problem will not be too high. Although there is no limit on the number of tracks to be used in \bar{S} , due to the inherent restriction of planar routing, \bar{S} cannot be very dense. Therefore, in many cases, we have $d(\bar{S}) \leq h$, so that the solution S we construct is optimal. In other cases, $d(\bar{S})$ usually exceeds h by just a small constant so that the solution we construct is close to optimal since $w(S)/w(S^*) \geq h/d(\bar{S})$. For example, for the 13 rows of cells in the Primary1 circuit [18], the maximum of $d(\bar{S})$ is 13, and the average of $d(\bar{S})$ is 8.5. However, for the cell family used at Xerox PARC, the number of available tracks over the cells is 13 (i.e., $h = 13$). Thus for all the rows in the Primary1 circuit, the solutions to the TFPR problems constructed by our algorithm are optimal.

Now we will discuss the two steps in our algorithm for the TFPR problem in detail. In the first step, we need to compute an optimal solution \bar{S} to the corresponding TUPR problem. \bar{S} can be computed as follows. We find two maximum-weighted one-sided planar routing solutions, S^l and S^u , for the lower and upper terminals of R independently, regardless of the availability of routing tracks. (A one-sided routing solution has the property that all the connections are on one side of a row of terminals. In our case, S^l is always above the lower terminals and S^u is always below the upper terminals). Then, the union of S^l and S^u is an optimal solution to the TUPR problem. This is the case because in the TUPR problem we do not have to connect any lower terminal with any upper terminals; nor do we have to consider track sharing of the routing solutions for the two rows since we have an infinite number of tracks. In order to compute S^l or S^u , a maximum-weighted one-sided planar routing solution must be found for a row of terminals. This problem is solved optimally using a weighted version of the algorithm for the MSOP problem in [5]. Based on the results in [5], we have:

Theorem 3: For a TFPR problem, the corresponding TUPR problem can be solved optimally in $O(k^2n^2)$ time, where n is the number of terminals and k is the maximum size of a net. Thus, the first step of the approximation algorithm to the TFPR problem can be solved in $O(k^2n^2)$ time.

In the second step, which is performed only when $d(\bar{S}) > h$, a subset S of connected pairs are chosen from \bar{S} such that $d(S)$ is no more than h and $w(S)$ is maximum. This problem can be solved by finding a maximum-weighted h -family in a partially ordered set. A *partially ordered set* P is a collection of elements together with a binary relation \leftarrow defined on $P \times P$ which satisfies the following conditions [16]:

- 1) *reflexive*, i.e., $x \leftarrow x$ for all $x \in P$;
- 2) *antisymmetric*, i.e., $x \leftarrow y$ and $y \leftarrow x$ implies $x = y$;
- 3) *transitive*, i.e., $x \leftarrow y$ and $y \leftarrow z$ implies $x \leftarrow z$.

We say that x and y are *related* if $x \leftarrow y$ or $y \leftarrow x$. A *chain* in P is a subset of elements such that every two of them are related. An *antichain* in P is a subset of elements such that no two are related. An h -family in P is a subset of elements such that it contains no chain of size $h + 1$ [13]. We can have an integer weight $w(p)$ associated with each element p in P . The weight of a subset Q of elements in P , denoted $w(Q)$, is defined to be the sum of the weights of the elements in Q . A *maximum-weighted h -family* in P is an h -family whose weight is maximum. For each connected pair A in \bar{S} , it defines an interval $i(A) = [x, y]$ where x and y ($x \leq y$) are the two column indexes of the two terminals in A . A partially ordered set $P(\bar{S})$ is constructed for the planar routing solution \bar{S} computed in the first step as follows. Each element in $P(\bar{S})$ represents a connected pair in \bar{S} . We say that a connected pair A_1 *dominates* a connected pair A_2 in $P(\bar{S})$ (or A_2 is *dominated* by A_1) if one of the following three conditions holds:

- 1) Both A_1 and A_2 are in the lower row, and $i(A_1)$ contains $i(A_2)$;
- 2) Both A_1 and A_2 are in the upper row, and $i(A_2)$ contains $i(A_1)$;
- 3) A_1 is in the upper row and A_2 is in the lower row, and $i(A_1)$ intersects $i(A_2)$.

Let the dominance relation be the binary relation in $P(\bar{S})$. Then, we can show that:

Lemma 1: $P(\bar{S})$ thus constructed is a partially ordered set.

Since it is straightforward to verify that the dominance relation thus defined is reflexive, antisymmetric and transitive, we leave the reader to complete the proof of Lemma 1. Intuitively, A_1 dominates A_2 if and only if the connection of A_1 must be above the connection of A_2 . Fig. 11 shows the partially ordered set of the solution shown in Fig. 10. The reason that we introduce the notion of a partially ordered set is clear from the following result:

Lemma 2: A subset of S of connected pairs from \bar{S} satisfies the condition $d(S) \leq h$ if and only if S is an h -family of $P(\bar{S})$.

Proof: If $d(S) \leq h$, the connections of the pairs in S can be routed in at most h tracks (since there is no vertical constraint in this case). It is easy to verify that if the connections of two pairs share the same track, then these two pairs are not related in $P(\bar{S})$ under the dominance relation. Therefore, S can be partitioned into h antichains. So, S is an h -family of $P(\bar{S})$.

On the other hand, if S is an h -family of $P(\bar{S})$, S can be partitioned into at most h antichains by recursively peeling off the maximal elements in S . It is easy to see that the density of the connections of the pairs in an antichain is one. Thus the density of the connections of the pairs in S is no more than h . \square

According to Lemma 2, we see that the problem of finding a maximum-weighted subset of connected pairs s from \bar{S} such that $d(S) \leq h$ is equivalent to the problem of finding a maximum-weighted h -family in $P(\bar{S})$. According to the results to be presented in Section IV, a maximum-weighted h -family in a partially ordered set can be computed in $O(h \cdot mn \log n^2/m)$ time, where n is the number of the elements in the partially ordered set, and m is the number of related pairs in the partially ordered set. Thus S can be computed efficiently. Moreover, we shall show that S is a good approximation of the optimal solution S^* to the TFPR problem. The following theorem states these results.

Theorem 4: If the routing solution \bar{S} computed in the first step is too dense (i.e., $d(\bar{S}) > h$), we can choose a maximum weighted subset of connected pairs S from \bar{S} in $O(h \cdot mn \log n^2/m)$ time

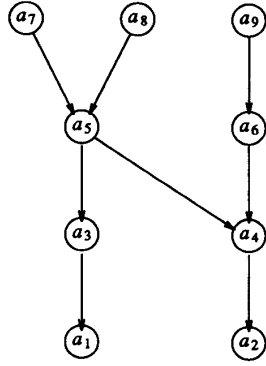


Fig. 11. The partially ordered set $P(\bar{S})$ of the solution in Fig. 10.

such that $d(S) \leq h$ and

$$\frac{w(S)}{w(S^*)} \geq h/d(\bar{S}).$$

Proof: According to Lemma 2, we can obtain S by computing a maximum weighted h -family in $P(\bar{S})$, which can be carried in $O(h \cdot mn \log n^2/m)$ based on the algorithm to be presented in Section IV. Now let us show that the weight of S satisfies the inequality stated above.

Let g denote $d(\bar{S})$. According to Lemma 2, \bar{S} is a g -family in $P(\bar{S})$. Thus \bar{S} can be decomposed into g antichains. Let u_1, u_2, \dots, u_g denote the weights of the g antichains sorted in non-increasing order (the weight of an antichain in $P(\bar{S})$ is defined to be the sum of the weights of the elements in the antichain). Since S is a maximum weighted h -family in $P(\bar{S})$, we have

$$w(S) \geq u_1 + u_2 + \dots + u_h \geq h \cdot u_h.$$

Moreover,

$$\begin{aligned} w(\bar{S}) &= u_1 + u_2 + \dots + u_g \leq w(S) + u_{h+1} + \dots + u_g \\ &\leq w(S) + (g - h) \cdot u_h \leq w(S) + \frac{g - h}{h} w(S) \\ &\leq \frac{g}{h} w(S) = \frac{d(\bar{S})}{h} w(S). \end{aligned}$$

Clearly, we have $w(S^*) \leq w(\bar{S})$. Therefore,

$$\frac{w(S)}{w(S^*)} \geq h/d(\bar{S}). \quad \square$$

According to this theorem, the time complexity of the second step (remember that it is carried out only when $d(\bar{S}) \geq h$) of our algorithm is $O(h \cdot mn \log n^2/m)$. Since the time complexity of the first step is $O(k^2 \cdot n^2)$, the overall complexity of our algorithm for the TFPR problem is $O(k^2 \cdot n^2 + h \cdot mn \log n^2/m)$, where n is the number of terminals, k is the maximum size of a net, and h the number of available tracks. Since k is a constant for most circuits, we have

Corollary 3: When the maximum size of a net is bounded by a constant, our approximation algorithm for the TFPR problem produces a solution in S in $O(n^2 + h \cdot mn \log n^2/m)$ time such that

$$\frac{w(S)}{w(S^*)} \geq \min(1, h/d(\bar{S}))$$

where S^* is an optimal solution to the TFPR problem.

Based on these results, our over-the-cell routing algorithm for the HCVC model works as follows. For each row of cells, the lower and upper terminals of each row of cells are routed at the same time. First, we compute an optimal solution \bar{S} of the corresponding TUPR problem. If the density of \bar{S} is no more than the number of tracks available, we output \bar{S} . Otherwise, we construct the partially ordered set $P(\bar{S})$ associated with \bar{S} and compute a maximum-weighted h -family.

3.5. The Algorithm for the HDVC Model

In the HDVC model, for each row R of cells, the over-the-cell routing region is divided vertically by the feedthroughs into many sub-regions. Clearly, a terminal in one sub-region cannot be connected to a terminal in another sub-region. Therefore, over-the-cell routing is carried out independently for each sub-region. Since there is no horizontal partition of the sub-regions, each sub-region can be shared by the lower and upper terminals in the given sub-region. Also, the number of tracks in each sub-region is limited by the height of cells. Therefore, the routing problem for each sub-region is exactly the TFPR problem formulated in the preceding sub-section. Thus, the algorithm for the TFPR problem described in the preceding sub-section can be applied to each sub-region. Let f_1, f_2, \dots, f_l be the feedthroughs in the row R . Let n_i be the number of terminals between f_i and f_{i+1} , and n be the total number of terminals in the row R . Clearly, $\sum_{i=1}^{l-1} n_i \leq n$. According to Corollary 3, the time to route in the sub-region between f_i and f_{i+1} is $O(h \cdot n_i m_i \log n_i^2/m_i) = O(h \cdot n_i^3)$, the total routing time for the row R is

$$O\left(\sum_{i=1}^{l-1} h \cdot n_i^3\right) = O\left(h \cdot \left(\sum_{i=1}^{l-1} n_i\right)^3\right) = O(h \cdot n^3).$$

IV. COMPUTING A MAXIMUM WEIGHTED h -FAMILY IN A PARTIALLY ORDERED SET

In this section, we present an $O(h \cdot mn \log n^2/m)$ time algorithm for computing a maximum-weighted h -family in a partially ordered set, where n is the number of elements in the partially ordered set and m is the number of related pairs in the partially ordered set. Clearly, $m = O(n^2)$. Note that the time complexity of our algorithm is independent of the magnitude of the weights of the elements. Therefore, our algorithm is a strong polynomial time algorithm. We proved that the problem of computing a maximum-weighted h -family in a partially ordered set can be reduced to the one of computing a maximum flow in a network with bounded unit flow cost. Our algorithm is based on Ford and Fulkerson's algorithm for computing minimum cost flows. We shall not include the proof of correctness of our algorithm and the analysis of time complexity of our algorithm, since the proof and analysis are rather lengthy and complicated and they might not be of general interest of the researchers in CAD area. Those results will be presented separately in a forthcoming paper [4]. Nevertheless, we shall describe our algorithm in detail so that it can be implemented by other researchers in the field in a straightforward way. (For basic concepts and terminologies in network flow, see [22].)

Let P be a partially ordered set with positive weights. Let p_1, p_2, \dots, p_n be the elements in P and \leftarrow be the partial ordering relation. Let w_i denote the weight of p_i . First, we construct the *split graph* $G(P) = (V, E)$ associated with P as follows: For each element p_i in P , we introduce two vertices x_i and y_i in V . We introduce an directed edge (x_i, y_i) in E if $p_j \leftarrow p_i$. Moreover, we introduce two more vertices s (source) and t (sink) in V and add edges (s, x_i) and (y_i, t) for $1 \leq i \leq n$ in E . Fig. 12 shows an example of a partially

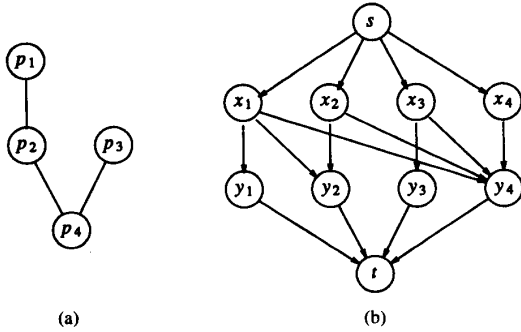


Fig. 12. (a) A partially ordered set P . (b) Its split graph $G(P)$.

ordered set and its corresponding split graph. Now we define the capacity of each edge (x, y) , denoted $c(x, y)$, as follows:

$$\begin{aligned} c(s, x_i) &= c(y_i, t) = w_i, & \text{for } 1 \leq i \leq n \\ c(x_i, y_j) &= \infty, & \text{for } 1 \leq i, j \leq n. \end{aligned}$$

Furthermore, we define the cost of each edge (x, y) , denoted $a(x, y)$, as follows:

$$\begin{aligned} a(s, x_i) &= a(y_i, t) = 0, & \text{for } 1 \leq i \leq n \\ a(x_i, y_j) &= 0, & \text{for } 1 \leq i \neq j \leq n \\ a(x_i, y_i) &= 1, & \text{for } 1 \leq i \leq n. \end{aligned}$$

So far, we have constructed a network $G(P)$ for the given partially ordered set P . Each edge in the network has a nonnegative capacity and a nonnegative cost associated with it.

Now let us consider a flow f from s to t in $G(P)$. (For convenience, any flow referred in the rest of this section is a flow from s to t in $G(P)$.) We use $f(x, y)$ to denote the value of the flow on edge (x, y) . Clearly, f has to satisfy the flow conservation property at any intermediate vertex x , i.e.,

$$\sum_{(x,y) \in E} f(x, y) = \sum_{(y,x) \in E} f(y, x), \quad (\text{for } x \neq s, t).$$

Let $v(f)$ denote the total value of flow f from s to t , i.e.,

$$v(f) = \sum_{(s,x) \in E} f(s, x) = \sum_{(y,t) \in E} f(y, t).$$

For a positive integer h , we define the h -bounded gain of flow f to be

$$h \cdot v(f) - \sum_{(x,y) \in E} a(x, y) \cdot f(x, y).$$

We are interested in finding a flow f with the maximum h -bounded gain. We shall show later that a flow with the maximum h -bounded gain in $G(P)$ leads to a maximum-weighted h -family in P .

In fact, the problem of computing a flow with the maximum h -bounded gain can be formulated as the following linear programming problem:

$$\sum_{(s,x) \in E} f(s, x) - v(f) \leq 0 \quad (4.1)$$

$$\sum_{(x,y) \in E} f(x, y) - \sum_{(y,x) \in E} f(y, x) \leq 0, \quad x \in V; x \neq s, t \quad (4.2)$$

$$v(f) - \sum_{(y,t) \in E} f(y, t) \leq 0 \quad (4.3)$$

$$0 \leq f(x, y) \leq c(x, y), \quad (x, y) \in E \quad (4.4)$$

$$\max h \cdot v(f) - \sum_{(x,y) \in E} a(x, y) \cdot f(x, y). \quad (4.5)$$

Note that although we use inequalities in (4.1)–(4.3), it is easy to see that the equalities are implied for these equations. (These equations enforce the flow conservation property.) Now let us consider the dual problem of this linear programming problem. For each flow conservation equation stated in (4.1)–(4.3), we introduce a dual variable $\pi(x)$. For each capacity constraint stated in (4.4), we introduce a dual variable $\gamma(x, y)$. Then, it is easy to derive that the dual problem has the following formulation (see [11, p. 114]):

$$-\pi(s) + \pi(t) = h \quad (4.6)$$

$$\pi(x) - \pi(y) + \gamma(x, y) \geq -a(x, y), \quad (x, y) \in E \quad (4.7)$$

$$\pi(x) \geq 0, \quad x \in V \quad (4.8)$$

$$\gamma(x, y) \geq 0, \quad (x, y) \in E \quad (4.9)$$

$$\min \sum_{(x,y) \in P(E)} c(x, y) \gamma(x, y). \quad (4.10)$$

Each dual variable of the form $\pi(x)$ is called the *node potential* of vertex x . And each dual variable of the form $\gamma(x, y)$ is called the *edge slack* of edge (x, y) . An important result which we showed in [4] is that the node potentials in an optimal solution to (4.6)–(4.10) give us a maximum weighted h -family in P . In fact, we have the following theorem.

Theorem 5: Suppose that $\{\hat{\pi}(x) | x \in V\}$ is the set of node potentials in an optimal solution to (4.6)–(4.10). Then, we have:

- 1) $0 \leq h - \hat{\pi}(x_i) + \hat{\pi}(s) \leq h$ for $1 \leq i \leq n$;
- 2) $A_k = \{p_i | h - \hat{\pi}(x_i) + \hat{\pi}(s) = k \text{ and } \hat{\pi}(y_i) - \hat{\pi}(x_i) = 1\}$ is an antichain in P for each $0 \leq k \leq h$;
- 3) $A_1 \cup A_2 \cup \dots \cup A_h$ is a maximum h -family in P .

The proof of Theorem 5 can be found in [4]. According to this theorem, in order to compute a maximum weighted h -family in a partially ordered set P , we simply need to solve the dual problem defined in (4.6)–(4.10) of computing a flow with maximum h -bounded gain in $G(P)$. Since such a linear programming problem can be solved by Ford and Fulkerson's algorithm for computing a minimum cost flow in $O(h \cdot mn \log n^2/m)$ time (we shall describe this later in more detail), we conclude that

Theorem 6: For a weighted partially ordered set of n elements, the problem of computing a maximum weighted h -family can be solved in $O(h \cdot mn \log n^2/m)$ time, where m is the number of related pairs in the partially ordered set.

According to these results, our algorithm for computing a maximum-weighted h -family can be described in Fig. 13.

Now we shall present the algorithm due to Ford and Fulkerson for computing a flow with maximum h -bounded gain. They used this algorithm for computing a minimum-cost maximum flow (since one can show that when h is large enough, a flow with the maximum h -bounded gain is a minimum-cost maximum flow). In fact, their algorithm solves both the primal problem defined in (4.1)–(4.5) and the dual problem defined in (4.6)–(4.10) because their algorithm is essentially a primal-dual algorithm. Their algorithm can be described as follows:

Given a direct graph $G = (V, E)$ in which each edge has a nonnegative capacity and a nonnegative cost, Ford and Fulkerson's algorithm starts with the initial solution $f(x, y) = 0$ (for all $(x, y) \in E$) and $\pi(x) = 0$ (for all $x \in V$). In order to compute a flow with the maximum h -bounded gain, the algorithm goes through h iterations. During each iteration, first, we construct the admissible graph $H = (V, E')$ of G . The admissible graph H has the same vertex set as G . The edge set E' in H is defined as follows: An edge

Algorithm Computing_Max_Weighted_Family.**Input** A weighted partially ordered set P and an integer h .**Output** A maximum weighted h -family of P .

1. Construct the split graph $G(P) = (V, E)$ and assign the capacity and cost for each edge in E ;
2. Compute node potentials $\pi(x)$'s in an optimal solution to the dual problem defined in (4.6)-(4.10) of computing a flow with the maximum h -bounded gain.
3. For each k , let $A_k = \{p_i | h - \pi(x_i) + \pi(s) = k \text{ and } \pi(y_i) - \pi(x_i) = 1\}$;
4. Output $A_1 \cup A_2 \cup \dots \cup A_h$

Fig. 13. Algorithm for computing a maximum-weighted h -family in a partially ordered set. (x, y) belongs to E' if and only if

- i) $a(x, y) + \pi(x) - \pi(y) = 0$ and $f(x, y) < c(x, y)$ in G ; or
- ii) $a(x, y) + \pi(x) - \pi(y) = 0$ and $f(y, x) > 0$ in G .

In case i), we define the capacity of edge (x, y) in H to be $c(x, y) - f(x, y)$. In case ii), we define the capacity of edge (x, y) in H to be $f(y, x)$. Next, we compute a maximum flow f' in H from s to t . Then, we augment the flow f in G by f' . Moreover, let $R_f(H)$ be the residual graph for flow f' in H . We increase the node potential $\pi(x)$ by one if x is not reachable from s in $R_f(H)$. (Note that at least t is not reachable from s in $R_f(H)$ since f' is a maximum flow in H .) The updated f and π are used in the construction of the admissible graph in the next iteration. At the end of h th iteration, we can show that f is a flow with the maximum h -bounded gain and π is the node potential function in an optimal solution to the dual problem defined in (4.6)-(4.10). (The proof of the correctness of the algorithm can be found in [11, pp. 113-127].) In summary, Ford and Fulkerson's algorithm for computing a flow with the maximum h -bounded gain can be described as in Fig. 14.

Note that during each iteration, the most time-consuming step is to compute a maximum flow in the admissible graph H , which can be carried out in $O(mn \log n^2/m)$ time using an algorithm by Goldberg and Tarjan [12]. Therefore, Ford and Fulkerson's algorithm for computing a flow with the maximum h -bounded gain has the complexity $O(h \cdot mn \log n^2/m)$.

V. EXPERIMENTAL RESULTS

We tested our algorithms on two circuits from Xerox PARC and the Physical Design Workshop benchmarks. In order to concentrate on the results produced by the algorithms, we used the same placement and global routing for comparison. Placement and global routing for these circuits were carried out by existing design tools used at Xerox PARC. The density of each channel is computed after global routing. This density is labeled "original density" in Tables II and III. Clearly, this density is a lower bound of the number of tracks used by any conventional two-layer channel router. Our over-the-cell router uses the global routing solution as input. After completing the over-the-cell routing stage, the net segment selection algorithm in [5] was used to determine the terminals to be connected in each channel. After this stage, the resulting density of each channel was computed.

The Xerox PARC standard cell family was used for testing the algorithms. The existing cell family follows the HDVC model closely and was taken as the base layout. We laid out a representative sample of the Xerox PARC cells according to the three models described in this paper. For each layout model, the driver characteristics of the output transistors and the capacitance at the input terminals were unchanged. Thus all of the cells in all three models have the same height (determined by the widths of the n- and p-transistors and the minimum n-transistor to p-transistor spacing). The relative positions of the input-output terminals were held

Algorithm Flow_with_max_h-bounded_gain:**Input** A directed graph $G = (V, E)$ with edge capacities and edge costs, and an integer h ;**Output** A flow with maximum weighted h -bounded gain and the associated node potentials:

1. For each $x \in V$, assign $\pi(x) = 0$;
For each $(x, y) \in E$, assign $f(x, y) = 0$;
2. While $\pi(t) < h$ do
 - 2.1 Construct the admissible graph $H = (V, E')$ and determine the edge capacities in H ;
 - 2.2 Find a maximum flow f' in H ;
 - 2.3 Augment f by f' in G ;
 - 2.4 For each vertex x which is not reachable from s in $R_f(H)$ assign $\pi(x) = \pi(x) + 1$;
3. Output f and π .

Fig. 14. Algorithm for computing a flow with the maximum h -bounded gain.TABLE II
REDUCTION OF CHANNEL DENSITIES IN THE REED-SOLOMON DECODER

Channel	Original Density	HCVD Density	HCVC Density	HDVC Density
16	12	8	8	11
15	7	5	5	7
14	13	11	11	13
13	10	9	9	10
12	10	8	8	10
11	11	9	9	11
10	9	7	7	9
9	10	10	10	10
8	13	10	10	13
7	10	8	8	10
6	11	8	8	10
5	7	7	7	7
4	11	9	9	11
3	9	8	8	9
2	11	9	9	11
total	154	126	126	152
reduction		17.5%	18.2%	1.3%

TABLE III
REDUCTION OF CHANNEL DENSITIES IN PRIMARY I

Channel	Original Density	HCVD Density	HCVC Density	HDVC Density
14	21	15	15	19
13	19	15	15	19
12	19	15	15	19
11	26	22	21	26
10	27	22	22	27
9	25	21	21	25
8	22	18	18	21
7	24	19	19	22
6	16	13	13	16
5	25	22	21	25
4	18	12	12	18
3	14	13	11	14
2	19	14	14	18
total	275	221	217	269
reduction		18.5%	21.1%	2.2%

constant in all three layout models. This restriction penalized the HCVC and HCVD cells to some extent because the input-output terminal positions and the internal wiring had been optimized for the original cells (following the HDVC model). We found little increase in area for the HCVD cells. However, the width of the HCVC cells increased because of the need to bring power and ground connections into the cells. The cells in the cell family at Xerox PARC are 108λ high. The center-to-center track spacing on

TABLE IV
COMPARISON WITH THE ROUTER IN [21]

Examples	Channel Length	Original Density	No. Tracks in Channel		Improvement
			The Router in [21]	Our Router	
Ex1	43	12	10	9	10.0%
Ex3a	89	15	15	12	20.0%
Ex3b	84	17	16	13	18.7%
Ex3c	103	18	16	15	6.3%
Ex4b	119	17	16	12	25.0%
Ex5	128	20	14	12	14.2%
Deutsch average	175	19	20	17	15.0%

the M2 layer is 8λ . Therefore, there are 13 tracks over the cells. In the HCVD model, since the power and ground buses run in the middle of the M2 layer over the cells and they are wider than regular tracks, there are 5 tracks available in each of the two sub-regions for over-the-cell routing of the lower or upper channels. In the HCVC and HDVC models, all of the 13 tracks are available for over-the-cell routing of both the lower and upper terminals.

The algorithms were tested on two circuits. One circuit is labeled the Reed-Solomon Decoder (RSD). It has 210 cells and 211 nets. Another circuit is labeled Primary1. It is a benchmark example used in the Physical Design Workshop [18]. It is composed of 752 cells and 904 nets.⁵ The cells in the RSD are placed in 16 rows. Table II shows reduction of channel densities in the RSD for each of the over-the-cell routing models. The cells in Primary1 are placed in 14 rows. Table III shows reduction of channel densities in Primary1 for each of the over-the-cell routing labels. For both examples, the density reduction in the HCVD and HCVC models is around 20%. However, the density reduction in the HDVC model is at most 2%. This is because the cell family at Xerox PARC does not use built-in feedthroughs. As a result, many feedthroughs are introduced in the global routing phase. Therefore, the over-the-cell routing region in the HDVC model is divided into many small sub-regions and is difficult to utilize effectively. Also note that the density reduction achieved in the HCVC model is slightly better than the reduction achieved in the HCVD model. This is as expected, since the entire M2 layer over the cells in the HCVC model is used for over-the-cell routing. However, in the HCVC model the power and ground buses must be routed for each channel, and the HCVC cells must be wider than the cells in the other models. If we include the overhead in the HCVC model, we can conclude that the HCVD model is the most area-efficient among the three over-the-cell routing models.

In the HCVD model, we compared our over-the-cell router with the over-the-cell router by Shiraishi and Sakemi [21]. Their router, called the permeation router, is based on heuristic graph coloring algorithms. The HCVD model is equivalent to their routing model without using diffused underpass.⁶ Table IV shows the comparison of the two over-the-cell routers on the benchmark channel routing examples in [23]. (Note that in the HCVD model, we do not need to know the feedthrough positions to carry out over-the-cell routing.)

⁵An interesting test case would be Deutsch's Difficult Example. However, for that channel, the feedthrough positions and the information on adjacent channels are no longer available [8].

⁶They did not specify the cell height in their paper. In some cases, they allow as many as 28 tracks to be routed in over-the-cell region. In this experiment, we set the number of available tracks to be 20 for over-the-cell routing.

ing.) On the average, our over-the-cell router uses 15.6% fewer tracks in the channels.

VI. CONCLUSIONS

In this paper, we present three physical models which enable us to utilize cell area for over-the-cell routing. We describe three algorithms for these over-the-cell routing models. Each algorithm produces provably good results in polynomial time. The saving in routing area achieved by these algorithms is up to 21%. In solving the over-the-cell routing problems, we also developed a polynomial time for computing a maximum-weighted h -family in a partially ordered set. We believe that this algorithm can be applied to the solutions of many other CAD problems.

Currently, we are interested in exploiting the possibility of combining the global routing step with the over-the-cell routing step since the decision on the positions of feedthroughs has significant impact on both the channel density and the utilization of over-the-cell routing area. Moreover, in the combined approach, some feedthroughs may be replaced by the connections in the over-the-cell routing area in HCVC and HDVC models. We are also interested in combining the cell generation with the over-the-cell routing approach together so that we have more efficient utilization of the routing area over the cells for both intracell connections and intercell connections.

Over-the-cell connections may go over the active regions in the cells and pick up more capacitance than the connections in the channels. Therefore, it is preferable to route non-critical nets in the over-the-cell routing regions. In general, as device dimension decreases and circuit speed increases, interconnection delay is becoming a more and more important factor in the design of high-performance VLSI circuits and systems. It would be important to develop detailed routers which take both area and performance optimization into consideration.

ACKNOWLEDGMENT

The authors thank Dr. David Deutsch for his valuable insights and suggestions.

REFERENCES

- [1] M. Burstein and R. Pelavin, "Hierarchical channel router," *Integration, VLSI J.*, vol. 1, pp. 21-38, 1983.
- [2] J. Cong and B. Preas, "A new algorithm for standard cell global routing," in *Proc. Int. Conf. on Computer-Aided Design*, pp. 176-179, 1988.
- [3] J. Cong and C. L. Liu, "Over-the-cell channel routing," in *Proc. Int. Conf. Computer-Aided Design*, pp. 80-83, 1988.
- [4] J. Cong, "Computing maximum weighted k -families or k -cofamilies

- in a partially ordered set," UCLA Computer Science Dept. Tech. Rep. 930014, Los Angeles, CA, 1993.
- [5] J. Cong and C. L. Liu, "Over-the-cell channel routing," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 408-418, Apr. 1990.
 - [6] J. Cong, B. Preas, and C. L. Liu, "General models and algorithms for over-the-cell routing in standard cell design," in *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 709-715, 1990.
 - [7] D. N. Deutsch, "A dogleg channel router," in *Proc. 13th Design Automation Conf.*, pp. 425-433, 1976.
 - [8] —, private communications.
 - [9] D. N. Deutsch and P. Glick, "An over-the-cell router," in *Proc. 17th IEEE/ACM Design Automation Conf.*, pp. 32-39, 1980.
 - [10] S. Even and A. Itai, "Queues, stacks, and graphs," in *Theory of Machines and Computations*, A. Kohavi, Ed. New York: Academic, 1971, pp. 71-86.
 - [11] L. R. Ford and D. R. Fulkerson, *Flow in Networks*. Princeton, NJ: Princeton Univ. Press, 1962.
 - [12] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," *Proc. 8th Annual ACM Symp. on Theory of Computation (1987)*, pp. 136-146.
 - [13] C. Greene and D. Kleitman, "The structure of Sperner k -family," *J. Combinat. Theory, Ser. A*, vol. 20, pp. 80-88, 1976.
 - [14] G. Gudmundsson and S. Ntafos, "Channel routing with superterminals," in *Proc. 25th All. Conf. on Computing, Control and Communication*, pp. 375-376, 1987.
 - [15] H. E. Krohn, "An over-the-cell gate array channel router," in *20th IEEE/ACM Design Automation Conf.*, pp. 665-670, 1983.
 - [16] C. L. Liu, *Elements of Discrete Mathematics*. New York: McGraw-Hill, 1977.
 - [17] K. F. Liao, D. T. Lee, and M. Sarrafzadeh, "Planar subset of multi-terminal nets," *Integration, VLSI J.*, vol. 10, pp. 184-190, Sept. 1990.
 - [18] B. Preas, "Benchmarks for cell-based layout systems," in *Proc. 24th ACM/IEEE Design Automation Conf.*, pp. 318-320, 1987.
 - [19] J. Reed, A. Sangiovanni-Vincentelli, and M. Santomauro, "A new symbolic channel router, YACR2," *IEEE Trans. Computer Aided Design*, vol. CAD-4, pp. 208-219, 1985.
 - [20] R. L. Rivest and C. M. Fiduccia, "A 'greedy' channel router," in *Proc. 19th Design Automation Conf.*, pp. 418-424, 1982.
 - [21] Y. Shiraishi and Y. Sakemi, "A permeation router," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 462-471, May 1987.
 - [22] R. E. Tarjan, *Data Structures and Network Algorithms*. Soc. Industrial and Applied Mathematics, Philadelphia, PA, 1983.
 - [23] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. Computer Aided Design*, vol. CAD-1, pp. 25-35, Jan. 1982.

Divergence and Scheduling in Functional Level Concurrent Fault Simulation

Ohyoung Y. Song, Bong-Hee Park, and P. R. Menon

Abstract—Concurrent fault simulation is widely used because of its flexibility and applicability at different levels of modeling. Two important components of the concurrent simulation algorithm are diver-

Manuscript received January 23, 1990; revised August 19, 1991. This work was supported in part by a grant from Analog Devices, Inc., Norwood, MA. This paper was recommended by Associate Editor R. Bryant.

O. Y. Song was with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. He is now with IBM Corp., Endicott, NY 13860.

B.-H. Park was with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. He is now with Intel Corp., Hillsboro, OR 97124.

P. R. Menon is with the Department of Electrical and Computer Engineering, University of Massachusetts at Amherst, Amherst, MA 01003.

IEEE Log Number 9107745.

gence and scheduling. This note examines the effects of timing specifications in functional modules on divergence and scheduling, and proposes an efficient method for performing these operations.

I. INTRODUCTION

Concurrent fault simulation [1] is the most widely used method of fault simulation. This method is preferred over other methods like parallel [2] and deductive [3] mainly because of its flexibility. Although both parallel and deductive simulation may be used at the behavioral level, the transformations needed for performing the simulation limit their applicability [4], [5]. Detailed timing analysis is also possible with these methods, but is rather complicated [5], [6]. Since fault-free modules and faulty modules are treated explicitly and independently in concurrent fault simulation, arbitrary behavioral descriptions of modules, different sets of logic values and signal strengths, and detailed timing can be handled with relative ease.

We shall assume that the reader is familiar with the basic concurrent fault simulation algorithm. For the sake of completeness, we shall define the terms used in the rest of the paper.

The circuit being simulated is assumed to be an interconnection of elements, where each element may be a gate, or a module described behaviorally. The set of signal values associated with the inputs, state variables, and outputs of an element in the fault-free circuit is called its *fault-free state*. The corresponding set of values in the presence of a fault is called the *fault state* of the fault. We shall denote the fault-free state of an element E by E_0 and its fault state with the fault α by E_α . Associated with each element in the circuit is a *fault state list*, consisting of its fault-free state and all fault states that differ from the fault-free state.

When any input or internal state value in the fault-free state or fault state of an element changes, the element is *evaluated*, to determine the next state and/or output(s). If the evaluation results in a change in an output or state-variable value, an *event* is said to be produced, and is *scheduled* for the time determined from the delay specified in the description of the element. An event is processed at the appropriate time by updating the signal value, and propagating changes to the fan-outs of the line, if the update is to an output of an element. This is called *fan-out processing*. If the update is to a state variable, the element must be evaluated to determine the effects of the state change. If the inputs, state variables, and outputs in a fault state of an element have the same values as in the fault-free state, *convergence* is said to have occurred, and the fault state is deleted. If any output in the fault state of an element becomes different from its fault-free value, and the fault-state list of a fan-out of the element does not contain the fault state for the particular fault, a new fault state must be created and added to the fault-state list of the fan-out element. This process is called *divergence*.

In this short paper, we consider the divergence and scheduling when detailed timing is specified for individual elements and the accurate timing behavior of the fault-free and faulty circuits must be simulated. The case where the inputs of an element may change before the response to the previous input combination has been completed is of particular interest, because it may occur in behaviorally modeled elements, especially those with internal pipelining. We first present a basic method of divergence that accurately simulates the timing behavior of the fault-free and faulty circuits, but may require more events to be scheduled than in gate-level simulation. We then present a new method that is more efficient than the basic method when elements may contain internal pipelining.