PAPER *Special Section on Discrete Mathematics and Its Applications*

An Improved Algorithm for the Net Assignment Problem

Takao ONO[†a)], *Nonmember and* **Tomio HIRATA**[†], *Regular Member*

SUMMARY In this paper, we consider the net assignment problem in the logic emulation system. This problem is also known as the board-level-routing problem. There are field programmable logic arrays (FPGAs) and crossbars on an emulator board. Each FPGA is connected to each crossbar. Connection requests between FPGAs are called nets, and FPGAs are interconnected through crossbars. We are required to assign each net to the suitable crossbar. This problem is known to be NP-complete in general. A polynomial time algorithm is known for a certain restricted case, in which we treat only 2-terminal nets. In this paper we propose a new polynomial time algorithm for this case.

key words: *logic emulator, net assignment problem, edge coloring, nearly equitable edge coloring*

1. Introduction

Logic verification is the crucial step in the development of VLSI such as application specific IC (ASIC) and CPU chips. In this step we verify that the designed circuit logically satisfies the required specification. The logic simulation software has been used for this step. However, we cannot verify the recent huge circuits with the simulator in practical time. This motivates the logic emulation hardware (logic emulator), which has a lot of field-programmable gate arrays (FPGAs) to implement a prototype of the designed circuit. We can verify the circuit with logic emulator 100 to 1000 times as fast as the simulation software.

Logic emulator has a lot of FPGAs and crossbars. A crossbar is a kind of switch and can arbitrarily connect its own I/O pins. Figure 1 illustrates an example of a logic emulator.

Large circuit is divided into pieces and each piece is programmed into an FPGA. The original circuit is implemented by connecting the I/O pins on the FPGAs through crossbars. Connection requests between FPGAs are called nets, and we assign these nets to the crossbars to realize the circuit. The net assignment problem is that of finding an assignment of the nets to the crossbars. Generally, this problem is known to be NP-complete [5].

Manuscript received August 30, 2000.
Manuscript revised October 20, 2000.

[†]The authors are with the School of Engineering, Nagoya University, Nagoya-shi, 464-8603 Japan.
a) E-mail: takao@hirata.nuee.nagoya-u.ac.jp

2. Preliminaries

We use the following notation for the net assignment problem.

Let $\mathcal{F} = \{F_1, F_2, \ldots, F_f\}$ and $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$ be the set of the FPGAs and the crossbars on the emulator board, respectively. There are m connections between an FPGA and a crossbar. Thus there are $m \times c$ I/O pins on a FPGA and $m \times f$ I/O pins on a crossbar.

A net $N_i \subset \mathcal{F}$ is a set of FPGAs which we need to connect. The cardinality of N_i, denoted by $|N_i|$, is called the size of the net N_i. If the size of a net is t, this net is called a t-terminal net. A netlist is the multiset of the nets $\mathcal{N} = \{N_1, N_2, \ldots, N_n\}$.

The solution to the net assignment problem is an assignment (mapping) $a\colon \mathcal{N} \to \mathcal{C}$, where this assignment must satisfy the constraints that each crossbar has at most m nets including the same FPGA.

It seems that this problem is combinatorially easy when $m = 1$, but it is not; Consider the graph $G = (V, E)$. $V = \mathcal{N}$, that is, each vertex in graph G corresponds to a net in the netlist. $E = \{(N_i, N_j) \mid N_i \cap N_j \neq \emptyset\}$, that is, graph G has an edge (N_i, N_j) if

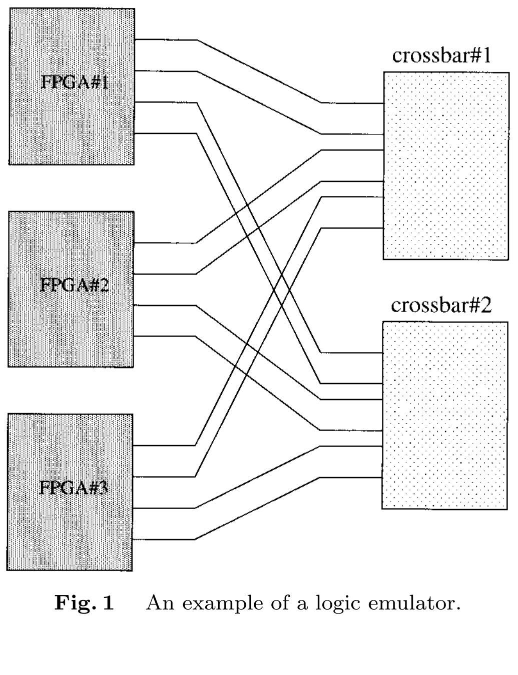


Fig. 1 An example of a logic emulator.

N_i and N_j have at least one FPGA in common. Solving the net assignment problem is equivalent to finding the vertex coloring of the graph G with c colors because of the following argument. Assume that two nets N_i and N_j share FPGA F_k. Since $m = 1$, there is only one wire between F_k and each crossbar. Thus we cannot assign these nets to the same crossbar. This corresponds to assigning the different colors to the adjacent vertices N_i and N_j. For any graph G, there exists a netlist from which we can construct G. This implies that the net assignment problem is NP-complete even if $m = 1$.

Strictly speaking, the net assignment problem has the board and the netlist as input. On the practical point of view, however, we can consider that the board is fixed; The only input is the netlist.

For the general case, there exists a series of greedy algorithms [1]–[4]. These algorithms share the same framework, in which we repeatedly select a net and assign it to a crossbar until all of the nets are assigned. In each step in the algorithms we select a net and a crossbar according to the values of the evaluation functions, whose details differ from one algorithm to other.

In the exceptional case, where every net is a 2-terminal net and m is even, there exists an $O(n^2)$-time algorithm for the netlist with n nets [5]. In Sect. 3 we will propose a new algorithm for this case, which improves the running time.

3. A New Polynomial Time Algorithm for the Restricted Problem

Mak and Wong proved that the special case of the net assignment problem is solvable in polynomial time [5]. Their algorithm runs in $O(n^2)$ time. In this section, modifying the Mak-Wong's algorithm, we show that the running time can be reduced to $O(n^2/c)$.

We restate the restricted version of the net assignment problem as follows:

Problem 1 (m-edge coloring): Given a multigraph $G = (V, E)$ with no self loop, where $|V| = f$ and $|E| = n$, and c colors, find a coloring of the edges so that no more than m edges incident to a node have the same color.

In Problem 1, nodes, edges, and colors correspond to the FPGAs, nets, and crossbars, respectively.

Let us denote the degree of node v by $d(v)$, and the degree of G by $\Delta = \max_{v \in V} d(v)$. We call an edge with color α an α-edge. $d_G(v, \alpha)$ stands for the number of α-edges incident to node v in G, and $e_G(\alpha)$ for the number of α-edges in G. We omit the subscript G if it is clear from the context. Using these notations we write Problem 1 as follows:

Problem 2 (m-edge coloring): Given a graph G and c colors $\mathcal{C}$, color the edges so that the following "capacity constraint" is satisfied:

$$d(v, \alpha) \leq m \text{ for any node } v \in V \text{ and color } \alpha \in \mathcal{C}. \quad (1)$$

It is known that we can find a solution (legal assignment) for any m-edge coloring instance, provided that m is even and $\Delta \leq mc$ [5]. In the following we assume this condition.

3.1 A New Algorithm

Mak and Wong presented an $O(n^2)$-time algorithm for the m-edge coloring problem [5]. Our algorithm is almost the same as Mak-Wong's algorithm. The difference is that our algorithm in fact solves the balanced m-edge coloring problem, a constrained version of Problem 2:

Problem 3 (Balanced m-edge coloring): This problem is the same as Problem 2, except that the coloring must satisfy the following "balance constraint:"

$$|e(\alpha) - e(\beta)| \leq 1 \text{ for any colors } \alpha \text{ and } \beta. \quad (2)$$

The idea of the balance constraint is inspired from Nakano-Suzuki-Nishizeki's algorithm for nearly equitable edge-coloring [6].

Mak-Wong's algorithm initially assigns only one color to all of the edges and repeatedly exchanges two colors violating the capacity constraint. On the other hand, our new algorithm in Fig. 2 assigns each color to $\lceil m/c \rceil$ (or $\lfloor m/c \rfloor$) edges and repeats this exchange of colors maintaining the balance constraint. Note that the initial coloring satisfies the balance constraint (2). In Fig. 2, $G_{\alpha\beta}$ is the subgraph of G induced by α- and β-edges.

Recolor($G_{\alpha\beta}, \alpha, \beta$) will recolor the edges in $G_{\alpha\beta}$ with colors α and β so that capacity constraint for any node v and the following "strict balance constraint for α and β"

$$e_G(\beta) \leq e_G(\alpha) \leq e_G(\beta) + 1 \quad (3)$$

are satisfied. Recolor-Component(H, α, β) does the same process as Recolor($G_{\alpha\beta}, \alpha, \beta$) for a connected component H.

3.2 Analysis of the Algorithm

We show in this section the correctness and the running time of our algorithm. We will give the proof for lemmas of [5] to make this paper self-contained.

Lemma 1 ([5]): The coloring after an invocation of Recolor-Component(H, α, β) for a connected graph $H = (V_H, E_H)$ satisfies the following conditions:

1. If all nodes in H are even-degree and $|E_H|$ is even, then $d(v, \alpha) = d(v, \beta)$ for any node v.
2. If all nodes in H are even-degree and $|E_H|$ is odd, then $d(u, \alpha) - d(u, \beta) = 2$ for the start node u and $d(v, \alpha) = d(v, \beta)$ for any node v other than u.

```
BALANCED-$m$-EDGE-COLORING($G, \mathcal{C}$)
    ▷ solves the balanced $m$-edge coloring problem for $G = (V, E)$
    using colors in $\mathcal{C}$
1   Assign $C_1$, $C_2$, ..., $C_c$ to $m$ edges repeatedly, so that
    $\lceil m/c \rceil$ (or $\lfloor m/c \rfloor$) edges have the same color.
2   while there exists $v \in V$ and $\alpha \in \mathcal{C}$ such that $d(v, \alpha) > m$ do
3       Select $\beta \in \mathcal{C}$ such that $d(v, \beta) < m$
4       RECOLOR($G_{\alpha\beta}, \alpha, \beta$)

RECOLOR($G, \alpha, \beta$)
    ▷ recolors the edges in G with $\alpha$ and $\beta$ so that $e(\beta) \le e(\alpha) \le$
    $e(\beta) + 1$ is satisfied.
1   Let $x \leftarrow \alpha$ and $y \leftarrow \beta$
2   for each connected component $H$ in $G$ do
3       RECOLOR-COMPONENT($H, x, y$).
4       if $H$ has odd number of edges then
5           Swap $x$ and $y$.

RECOLOR-COMPONENT($G, \alpha, \beta$)
    ▷ colors the edges in G with $\alpha$ and $\beta$ so that $e(\beta) \le e(\alpha) \le$
    $e(\beta) + 1$.
1   if $G$ has an odd-degree node then
2       Add a new node $w$ adjacent to all the odd-degree
        nodes to form a new graph $G'$.
3       Traverse the Euler circuit starting at node $w$ and
        assign colors $\beta$ and $\alpha$ ($\beta$ comes first) to the edges
        alternately along the way.
4   else
5       if $G$ has odd number of edges then
6           Select a node $u$ whose degree is not $2m$.
7       else
8           Let $u$ be the arbitrary node.
9       Traverse the Euler circuit starting at node $u$ and
        assign colors $\alpha$ and $\beta$ ($\alpha$ comes first) to the edges
        alternately along the way.
```

Fig. 2 A new algorithm for balanced m-edge coloring.

3. If there are odd-degree nodes in H, then $|d(v,\alpha) - d(v,\beta)| = 1$ for any odd-degree node v and $d(v,\alpha) = d(v,\beta)$ for any even-degree node v.

Proof: We first prove the cases 1 and 2. H has an Euler circuit because all nodes are even-degree. We traverse the Euler circuit starting and ending at u and assign colors α and β to the edges alternately along the way. It is obvious that $d(v,\alpha) = d(v,\beta)$ for any node v in the Euler circuit other than u. If $|E_H|$ is even, then the first edge of the Euler circuit is α-edge and the last edge is β-edge, which implies that $d(u,\alpha) = d(u,\beta)$. On the other hand, if $|E_H|$ is odd, then both the first and the last edges are α-edges. This implies that $d(u,\alpha) = d(u,\beta) + 2$.

Now we prove the case 3. Because $\sum_{v \in V_H} d(v) = 2|E_H|$, there must be even number of odd-degree nodes. Thus all nodes in $H' = (V_{H'}, E_{H'})$ (introduced at line 2) are even-degree. We consider two cases.

If $|E_H|$ is even, implying that $|E_{H'}|$ is also even, then $d(v,\alpha) = d(v,\beta)$ for any node in H' including w. When we remove w and its incident edges, one edge (v,w) for any odd-degree node v is removed. Thus for any odd-degree node v, $|d(v,\alpha) - d(v,\beta)| = 1$. For any even-degree node v, it is obvious that $d(v,\alpha) = d(v,\beta)$.

If $|E_H|$ is odd, then $d(w,\alpha) = d(w,\beta) + 2$ and $d(v,\alpha) = d(v,\beta)$ for any node $v \in V_{H'} \setminus \{w\}$ from the case 2 of the lemma applied to H'. We can apply the argument for the previous case to show that for any odd-degree node v, $|d(v,\alpha) - d(v,\beta)| = 1$ and for any even-degree node v, it is obvious that $d(v,\alpha) = d(v,\beta)$. □

Lemma 2 ([5]): Assume that m is even. The coloring of H after the invocation of RECOLOR-COMPONENT(H, α, β) satisfies the following condition: For any node v,

1. $d(v,\alpha) \le m$ and $d(v,\beta) \le m$ if $d(v,\alpha) + d(v,\beta) \le 2m$ before the invocation, and
2. $d(v,\alpha) \ge m$ and $d(v,\beta) \ge m$ if $d(v,\alpha) + d(v,\beta) \ge 2m$ before the invocation.

Proof: If $H = (V_H, E_H)$ has even number of edges or H has odd-degree nodes, then $|d(v,\alpha) - d(v,\beta)| \le 1$ for any node v. The lemma obviously holds in this case.

Now we consider the case that $|E_H|$ is odd and all nodes in H are even-degree. In this case, there must be a node w such that $d(w) \ne 2m$. Otherwise, $\sum_{v \in V_H} d(v) = 2m|V_H| = 2|E_H|$, that is, $m|V_H| = |E_H|$. This is impossible, however, because m is even and $|E_H|$ is odd. Thus we pick u as the start node for which $d(u) \ne 2m$ at line 6 of RECOLOR-COMPONENT. For any node other than u, the lemma is obvious. For the node u, $d(u) = 2d(u,\alpha) - 2 = 2d(u,\beta) + 2$ from Lemma 1. Thus if $d(u) \le 2m - 2$, $d(u,\beta) < d(u,\alpha) \le m$. Otherwise, $d(u,\alpha) > d(u,\beta) \ge m$. □

Corollary 1: Lemma 2 holds for RECOLOR($G_{\alpha\beta}, \alpha, \beta$) instead of RECOLOR-COMPONENT(H, α, β).

Proof: Consider componentwise. □

Lemma 3: The coloring after an invocation of RECOLOR-COMPONENT(H, α, β) satisfies the strict balance condition, that is, $e_H(\beta) \le e_H(\alpha) \le e_H(\beta) + 1$.

Proof: If $H = (V_H, E_H)$ has no odd-degree nodes, the lemma is obvious: If $|E_H|$ is even, then the coloring inside RECOLOR-COMPONENT begins with α and ends with β, implying that $e_H(\alpha) = e_H(\beta)$. If $|E_H|$ is odd, then the coloring begins and ends with α, implying that $e_H(\alpha) = e_H(\beta) + 1$.

Now we consider the case where H has odd-degree nodes. In this case RECOLOR-COMPONENT can be viewed as the procedure which assigns β and α alternately to the edges staring at w along the way of an Euler circuit and removes node w and its incident edges. If $|E_H|$ is even (and also is $|E_{H'}|$), $e_{H'}(\alpha) = e_{H'}(\beta)$. Because $d_{H'}(w,\alpha) = d_{H'}(w,\beta)$, removing w means that $e_H(\alpha) = e_H(\beta)$. Finally if $|E_H|$ (and $|E_{H'}|$) is odd, then $e_{H'}(\beta) = e_{H'}(\alpha) + 1$ because the coloring begins and ends with β. The coloring begins and ends at node w, which implies that $d_{H'}(w,\beta) = d_{H'}(w,\alpha) + 2$. Thus removing w means that $e_H(\alpha) = e_H(\beta) + 1$. □

Corollary 2: Lemma 3 holds for RECOLOR$(G_{\alpha\beta}, \alpha, \beta)$ instead of RECOLOR-COMPONENT(H, α, β).

Proof: RECOLOR$(G_{\alpha\beta}, \alpha, \beta)$ invokes RECOLOR-COMPONENT(H, x, y) for all connected component H and swaps the colors if the component has odd number of edges. Thus we only have to consider the connected component with odd number of edges.

Assume that $G_{\alpha\beta}$ has s connected component H_1, $H_2, \ldots, H_s$ with odd number of edges. For odd i, we invoke RECOLOR-COMPONENT(H_i, x, y), where $x = \alpha$ and $y = \beta$, and obtain the coloring with $e_{H_i}(\alpha) = e_{H_i}(\beta)+1$. For even i, the coloring will satisfy $e_{H_i}(\beta) = e_{H_i}(\alpha) + 1$. Summing up these equalities, we conclude that $e_G(\alpha) = e_G(\beta)$ if s is even, and $e_G(\alpha) = e_G(\beta)+1$ if s is odd. □

Corollary 3: At any time of our algorithm, the inequality $|e(\alpha) - e(\beta)| \leq 1$ holds for any colors α and β.

Proof: Proof is done by the induction of the number of invocations of RECOLOR.

In the basis, at the beginning of the algorithm, the lemma holds because we assign the colors equitably to all the edges, which implies that $|e(\alpha) - e(\beta)| \leq 1$ for any colors α and β.

We assume that the lemma holds after the k-th invocation of RECOLOR. Now we consider the $(k+1)$-st invocation. If the colors α and β satisfy the equality $e(\alpha) = e(\beta)$, then $e(\alpha) = e(\beta)$ should hold after the invocation due to Cor. 2. On the other hand, if the colors α and β satisfies the equality $|e(\alpha) - e(\beta)| = 1$, then after the invocation the same inequality must be satisfied also due to Cor. 2. We conclude that in either case the lemma holds after the $(k+1)$-st invocation of RECOLOR. □

We define the excess $\Phi(v)$ for the node $v \in V$ as follows:

$$\Phi(v) = \sum_{\alpha\in\mathcal{C}:\ d(v,\alpha)>m} \bigl(d(v,\alpha) - m\bigr).$$

The excess Φ of the coloring is the summation of $\Phi(v)$ over all nodes v. We note that $\Phi \geq 0$ for any coloring.

The following lemma is used to prove that our algorithm terminates in finite time.

Lemma 4: Assume that there exist a node v_0 and colors α, β such that $d(v_0, \alpha) > m$, $d(v_0, \beta) < m$. Φ must decrease if we invoke RECOLOR$(G_{\alpha\beta}, \alpha, \beta)$.

Proof: Let $\Delta\Phi(v)$ denotes the difference of $\Phi(v)$ before and after the invocation of RECOLOR$(G_{\alpha\beta}, \alpha, \beta)$. In the following, $d(v, \alpha)$ and $d'(v, \alpha)$ denote the numbers of α-edges before and after the invocation of RECOLOR, respectively.

If $d(v, \alpha) \leq m$ and $d(v, \beta) \leq m$, then $d'(v, \alpha) \leq m$ and $d'(v, \beta) \leq m$ (Lem. 2). Thus $\Delta\Phi(v) = 0 - 0 = 0$.

If $d(v, \alpha) > m$ and $d(v, \beta) \geq m$, then $d'(v, \alpha) \geq m$ and $d'(v, \beta) \geq m$. Thus

$$\begin{aligned}\Delta\Phi(v) &= \bigl[\bigl(d'(v,\alpha) - m\bigr) + \bigl(d'(v,\beta) - m\bigr)\bigr] \\ &\quad - \bigl[\bigl(d(v,\alpha) - m\bigr) + \bigl(d(v,\beta) - m\bigr)\bigr] \\ &= \bigl(d'(v,\alpha) + d'(v,\beta)\bigr) - \bigl(d(v,\alpha) + d(v,\beta)\bigr) \\ &= 0,\end{aligned}$$

because $d(v, \alpha) + d(v, \beta) = d'(v, \alpha) + d'(v, \beta)$.

If $d(v, \alpha) > m$ and $d(v, \beta) < m$, we further consider two cases. If $d(v, \alpha) + d(v, \beta) \geq 2m$, then $d'(v, \alpha) \geq m$ and $d'(v, \beta) \geq m$. Thus

$$\begin{aligned}\Delta\Phi(v) &= \bigl[\bigl(d'(v,\alpha) - m\bigr) + \bigl(d'(v,\beta) - m\bigr)\bigr] \\ &\quad - \bigl(d(v,\alpha) - m\bigr) \\ &= \bigl(d'(v,\alpha) + d'(v,\beta)\bigr) - d(v,\alpha) - m \\ &= d(v,\beta) - m < 0.\end{aligned}$$

Thus, in this case $\Phi(v)$ will decrease. If $d(v, \alpha) + d(v, \beta) \leq 2m$, then $d'(v, \alpha) \leq m$ and $d'(v, \beta) \leq m$. Thus

$$\Delta\Phi(v) = 0 - \bigl(d(v,\alpha) - m\bigr) = m - d(v,\alpha) < 0.$$

In all cases, $\Phi(v)$ must not increase. Moreover, $\Phi(v_0)$ must decrease because v_0 matches one of the last two cases. Thus we have proven the lemma. □

The value of Φ is bounded as follows:

$$\begin{aligned}\Phi &= \sum_{v\in V} \Phi(v) \\ &= \sum_{v\in V} \sum_{\alpha:\ d(v,\alpha)>m} \bigl(d(v,\alpha) - m\bigr) \\ &\leq \sum_{v\in V} \sum_{\alpha\in\mathcal{C}} d(v,\alpha) \\ &= \sum_{v\in V} d(v) = 2|E| = 2n.\end{aligned}$$

From Lemma 4, in the course of the algorithm the value of Φ must decrease by at least 1 when we invoke RECOLOR. Thus after at most $2n$ invocations of RECOLOR, Φ must be 0.

Now we can prove the main theorem.

Theorem 1: BALANCED-m-EDGE-COLORING solves the balanced m-edge coloring problem in $O(n^2/c)$ time.

Proof: The balance constraint is always satisfied (Cor. 3). Because $\Phi = 0$ implies that the capacity constraint is satisfied, the algorithm terminates after at most $2n$ invocations of RECOLOR with a legal coloring.

Now we analyze the running time of RECOLOR.

Lemma 5: The running time of RECOLOR$(G_{\alpha\beta}, \alpha, \beta)$ is $O(e(\alpha) + e(\beta))$.

Proof: First we consider the running time of RECOLOR-COMPONENT(H, α, β) for the connected graph $H = (V_H, E_H)$. The running time of the traversal of an Euler circuit of a graph is linear to the number of edges. Thus, the traversal of an Euler circuit of H takes $O(|E_H|)$ time. On the other hand, the augmented graph H' has at most $V_H + E_H$ edges. Thus, the traversal of an Euler circuit of H' takes $O(V_H + E_H + n)$ time. In both cases, the running time of RECOLOR-COMPONENT(H, α, β) is $O(V_H + E_H)$.

Summing up the result of this lemma for all connected component of $G_{\alpha\beta}$, the running time of RECOLOR$(G_{\alpha\beta}, \alpha, \beta)$ is $O(e(\alpha) + e(\beta) + f)$ since $G_{\alpha\beta}$ has $e(\alpha)+e(\beta)$ edges and at most $|V| = f$ nodes. Moreover, for any graph $G = (V, E)$, $|V| \leq 2|E|$. Thus the lemma is proved. □

The following lemma shows the effect of adding the balance constraint (2).

Lemma 6: $e(\alpha) = O(n/c)$ for any color α at any time of the algorithm.

Proof: We assume that $e(\alpha) \geq e(x)$ for any color $x \in \mathcal{C}$, that is, α is the color with which we have the most number of edges. Because the balance constraint is satisfied, $e(\alpha) \leq e(x) + 1$ for any color x. This implies the inequality $e(\alpha) \leq n/c + 1$. Otherwise, the inequality $e(x) \geq e(\alpha) - 1 \geq n/c$ implies

$$\begin{aligned} n = |E| &\geq \sum_{x \in \mathcal{C}} e(x) \\ &= e(\alpha) + \sum_{x \in \mathcal{C} \setminus \{\alpha\}} e(x) \\ &\geq \left(\frac{n}{c} + 1\right) + (c-1) \cdot \frac{n}{c} = n + 1, \end{aligned}$$

a contradiction. Thus, at any time of the algorithm, $e(\alpha) = O(n/c)$ for any color α. □

From Lems. 5 and 6, RECOLOR$(G_{\alpha\beta}, \alpha, \beta)$ takes $O(n/c)$ time.

We conclude that the running time of BALANCED-m-EDGE-COLORING is $O(n \times n/c) = O(n^2/c)$. □

4. Conclusion

In this paper we gave a new algorithm for the special case of the net assignment problem and proved that its running time is $O(n^2/c)$, where n is the number of the nets and c is the number of the crossbars. This improves the previous $O(n^2)$-time algorithm given by Mak and Wong [5]. There are 100 or more crossbars on a board in application, thus this makes a great improvement in the computation time.

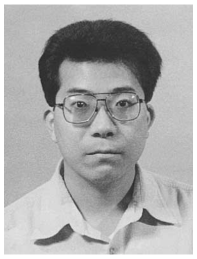
If m is odd or there exist multiterminal nets, we can treat them in the same manner as in [5]. If m is odd, we may limit ourselves to use at most $m-1$ wires between an FPGA and a crossbar. Thus if m is odd, we can still apply our algorithm in this way, provided $\Delta \leq (m-1)c$. For multiterminal nets, we use the multilayer crossbar structure and divide each multiterminal nets into distinct 2-terminal nets. See [5] for detail.

Acknowledgments

The research is supported in part by Tatematsu Foundation and a Scientific Research Grant-In-Aid from the Ministry of Education, Science and Culture, Japan.

References

[1] M. Butts, J. Batcheller, and J. Vargheseq, "An efficient logic emulation system," Proc. IEEE Int. Conf. Computer Design, pp.138–141, Oct. 1992.
[2] D. Ito, T. Ono, and T. Hirata, "A study on the heuristic algorithm for the net assignment problem," IEICE Technical Report, COMP2000-14, 2000.
[3] N. Funabiki and J. Kitamichi, "A neural-greedy combination algorithm for board-level routing in FPGA-based logic emulation systems," IEICE Trans. Fundamentals, vol.E81-A, no.5, pp.866–872, 1998.
[4] S.-S. Lin, Y.-J. Lin, and T.-T. Hwang, "Net assignment for the FPGA-based logic emulation system in the folded-clos network structure," IEEE Trans. Comput.-Aided Des., vol.16, no.3, pp.316–320, 1997.
[5] W.-K. Mak and D.F. Wong, "On optimal board-level routing for FPGA-based logic emulation," IEEE Trans. Comput.-Aided Des., vol.16, no.3, pp.282–289, 1997.
[6] S. Nakano, Y. Suzuki, and T. Nishizeki, "An algorithm for the nearly equitable edge-coloring of graphs," IEICE Trans., vol.J78-D-I, no.5, pp.437–444, 1995.



Takao Ono received B.E., M.E. and Ph.D. from Nagoya University in 1993, 1995 and 1999, respectively. He is currently a Research Associate in Department of Electronics, Nagoya University. His research interests include approximation algorithms.

Tomio Hirata received B.S., M.S. and Ph.D. in Computer Science, all from Tohoku University in 1976, 1978, and 1981, respectively. He is currently a Professor in Department of Electronics, Nagoya University. His research interests include graph algorithms and approximation algorithms.