Automation and Remote Control, Vol. 64, No. 12, 2003, pp. 1988–1999. Translated from Avtomatika i Telemekhanika, No. 12, 2003, pp. 177–190.

COMPUTER-AIDED DESIGN AND PROGRAMMING

Concurrent Placement and Routing in the Design of Integrated Circuits[1]

A. I. Erzin* **and J.D. Cho****

**Sobolev Institute of Mathematics, Siberian Branch, Russian Academy of Sciences, Novosibirsk, Russia*
***Sungkyunkwan University, Suwon, South Korea*

Received October 28, 2002

Abstract—For the problem arising in the design of integrated chips, an efficient heuristic approach was proposed. It unites the stages of placing the logical elements (devices) on the chip and performing their detailed routing. At that, it minimizes both the critical (maximum) delay and the chip area required for routing.

1. INTRODUCTION

Placement and routing (interconnection of the operators) is pivotal to designing chips with computational (logical) operator elements (devices). Not so long ago it was assumed in the design of logical networks that the circuit delay (time of computation) depends only on the time for executing operations in the logical elements. Since the advent of circuits with elements essentially smaller than one micron, the delay of data transmission between the elements became the major component of the total delay. As was stressed in [1], for the elements not exceeding 0.25 μm, the data transmission delay can be up to 80% of the total time of computations. Works [2–4] paid more attention to the delays caused by data transmission through a logical network. For example, a new procedure enabling design of circuits that can be conveniently placed on chip by planning interconnections concurrently with design of a logical network and with due regard for the interconnections between the elements was proposed in [2]. An algorithm of placement and local rearrangement of a logical circuit calculating Boolean function was proposed in [4]. It first determines the set of the so-called super-elements comprising a subset of logical elements and lying on the critical (maximum delay) paths and then generates for each super-element a set of the non-worst transformed designs. A new method of floorplanning and placement of the elements within floors was proposed in [3].

Approaches to combining the stages of placement and routing were described in a number of publications [5–12]. For example, [5] presents an algorithm to construct the optimal logical tree with determination of the outputs of elements for a set of ordered critical terminals. The algorithm makes use of dynamic programming and is applicable to a special topology of the so-called LT-trees. The authors of [8] extend the procedure of concurrent placement and global routing of the algorithm for transport-processing FPGAs [7] and propose a new algorithm with constrained length of each critical path. The algorithm is based on hierarchical decomposition of the chip area and application of the LookUp Table procedure for element placement.

The present paper considers delays both in the circuit elements and in the wires for data transmission between the elements. An efficient polynomial algorithm for concurrent placement of the elements and their interconnection according to the given circuit is proposed for the first time.

[1] This work was supported in part by the grants of KISTEP'99 and the Russian Foundation for Basic Research, project no. 02-01-00977.

Special attention is paid to the detailed routing for interconnecting the network elements by admissible paths. At that, the edge-disjoint shortest paths passing along the edges of a rectangular grid are admitted.

2. FORMULATION OF THE PROBLEM

Let us assume that each element of the network has a single output, that is, the result of its calculations is sent only to one circuit element. It is, therefore, easy to see that the logical network without hanging vertices is a tree whose vertices are the elements performing calculations and the root vertex (main output) yields the result. At that, the hanging vertices of the tree are the main chip inputs receiving the variables of the calculated function. Then, the problem can be formulated as follows. Given are the set of elements (operators) having one or more inputs and one output each and also the places of the main inputs on the chip boundary (rectangular area, as a rule), main output, and the corresponding logical network realizing the function (see Fig. 1). Additionally, each main input stores the time of data arrival. The data are as a rule the result of operation of other units and need not be concurrently delivered to all the main chip inputs. The problem lies in placing the operators and performing routing (according to the logical network) with the use of both chip surfaces, which means that one side is used by the horizontal lines (wires) and the other, by the vertical lines. This scheme is called the two-sided routing.

Let a chip wafer be given having a rectangular grid with unit distance between two parallel neighbor lines. On its side boundary positions of the main inputs, the set I, and one output 0 are fixed. We assume that each grid edge (interval between the neighbor nodes of the grid) has a unit delay (or length). The logical function represented as a logical network defines the topology of interconnections. Stated differently, for each operator o_i including main output 0, the set of elements V_i sending data to it is known (we call them the successors of the vertex i). Let us assume that any internal vertex of the chip (operator or element) has a single output to another element (called the predecessor) or to the main output. Consequently, the internal network is a tree with the root at the common output 0. Additionally, an initial delay—an integer $d_i \geq 0$ (the instant of data arrival)—is defined for each main input $i \in I$. Also, let the critical delay D be given. This means that the total time of calculations (delay) along the path leading from each main input to

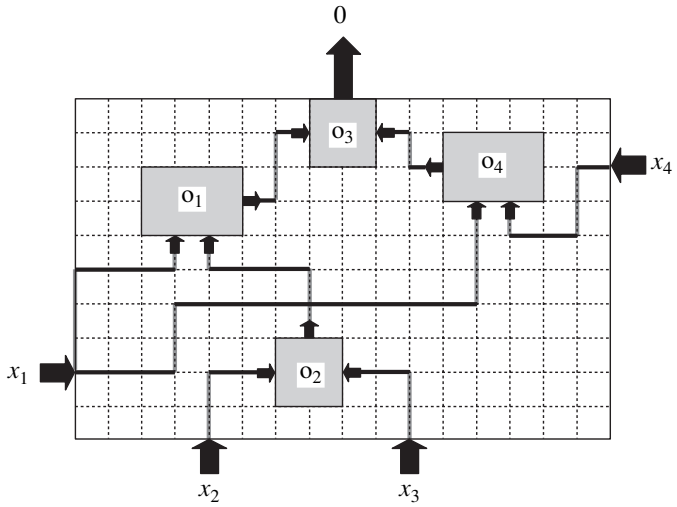

Fig. 1. Example of placement and routing for the formula $f = [x_1 o_1 (x_2 o_2 x_3)] o_3 [x_1 o_4 x_4]$, where x_i is the main input i; o_k is the operator k; and 0 is the main output. We assume that the vertical lines are on the other side of the chip.

the main output—which is the sum of the initial delay at the main input, the delays of transmission along the grid edges in the path, and the time of operator work—should not exceed D.

The problem under study lies in determining the optimal placement of rectangular elements (with the possibility of rotating them by $\pm 90^\circ$, 180°) and constructing edge-disjoint routes between the elements and their successors (according to the logical network and using the grid edges) where delays do not exceed the given values. The minimum of the occupied chip area and critical delay is used here as a criterion. The present paper proposes a heuristic approach uniting dynamic programming with detailed routing and constructs solution in a polynomial time.

3. METHOD

A logical network is given as the set V_i of immediate successors of the operator $o_i \in O$ and the main output 0. Any its vertex can be related to one of the levels $\ell \in [0, L]$ depending on the number of edges in the longest path from this vertex to the main input. For example, for the representation of the function

$$f = x_1(x_2 + x_3) + x_1x_4 = [x_1o_1(x_2o_2x_3)]o_3[x_1o_4x_4]$$

the main inputs x_i, $i = 1, 2, 3, 4$, constitute the zero level, the operators o_2, o_4 belong to the first level, the operator o_1, to the second level, the operator o_3, to the third level, and the main output 0, to the fourth level of the logical network (see Fig. 2).

Since the critical delay D of the circuit is known, the length of each path from the main input to the main output is limited, and one can determine the admissible region (area) for placement of each operator [2]. Let us assume that for each operator $o_i \in O$ these regions R_i are known and decompose the chip area into neighborhoods where the elements can be placed.

As was mentioned before, the concepts of dynamic programming will be used to find a solution. This method consists of two stages, forward and backward recursions. During the forward recursion, routing (in order to attach the successors) by the criterion for the minimum area occupied by wires is performed for each admissible placement of the element and any admissible delay from its inputs to the main output. The occupied area is calculated as the total number of grid edges in the routes. Backward recursion provides the values of particular delays and determines the places of operators. The routing algorithm constructs the admissible paths between the vertices and their successors.

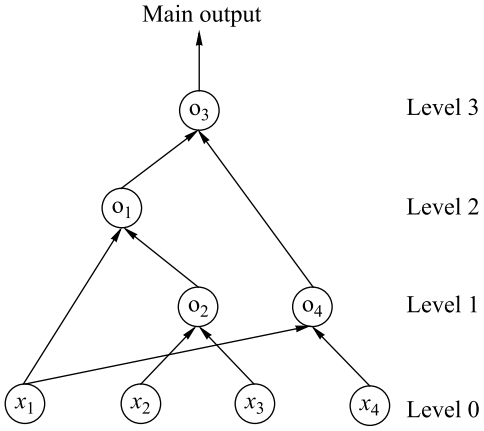

Fig. 2. Example of a logical network for the formula $f = x_1(x_2 + x_3) + x_1x_4 = = [x_1o_1(x_2o_2x_3)]o_3[x_1o_4x_4]$, where x_i is the main input i and o_k is the operator k.

3.1. Forward Recursion

This stage proceeds from level 0 of the logical network upward to the main output 0 at level L. For each internal vertex (operator) i, we introduce two cost functionals $S_i(t_i, p_i)$ and $S_{ij}(t_j, p_j)$. The first functional is the minimum total number of grid edges in the admissible paths from the successor vertices of the set V_i to the vertex i. We call it the cost of the subtree of the vertex i. Here, the variable t_i reflects the delay from the input element i to the main output 0, and p_i corresponds to the number of the place of the vertex i. The second functional is equal to the minimum of cost of the subtree $S_i(t_i, p_i)$ plus the length of the (still unknown) path from i to its predecessor vertex j. Precise value of the length of path from i to j will be known after routing. Here, t_j and p_j are the delay and place of the operator j, respectively. The aforementioned place p_i of the element i implies determination of the grid nodes containing the inputs and the output of the element. Stated differently, if the place of an operator is known (for example, p_i has a corresponding integer value, the number of the place), then this means that the grid nodes containing its inputs and output are known.

Location of each main input $i \in I$ is fixed, and the cost of the corresponding subtree is $S_i(t_i, p_i) = 0$ if $t_i + d_i \le D$ and $S_i(t_i, p_i) = +\infty$, otherwise. The value of the second cost functional for the main inputs will be determined after routing. It will be $S_{ij}(t_j, p_j) = S_i(t_j + \tau_{ij}, p_i) + \tau_{ij}$, where j is the predecessor vertex for i and τ_{ij} is the length of the corresponding route r_{ij} from i to j. Thus, the cost of a subtree is as follows:

$$S_j(t_j, p_j) = \min_{\tau_{ij}} \sum_{i \in V_j} S_{ij}(t_j, p_j).$$

The routing algorithm minimizing this expression will be described in Section 3.3.

Let j be an arbitrary operator of the level ℓ, $1 \le \ell \le L-1$, and the vertex k be its predecessor. The cost of the subtree $S_j(t_j, p_j)$ has been calculated for all admissible values of arguments. We assume that the functional

$$S_{jk}(t_k, p_k) = \min_{\tau_{jk}, p_j} \{\tau_{jk} + S_j(t_k + \tau_{jk} + \tau_j, p_j)\},$$

where τ_j is the time of work of the operator j. Therefore, this expression defines the place of the operator j and the delay (but not the route itself) from its output to the input of the element k and depends on the position of the vertex k. Since the best position of the successors of the element k and the delays to them are known, it is possible to carry out routing in order to calculate the functional

$$S_k(t_k, p_k) = \min_{r_{jk}} \sum_{j \in V_k} S_{jk}(t_k, p_k).$$

After determining the admissible paths, some of the previously determined delays τ_{jk} can be increased (this problem will be discussed in detail in Section 3.3). We assume that

$$S_{jk}(t_k, p_k) = S_j(t_k + \tau_{jk} + \tau_j, p_j) + \tau_{jk},$$

where τ_{jk} is now the length of the corresponding constructed path r_{jk}. Let us calculate the subtree cost

$$S_k(t_k, p_k) = \sum_{j \in V_k} S_{jk}(t_k, p_k).$$

Finally, we get $S_0(0, p_0)$, as well as the values of τ_{i0} and p_i, which provides a certain delay and a particular position of each element $i \in V_0$ of level $(L-1)$.

3.2. Backward Recursion

At this stage, placement and routing are carried out for each set V_i from the upper level and to the lower-level vertices, the main network inputs. Since at the end of the forward recursion the places of all vertices of V_0 are determined and the admissible paths from them to the main output are constructed, the real delays $t_i = \tau_{i0} + \tau_i$ at their inputs are known. Consequently, one can readily determine the places of the lower-level operators and the paths from them to the their predecessors. Indeed, let i be an arbitrary operator of the ℓth level. Positions of the inputs of its predecessor j and the delay d_j in them are also known. Values of the variables minimizing $\min\limits_{\tau_{ij}, p_i}\{\tau_{ij} + S_i(t_j + \tau_{ij} + \tau_i, p_i)\}$ define the position p_i of the element i and the delay $t_i = t_j + \tau_{ij} + \tau_i$ from its inputs to the main output 0. As soon as these parameters are established for all elements of V_j, the routing procedure constructs the admissible paths to connect the vertices from V_j with the vertex j. As the result, the actual delays will be determined for all vertices $i \in V_j$. Finally, the desired solution will be constructed by repeating the above procedure for all network vertices.

3.3. Routing Algorithm

The routing algorithm is the most important procedure in the above approach. The problem of routing is solved for each element (operator) with the aim of determining the admissible paths of the data from the successor vertices. This problem can be formulated as follows. The root vertices $0_i \in m_0 \subseteq V$ (inputs of the operator) and terminal vertices $V' \subseteq V$ (the set of the operator successors) are specified on the given grid graph $G = (V, E)$, where V is the set of vertices ($n = |V|$) and E is the set of edges. It is required to construct edge-disjoint shortest (or limited-length) paths from each terminal to an arbitrary root 0_i. The aforementioned constraints on the length (delay) are the maximum admissible lengths (the number of grid edges) d_k of the path from k. More precisely, we want to construct an admissible path from each terminal k that can intersect only the vertices of other paths. Moreover, the maximum exceedance of the path over the distance to the nearest root must be minimum and in any case needs not to exceed d_k. If t_i is the delay at an input of the vertex i, then $d_k = \max\{d \mid S_k(t_i + d + \tau_k) < +\infty\}$ for each $k \in V_i$. We assume here that $\tau_k = 0$ for all main inputs $k \in I$. In what follows, let $d = \max\limits_{k \in V'} d_k$.

The difference $g_k = d_k - \tau_{ki}$ between the upper delay boundary for the terminal $k \in V_i$ and the length of the path from this terminal to the root will be called the margin. Its value defines the increment in length of the corresponding path that does not violate the delay constraint, that is, does not increase the critical delay. For any admissible routing, obviously, $0 \le g_k \le d_k - d_{ki}$, where d_{ki} is the distance from k to the nearest root. It is clear that the best routing is that maximizing the margin for all terminals.

3.3.1. Construction of the hierarchical structure. We reduce the above routing problem to that on the $(d+1)$-level hierarchical structure (HS) constructed as follows. We introduce the set m_ℓ of vertices of each HS level $\ell \in [0, d]$ and the set of edges $V_{\ell i}$ outgoing from each vertex $i \in m_\ell$ to the vertices of the $(\ell+1)$th level. The zero (upper) level consists of the set m_0 (the root vertex 0). The first level $m_1 = \{j \in V \mid (0_i, j) \in V_{0i}, 0_i \in m_0\}$ includes all vertices adjacent to the vertex $0 \in m_0$. In the course of constructing the HS, a set of forbidden edges $F_{\ell i}$ will be assigned to each intermediate HS vertex $i \in m_\ell$, which means that any path passing though $i \in m_\ell$ to the lower levels will contain edges of the set $F_{\ell i}$, that is, the edges $F_{\ell i}$ belong to the single path from the vertices of preceding levels.

So, for any intermediate vertex $j \in m_1$ of the first level, the set $F_{1j} = \{(0_i, j) \mid (0_i, j) \in V_{0i}\}$. Let us assume that $(\ell - 1)$ of the first HS levels have been constructed. The set of vertices of the ℓth level can be defined as $m_\ell = \{j \in V \backslash \{\{k \in V' | d_k < \ell\} \cup m_0\} | (i, j) \in V_{\ell-1 i}, i \in m_{\ell-1} \backslash V'\}$, where $V_{\ell-1 i} = \{(i, j) \in E \backslash F_{\ell-1 i} | i \in m_{\ell-1}, i \notin V'\}$. We assume for each intermediate vertex $j \in m_\ell$

that

$$F_{\ell j} = \bigcap_{i \in m_{\ell-1} | (i,j) \in V_{\ell-1 i}} F_{\ell-1 i}.$$

If the vertex $j \in m_\ell$ has a single incident edge (i, j), $i \in m_{\ell-1}$, then we assume that $F_{\ell j} = F_{\ell j} \cup \{(i, j)\}$.

Understandably, a terminal can be included in the set of vertices of the ℓth level only if the upper delay boundary does not exceed the number of the ℓth level. Consequently, the terminal k can belong to m_ℓ if $\ell \in [d_{k0}, d_k]$, where d_{k0} is the distance from k to the nearest root.

Finally, the last HS level d can consist only of the terminals with the maximum delay boundaries. We note that the same vertex can be included in more than one HS level. We illustrate construction of an HS by way of the example shown in Fig. 3 having four terminals numerated from 1 to 4 and two roots 0_1 and 0_2. Let $d_1 = 3$, $d_2 = 4$, $d_3 = 4$, $d_4 = 5$. In this example, numerated are only the intermediate vertices that may be included in the desired paths. We note, for instance, that $d_2 = 4$ and the nearest root vertex of terminal 2 lies at distance 1. Therefore, terminal 2 can be situated at all HS levels from the first to the fourth. The maximum delay boundary for terminal 4 defining the number of HS levels is $d = 5$. The corresponding HS is depicted in Fig. 4. Here, the shortest paths are represented by the path of length 3 from terminal 1, path of length 1 from terminal 2,

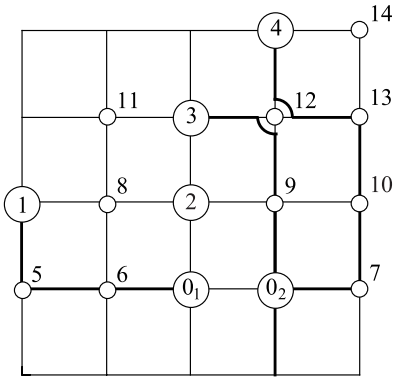

Fig. 3. Example of placement of the terminals and operators (root vertices). The edge-disjoint admissible paths are shown by bold lines.

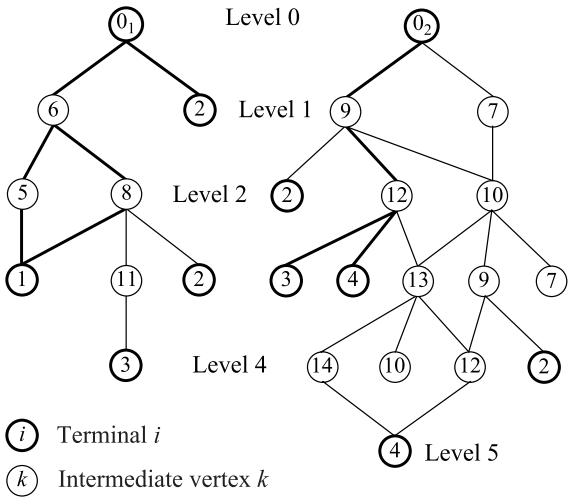

Fig. 4. Hierarchical structure and initial bundles of terminals (bold lines).

path of length 2 from terminal 3, and path of length 3 from terminal 4. As will be shown below, for this example one cannot construct the shortest edge-disjoint paths and has to extend the path from terminal 4 by two units.

3.3.2. Bundles of routes. In this section we define the notion of bundle T_k for all terminals $k \in m_{\ell_k}$. By the bundle for the terminal k is meant the set of edges $T_k = \{T_{1k}, T_{2k}, \ldots, T_{\ell_k k}\}$ in the HS that can be included in the desired path from k to the root. At that, the set $T_{\ell k} \in T_k$ consists of the edges connecting the vertices of level $(\ell - 1)$ and the vertices of level ℓ in the paths from the terminal k, and ℓ_k is the level where the terminal k is currently situated. Since the terminal can be at more than one HS level, first we determine the bundles for its "upper" position in the HS. For instance, terminal 1 in Fig. 4 has the bundle

$$T_1 = \{T_{11} = \{(0_1, 6)\}; \quad T_{21} = \{(6,5),(6,8)\}; \quad T_{31} = \{(5,1),(8,1)\}\}.$$

We note that the bundles for all terminals can be sought by one upward scan of the HS from the terminals to the roots.

As soon as the bundles for various terminals become known, they can be compared. For this purpose, we assign the parameter $p_{\ell k}(i,j) \in [0,1]$ to each edge $(i,j) \in T_{\ell k}$. It reflects the probability of including the edge (i,j) into a path from the terminal k as the ℓth—counting from the root vertex—edge. To simplify calculations, we assume that $p_{\ell k}(i,j) = 1/|T_{\ell k}|$. For the example of Fig. 4, $p_{31}(8,1) = 1/2$, $p_{24}(9,12) = 1$, and so on. When some edges will be included into solution, these parameters can change their values.

Comparison of the bundles of different terminals can cause conflict. For example, if $p_{\ell k}(i,j) > 0$ and $p_{pq}(i,j) > 0$, then the edge (i,j) can be used in alternative paths. To resolve such conflicts, heuristics is proposed in the next section.

3.3.3. Routing heuristics. Let the current position (level ℓ_k) be known for each terminal and the bundle T_k be also given for any terminal $k \in V'$. We formulate the following statements.

Lemma 1. *If for any pair of bundles T_p and T_q there exists an edge $(i,j) \in \{T_p \cap T_q\}$ with the characteristics $p_{mp}(i,j) = 1$ and $p_{\ell q}(i,j) = 1$, then there exists no edge-disjoint path from the terminals p and q. (Here and in what follows we assume that $(i,j) = (j,i)$).*

Proof. Since $p_{mp}(i,j) = 1$ and $p_{\ell q}(i,j) = 1$, the edge (i,j) is unique for connecting the levels $(m-1)$ and m for the bundle of the terminal p and the levels $(\ell - 1)$ and ℓ for the bundle of the terminal q. Therefore, if (i,j) is included in a path from p, it cannot be used in the edge-disjoint path from q. As the result, the terminal q remains disconnected from the root vertex.

Lemma 2. *If for any bundle $T_{\ell k}$ there exists an edge $(i,j) \in T_{\ell k}$ with the characteristic $p_{\ell k}(i,j) = 1$, then it should be included in the path from the terminal k as the ℓth edge.*

Proof. Since $p_{\ell k}(i,j) = 1$, the unique edge connecting the levels $(\ell - 1)$ and ℓ for the bundle of the terminal k is (i,j). For example, if (i,j) is included into another path, then it cannot be used in the edge-disjoint path from k. As the result, the terminal k remains disconnected from the root vertex.

Lemma 3. *If for any subset of the terminals $S \subseteq V'$ of cardinality $K(|S| = K)$ there exists a level ℓ such that $\left| \bigcup\limits_{k \in S} T_{\ell k} \right| < K$, then there exists no edge-disjoint path from all terminals $k \in S$.*

Proof. Since it is required to construct an edge-disjoint path for each terminal and each path has exactly one edge connecting the levels $(\ell - 1)$ and ℓ, there should be at least K different edges connecting these levels.

The routing procedure can be applied more than once until a solution is constructed. At first, we reduce the HS to a forest where each tree is rooted at one of the root vertices. Finally, in the reduced HS (RHS) there will be at least one path from each terminal. The reduction procedure starts from level 0. The vertices of this level are connected to the vertices of level 1 by the edge from T_{1k}. We arrange all positive $p_{1k}(i,j)$, $(i,j) \in T_{1k}$, $k \in V'$, in a nonascending order and denote this ordered list by R. We include in the RHS the edge (i,j) corresponding to the first element $p_{1m}(i,j)$ of R and the vertex j. Then, we eliminate from R all $p_{1k}(i,j)$, that is, assume that $R = R\backslash\{p_{1k}(i,j), k \in V'\}$. If $p_{1m}(i,j) = 1$, we assume that $p_{1k}(i,j) = 0$, $k \neq m$, and remove all other edges (i,j) (of course, also (j,i)) from the HS. We recalculate the probabilities $p_{qk}(i,j)$ for the remaining edges. If j is an intermediate vertex, we assign to it the weight $P_{kj} = p_{1k}(i,j)$, $k \in V'$, and the set of passed edges $PE_j = \{(i,j)\}$ as labels. If j is a terminal vertex, then we remove all edges (i,j) (and (j,i)) from the HS and recalculate the probabilities $p_{qk}(i,j)$, $q > 0$. In the last case, (i,j) is the desired path to the terminal j.

We take another element of R and repeat the above procedure. If at least one terminal of the first level $k \in m_1$ is disconnected from the root vertex after exhaustion of R, we stop routing. Otherwise, the procedure is repeated for other levels.

Let ℓ be an arbitrary HS level. Its vertices are connected with those of the level $\ell - 1$ by the edges from $T_{\ell k}$. We arrange all positive products $p_{\ell k}(i,j)P_{ki}$, $(i,j) \in T_{\ell k}$, $k \in V'$, in a nonascending order and again denote this ordered list by R. We take the first element of the list $p_{\ell m}(i,j)P_{mi} \in R$, $(i,j) \notin PE_i$. Let us include the edge (i,j) and the vertex j in the RHS and remove from R all $p_{\ell k}(i,j)P_{ki}$, that is, assume that $R = R\backslash\{p_{\ell k}(i,j)P_{ki}, k \in V'\}$. If $p_{\ell m}(i,j) = 1$, then we assume that $p_{1k}(i,j) = 0$, $k \neq m$, and remove all edges (i,j) and (j,i) from the HS, then recalculate the probabilities $p_{qk}(i,j)$. If j is an intermediate vertex, we label it by the weight $P_{kj} = p_{\ell k}(i,j)P_{ki}$, $k \in V'$, and the set of passed edges $PE_j = PE_i \cup \{(i,j)\}$. If j is a terminal, then we remove all edges (i,j) (and (j,i)) and all edges PE_i from the HS. We recalculate the probabilities $p_{qk}(i,j)$, $q \geq \ell$. In the last case, $PE_i \cup \{(i,j)\}$ is the desired path to the terminal j.

We take the next element from R and repeat the above procedure. If after exhausting R at least one terminal of the ℓth, $k \in m_\ell$, level it still disconnected from any vertex of the preceding level, we stop routing, Otherwise, we continue this procedure for the next levels of the HS.

The admissible paths to the terminals are as the result either constructed or not. We recall that the terminals can be at different HS levels and that now we discuss the uppermost positions (in the HS) of the terminals. In the case where an admissible path was not constructed to any terminal(s), it is possible to increase the admissible length of some paths and repeat routing. The terminal k to which the path must be increased can be chosen by minimizing the loss of margin. Since in the course of constructing the HS we admitted occurrence of the terminal k at different levels $d_{ki}, d_{ki} + 1, \ldots, d_k$, the levels to which the terminal k belongs are known. We choose among the disconnected terminals a terminal $k \in m_p$ such that it can be moved to a farther level $q > p$, $k \in m_q$, in the HS with the minimum loss of margin, that is, with the minimum difference $(q - p)$. For this terminal $k \in m_q$, we determine a new bundle T_k and compare it with the other bundles to detect conflicts (see Lemmas 1–3). If there are no conflicts, we carry out routing.

For Fig. 4, for example, the bundles for the initial position of the terminals $1 \in m_3$, $2 \in m_1$, $3 \in m_3$, and $4 \in m_3$ are as follows:

$$T_1 = \{T_{11} = \{(0_1, 6)\}; \quad T_{21} = \{(6,5),(6,8)\};$$
$$T_{31} = \{(5,1),(8,1)\}\}; \quad T_2 = \{T_{12} = \{(0_1, 2)\}\};$$
$$T_3 = \{T_{13} = \{(0_2, 9)\}; \quad T_{23} = \{(9,12)\};$$
$$T_{33} = \{(12,3)\}\}; \quad T_4 = \{T_{14} = \{(0_2, 9)\};$$
$$T_{24} = \{(9,12)\}; \quad T_{34} = \{(12,4)\}\}.$$

Hence, $p_{13}(0_2, 9) = p_{14}(0_2, 9) = 1$, and it follows from Lemma 1 that there are no edge-disjoint paths from terminals 3 and 4. Therefore, one of them must be moved to a farther level. Terminal 3 can be moved to the fourth level with loss of a unit of the margin, but this replacement causes conflict between the bundles of terminals 3 and 1. The conflict is unsolvable because the margin for terminal 1 is zero, and consequently, it cannot be moved to farther levels. Therefore, replacement of terminal 3 to the fourth level is inadmissible. Then, terminal 4 must be moved, and level 5 is the only level where it can be placed. The new bundle for terminal 4 is shown in Fig. 5 together with other bundles. Now, there are no conflicts between the bundles, and it is possible to start routing. The first ordered list for choosing the edges between the vertices of the zero and first levels is as follows:

$$R = \{p_{11}(0_1, 6) = 1;\ p_{12}(0_1, 2) = 1;\ p_{13}(0_2, 9) = 1;\ p_{14}(0_2, 9) = 1/2;\ p_{14}(0_2, 7) = 1/2\}.$$

Consequently, $(0_1, 6)$ is the first chosen edge. It is included in the RHS, and vertex 6 is labeled by $P_{16} = p_{11}(0_1, 6) = 1$ and $PE_6 = \{(0_1, 6)\}$. The next edge added to the RHS is $(0_1, 2)$. Then, $(0_2, 9)$. The labels $P_{39} = p_{13}(0_2, 9) = 1$, $P_{49} = 0$ (see Lemma 2), and $PE_9 = \{(0_2, 9)\}$ are assigned to the intermediate vertex 9. By that time, the sets $R = \{p_{14}(0_2, 7) = 1\}$ and $(0_2, 7)$ are undoubtedly included in the RHS after recalculation of the probabilities. As the result, the labels $P_{47} = p_{14}(0_2, 7) = 1$ and $PE_7 = \{(0_2, 7)\}$ are assigned to vertex 7. The underlined values of P_{ki} are shown in Fig. 5 near the vertices. For the second level, the set $R = \{p_{23}(9, 12) = 1;\ p_{24}(7, 10) = 1;\ p_{21}(6, 5) = 1/2;\ p_{21}(6, 8) = 1/2\}$. Consequently, the edge $(9, 12)$ is included in the RHS. Vertex 12 has the labels $P_{3,12} = 1$ and $PE_{12} = \{(0_2, 9), (9, 12)\}$. Moreover, the edge $(9, 12)$ belongs to T_4. Therefore, it is removed from T_4, and after recalculation $p_{34}(10, 13) = 1$.

Further construction of the RHS follows the same lines. The final solution is shown in Fig. 5 by bold lines. The corresponding paths are shown by bold lines also in Fig. 3. The dashed lines in Fig. 5 correspond to the communication lines that were eliminated from the bundles by the routing algorithm.

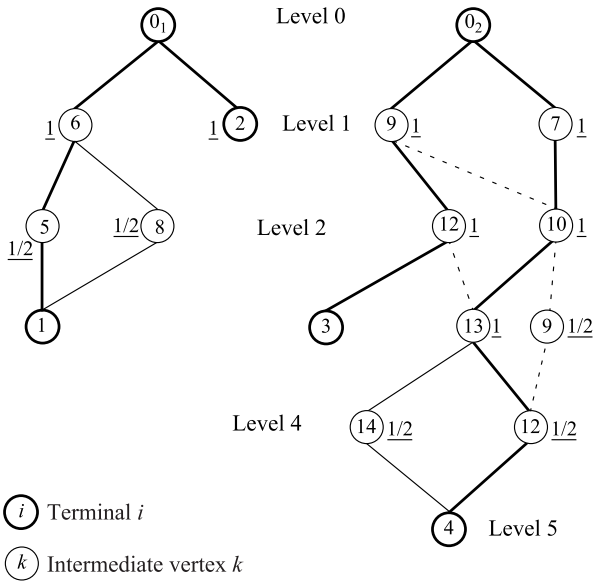

Fig. 5. Bundles after moving terminal 4 and the constructed paths (bold lines).

4. EFFICIENCY OF THE METHOD

In this section we discuss complexity of the proposed approach. Complexity of the forward recursion is, undoubtedly, defined by calculation of the functionals $S_i(t_i, p_i)$ and $S_{ij}(t_j, p_j)$, $0 \le t_i$, $t_j \le D$, $p_i \in R_i$, $p_j \in R_j$ for each vertex i, where R_i are the admissible areas for placement of the operator i. Consequently, the complexity of the forward recursion is bounded by

$$O(ND|R_i|(C_i + D|R_j|)) \le O(NDn(C_i + Dn)),$$

where N is the number of operators and C_i is the complexity of the procedure for connecting to the operator i its successors.

The routing complexity includes the complexity C_{HS} of constructing the HS and that of routing itself C_R. Since the degree of each grid vertex is bounded by 4, the complexity of constructing the HS does not exceed $O(nD^2)$. Bundles are constructed for each terminal in the course of routing. To this end, it suffices to perform one upward scanning of the HS. Hence, construction of bundles requires about $O(nD)$ elementary operations. The next step lies in calculating the probabilities $p_{\ell k}(i, j)$. This procedure has the same complexity $O(nD)$. However, in the course of routing the probabilities can be recalculated more than once. For example, after compilation of the list R and inclusion of edges in the RHS, some edges may be eliminated from the HS, and consequently, the probabilities $p_{\ell k}(i, j)$ must be recalculated. Obviously, recalculation of the probabilities $p_{\ell k}(i, j)$ has complexity $O(nD)$. Consequently, $C_R = O(n^2D^2)$ is the complexity of routing itself.

If the algorithm cannot construct admissible paths, some terminals can be moved to farther HS levels. The number of moves at the very most is $\sum\limits_{k \in V'} g_k \le \sum\limits_{k \in V'} (d_k - d_{ki}) \le Dn' \le DN$. As the result, the total complexity of the forward recursion and the entire method as a whole is bounded by $O(N^2n^3D^4)$. Obviously, the complexity of the backward recursion is smaller. By virtue of the fact that D is bounded from above n, this heuristics has polynomial complexity.

We assumed for calculation of complexity that the operator can be moved to any place on the chip. In actual fact, it is not always the case, and the areas of admissible operator placement can be much smaller. We also assumed that in the worst case the upper bounds of delays can coincide with the critical delay. This assumption also is not realistic. These remarks show promise that the actual complexity of this approach is much smaller than the upper bound.

5. CONCLUSIONS AND REMARKS

A heuristic polynomial method of concurrent placement of the elements and connections in compliance with the given logical network realizing in the integrated circuit, for example, calculation of a Boolean function was presented for the first time. As the result, the area occupied by the

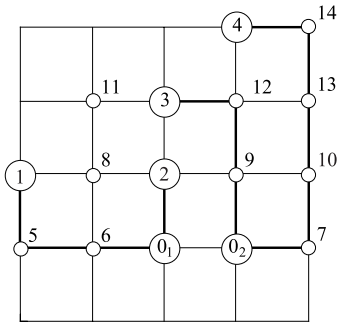

Fig. 6. Example of solution with independent paths (bold lines).

connections (wires) and the critical delay (time of calculation) are minimized. This is attained by constructing minimum-length edge-disjoint paths between the operator vertex and their successors that provide the data. If one fails to construct such paths, then the lengths of some paths are increased in a special way.

This approach solves a complex problem, but solution is not necessarily possible. Individual problems of both placement and routing are complicated by themselves, and concurrent placement and routing make the problem essentially more complicated. Therefore, no admissible solution is sometimes found by the proposed method, as well as other methods. In these cases, one might increase the critical delay. Another approach to this case lies in reducing the size of grid cells. This would provide more possibilities of routing, but increases the number of grid nodes, which complicates the algorithm.

It deserves noting that in this paper routing was based on constructing the edge-disjoint paths connecting the terminals and the root vertices. At that, it was admitted that paths intersect at vertices. Sometimes one needs to seek completely independent paths or, for bilateral routing, to permit only orthogonal intersection of two paths. In the example of Fig. 3, the paths from the terminals 3 and 4 have a common vertex 12 which is the turn point of two paths. Sometimes, such intersections may be forbidden. Constraints of this kind can be easily taken into consideration in the above scheme of routing. For example, when seeking independent paths in the routing procedure stored must be a list of the passed (included in the partially constructed path) vertices, rather than that of the passed edges.

For the case of constructing independent paths, one can readily see (Fig. 5) that terminal 4 lying at level 5 can be connected only to the intermediate vertex 14 (connection (4,12) is forbidden because vertex 12 is already included in level 2). As the result, we get solution shown in Fig. 6 where all paths are completely independent.

Routing can also take into account other requirements on path lengths within the framework of the concept of reduction to the HS problem. One of them is represented by identical (or weakly different) delays in each path [13]. This property is easily taken into account in the HS by positioning the terminals at one (or neighbor) levels.

REFERENCES

1. Keutzes, K., Newton, A.R., and Shenoy, N., The Future of Logic Synthesis and Physical Design in Deep-Submicron Process Geometries, *Proc. ISPD Conf.*, 1997, pp. 218–224.

2. Gosti, W., Narayan, A., Brayton, R.K., and Sangiovanni-Vincentelli, A.L., Wireplanning in Logic Synthesis, *Int. Conf. Comput. Aided Design*, 1998, pp. 26–33.

3. Jiang, Y. and Sapatnekar, S.S., An Integrated Algorithm for Combined Placement and Libraryless Technology Mapping, *Int. Symp. Circuits Syst.*, 1999, pp. 102–105.

4. Lou, J., Chen, W., and Pedram, M., Concurrent Logic Restructuring and Placement for Timing Closure, *Int. Symp. Circuits Syst.*, 1999, pp. 31–35.

5. Salek, A.H., Lou, J., and Pedram, M., A Simultaneous Routing Tree Construction and Fanout Optimization Algorithm, *Proc. ICCAD98, San Jose, CA, USA*, 1998, pp. 625–630.

6. Chowdhary, A. and Bhatia, D., Detailed Routing of Multi-Terminal Nets in FPGAs, *Proc. VII Int. Conf. on VLSI Design*, 1994, pp. 237–242.

7. Togawa, N., Sato, M., and Ohtsuki, T., Simultaneous Placement and Global Routing Algorithm for Transport-Processing FPGAs, *IEEE Trans. Fundamentals*, 1997, vol. E80-A, pp. 1795–1806.

8. Togawa, N., Sato, M., and Ohtsuki, T., A Simultaneous Placement and Global Routing Algorithm with Path Length Constraints for Transport-Processing FPGAs, *Proc. Design Automat. Conf.*, 1997, pp. 569–578.

9. Hur, S.W., Jagannathan, A., and Lillis, J., Timing Driven Maze Routing, *Proc. ISPD'99*, Monterey, USA, 1999, pp. 208–213.

10. Kamoshida, A. and Tsukiyama, S., A Positioning Problem of Terminals for a Parallel Router Based on Area Division, *Proc. 1997 IEEE Int. Symp. Circuits Syst. (ISCAS'97)*, 1997, vol. 3, pp. 1556–1559.

11. Alexander, M.J., Cohoon, J.P., Ganley, J.L., *et al.*, Performance-oriented Placement and Routing for Field-Programmable Gate Arrays, *Proc. Europ. Design Automat. Conf. (EURO-DAC '95)*, 1995, pp. 80–85.

12. Sechen, C., Chip-planning, Placement, and Global Routing of Macro/Custom Cell Integrated Circuits Using Simulated Annealing, *Proc. XXV Design Automat. Conf. (ACM/IEEE)*, 1998, pp. 73–80.

13. Erzin, A.I. and Cho, J.D., Skew Minimization Problem with Possible Sink Displacement, *Avtom. Telemekh.*, 2003, no. 3, pp. 163–176.

This paper was recommended for publication by A.P. Uzdemir, a member of the Editorial Board