A New System Partitioning Method under Performance and Physical Constraints for Multi-Chip Modules

Yoshinori Katsura* Tetsushi Koide* Shin'ichi Wakabayashi* Noriyoshi Yoshida†

Faculty of Engineering, Hiroshima Univ.*
4-1, Kagamiyama 1 chome,
Higashi-Hiroshima, 739 JAPAN
Tel: +81-824-24-7676, Fax: +81-824-22-7195
e-mail: koide@ecs.hiroshima-u.ac.jp

Faculty of Information Sciences, Hiroshima City Univ.†
151-5, Ozuka, Numata-Cho,
Asa-Minami-Ku, Hiroshima 731-31, JAPAN
Tel: +81-82-830-1760, Fax: +81-82-830-1792
e-mail yoshida@ce.hiroshima-cu.ac.jp

Abstract— In this paper, we propose a new and the first MCM system partitioning method considering chip-to-chip delays, chip areas and I/O pins constraints. The proposed method consists of two steps, a clustering step considering the three constraints and an iterative improvement step with mathematical programming. In the first step, we apply two clustering algorithms considering the three constraints and reduce the size of the large MCM system partitioning problem so as to get a solution within a practical computation time. Next, it generates an initial partitioning with 0-1 integer linear programming(ILP) so as to minimize the total wire length. Since there may exist constraint violations in the initial solution, in the second step, we formulate the partitioning problem as a LP problem selecting a maximal independent set and improve it until the total number of cuts is not decreased and the three constraints are satisfied. We also showed that the number of the timing constraints can be reduced by deleting redundant timing constraints. Experimental results showed that the proposed method is able to produce partitions satisfying the three constraints and improves the number of cuts by a 27 % on an average and a 30 % in maximum over the conventional method[15] considering only two constrains.

I Introduction

Recently, as VLSI fabrication technology has been advancing rapidly, the development of an electronic system with high performance and smallsized is strongly needed. A printed circuit board (PCB) has been used as a packaging technology for conventional electronic systems. In this technology, every packaged chip is mounted on a PCB and wires are interconnected between packaged chips. Accordingly, the wire density on and the area of a PCB have become a bottleneck for implementing a high performance system and some better packaging technologies have been desired. Therefore, Multi-Chip Module (MCM) technology has been attracted as a new packaging approach [2, 4]. This technology is able to increase the packaging density and to eliminate the single chip package by mounting and interconnecting the bare chips directly onto a higher density substrate. Consequently, MCM is smaller in the size and in the shorter total wire length than PCB.

For physical design of MCMs, we need to consider delays between chips for the performance constraint, and chip areas, the number of chip I/O pins, the thermal, and the consuming power for the physical constraints. But conventional partitioning and clustering methods [1, 3, 5, 6, 12, 16, 17, 19] considered only the chip area constraint. Therefore it is not adequate to apply these methods for ICs to MCM directly. Recently, several partitioning methods considering such constraints have been proposed[10, 11, 13, 14, 15, 18]. In Ref. [15], Shih and Kuh proposed a partitioning method under the chip I/O pin and the chip area constraints. But they do not consider the timing constraint. In Ref.[18], Woo and Kim also considered only the chip area and the chip I/O pins constraints for FPGAs. In Ref. [10, 11, 13, 14], the chip areas and the timing constraints were considered but the chip I/O pin constraint was not considered. There are no methods considering the delay, the chip area, and the chip I/O pins constraints up to the present.

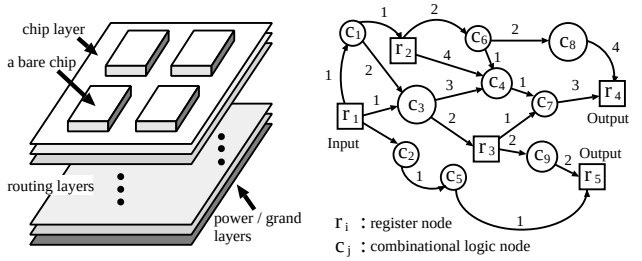
In this paper, we present a new and the first MCM system partitioning method under the three constraints mentioned above. The proposed method consists of two steps, a clustering step considering the three constraints and an iterative improvement step with mathematical programming. In the first step, we use two clustering algorithms under the constraints to reduce the size of the problem as much as possible so that the proposed method can handle a large size MCM partitioning problem within a practical computation time. Next, we select a maximal independent set to represent the number of cuts with a linear expression exactly, and formulate the partitioning problem to a 0-1 ILP problem to the selected set. And then, we convert the 0-1 ILP problem as a LP problem under three constraints and solve the problem. This iterative improvement step is repeated while the number of cuts is improved and the three constraints mentioned above are satisfied. We also showed that the number of the timing constraints can be reduced by deleting redundant timing constraints and representing the constraints, whose source node is assigned to the same chip, with one constraint. Experimental results showed that the proposed method improves the number of cuts by a 27 % on an average and a 30 % in maximum over the conventional method[15] considering only two constraints. From experimental results for large size data, we can also obtain a solution within a practical computation time applying a clustering method considering the three constraints.

The reminder of this paper is organized as follows. Some assumptions of our system partitioning problem, the interconnection delay model, and the timing constraints will be given in Section II. Then we will propose a new MCM system partitioning algorithm in Section III. In Section IV, experimental results for the proposed method will be given. Finally, we will conclude this paper in Section V.

II System Partitioning for MCM

A. Preliminaries

Figure 1(a) shows a diagram of an MCM substrate. We assume that an MCM consists of a chip layer, on which bare chips without the individual packages are mounted, multiple routing layers providing all of the chip-to-chip interconnections, and a power/grand layer.



(a) MCM model. (b) System graph.

Fig. 1. MCM model.

In this paper, for simplicity, we assume that a system consists of registers and blocks of combinational logic, and all clock generation and distribution circuits are ignored. When a system is given as inputs, the system can be represented as a graph $G = (V, E) \equiv (R \cup C, E)$, where $V = \{v_1, v_2, \cdots, v_I\}$ is a set of register nodes R and combinational nodes C, and E is a set of directed edges, which correspond to signal flows in the system. The graph is called *system graph*[10]. Each edge is associated with a weight ω_{lm}, which represents the number of wiring between two nodes $v_l, v_m \in V$. Figure 1(b) shows an example of a *system graph*. The rectangular and circular nodes correspond to registers and blocks of combinational logic, respectively. The directed edges represent the signal flows. The values attached to directed edges are the number of wiring between two nodes.

Let $S = \{s_1, s_2, \cdots, s_J\}$ be the set of chips (slots), where J is the number of chips. We define $M = (G, S)$ as an *MCM system*. The positions of bare chips on the MCM are fixed. The center coordinate of a chip $s_j \in S$ is denoted by (px_j, py_j). For each chip $s_j \in S$, chip areas A_j $(1 \leq j \leq J)$ and the number of I/O pins of chips IO_j $(1 \leq j \leq J)$ are given as inputs. For simplicity of the presentation, we assume that all bare chip areas are the same size, that is, $A_1 = A_2 = \cdots = A_J$, and all multi-terminal nets are converted into two-terminal nets under a weighted clique model like Ref.[15] in advance. For the case that areas of bare chips are different, we can easily extend the proposed method.

B. An Interconnection Delay Model

In general, an interconnection is modeled as a distributed RC circuit[9]. Internal delays of registers and combinational blocks are less than interconnection delays between chips. Therefore, we suppose that internal delays of registers and combinational blocks are equal to zero. To calculate the interconnection delays between chips, we estimate the wire length of a net by the half perimeter of bounding box of the pins of nets. For simplicity, we assume that the interconnection delay is proportional to the wire length. Then the delay D_{pq} $(1 \leq p, q \leq J)$ from a source node assigned to the chip s_p to a load node assigned to the chip s_q of a net is represented by the following equation.

$$D_{pq} = D_{unit\_length} \times length, \tag{1}$$

where, D_{unit_length} is the per unit length wiring delay which depends on the material(dielectric constant ϵ_τ). $length$ is the half perimeter of bounding box of the pins of the net, that is, $length = |px_p - px_q| + |py_p - py_q|$.

C. Timing Constraint

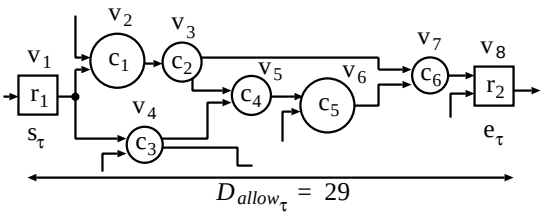
In this paper, we consider the long path problem. As there are many paths from a primary input(PI) or an output of flip-flops(FFs) to a primary output(PO) or inputs of FFs, they can be specified by pairs of pins, source ones and sink ones. Thus we specify a timing constraint as $t_\tau = (s_\tau, e_\tau, D_{allow_\tau})$, where s_τ is a source pin (or node), e_τ is a sink pin (or node), and D_{allow_τ} is the maximum allowable delay from the source to the sink. If delays of all paths from a PI or an output of FF to a PO or an input of FF are less than D_{allow_τ}, it satisfies the timing constraint of the circuit.

For example, Fig. 2(b) shows *the constraint graph* of a timing constraint τ, denoted $G_\tau = (V_\tau, E_\tau), V_\tau \subseteq V, E_\tau \subseteq E$, of the circuit shown in Fig. 2(a). In this graph, nodes correspond to blocks of a subcircuit corresponding to a timing constraint τ and directed edges correspond to signal flows in the system. If a circuit and its timing constraint are given, the delay of any path from s_τ to e_τ, in this case three paths, must be less than D_{allow_τ}. We have to get the layout satisfying all elements of the set of timing constraints $\mathcal{T}$.

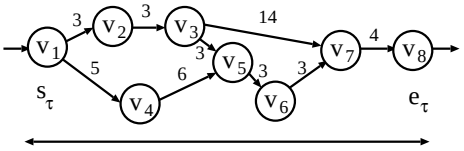
We apply the zero slack algorithm [7] to distribute D_{allow_τ} to the maximum allowable delays d_{lm} $(1 \leq l, m \leq I)$ between of the edge of two nodes $v_l, v_m \in V_\tau$. Nodes have to be assigned to chips so as to satisfy the maximum allowable delays between nodes. We can get the maximum allowable delays between nodes as shown in Figure 2 when we assume that $D_{allow_\tau} = 29$ and apply the zero slack algorithm [7]. In Figure 2(b), the maximum allowable delay between nodes v_3 and v_7, i.e., d_{37}, is 14, and it is the largest slack. The maximum allowable delays between nodes, d_{12}, d_{23}, d_{35}, d_{56}, and d_{67}, are 2 and they are the smallest slack.

D. The MCM System Partitioning Problem

In the MCM design, we should consider delays between nodes for performance and chip areas and the number of I/O



(a) A timing constraint



(b) A constraint graph

Fig. 2. A timing constraint.

pins of chips for physical constraint. The number of cuts is the number of the nets which are connected among different chips. We define that oe_{ij} is the number of nets that go out from a chip s_j when a node v_i is assigned to a chip s_j. Then we can represent the number of cuts with $\frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J} oe_{ij}$ where a node must be assigned to only one chip necessarily. Now, we formulate the MCM partitioning problem considering above three constraints as follows.

[The MCM System Partitioning Problem]
Input: an MCM system $M = (G, S)$
Output: a chip assignment $F : V \rightarrow S$ so as to minimize the objective function
Objective function: the number of cuts $\frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J} oe_{ij}$
Constraints: (1) timing constraints $\mathcal{T}$, (2) chips areas constraints $\sum_{\forall v_i, F(v_i)=s_j} a_i \leq A_j$, and (3) chip I/O pins constraints $\sum_{\forall v_i, F(v_i)=s_j} oe_{ij} \leq IO_j$ □

Figure 3 shows an example of the MCM partitioning problem, where the area of nodes a_1 and a_4 are 2, and the areas of the other nodes are 1. The chip area constraint A_j is 3 and the I/O pin constraint IO_j is 4. The total of node areas assigned to each chips s_1 and s_3 are 2, and that of chips s_2 and s_4 are 3, respectively. The total number of I/O pins of chips s_1 and s_4 is 3, and that of chips s_2 and s_3 are 4 and 2, respectively. The number of cuts is 5. We suppose that the maximum allowable delays between nodes are all $1 \times D_{min}$ in Figure 3, where $D_{min} = min\ D_{pq}$ $(1 \leq p, q \leq J)$ is the minimum interconnection delay between chips. In this example, the interconnection delay between chips s_1 and s_4, and chips s_2 and s_3 is $2 \times D_{min}$, respectively. The interconnection delays between the other chips are all $1 \times D_{min}$.

III A New Partitioning Method under Performance and Physical Constraints

A. Outline of the Method

The proposed method consists of four phases. In phases 1 and 2, we apply two clustering algorithms considering constraints mentioned in the previous section and reduce the number of nodes so as to get a solution within a practical computation time. Next, it generates an initial partitioning based on 0-1 integer linear programming(ILP) so as to minimize the total wire length in phase 3. Since there may exist constraint violations in phase 3, in phase 4, we improve the partition with 0-1 ILP until the timing, the chip area, and the I/O pin constraints are satisfied. If we solve the 0-1 integer linear programming problem directly, it is difficult to obtain a solution in a practical computation time. So, we convert the 0-1 ILP problem into a linear programming(LP) problem and solve it. We move nodes to minimize the total number of cuts under the timing, the I/O pin, and the chip area constraints. The improvement process is repeated while the total number of cuts is decreased and the three constraints are satisfied. We will explain the details of the proposed method in the following subsections.

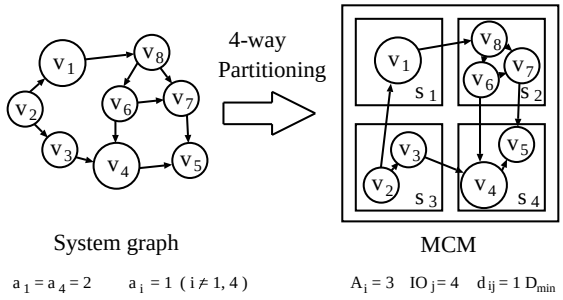


Fig. 3. System partitioning considering three constraints.

B. Phase 1 : Initial Clustering under Timing Constraint

In order to reduce the computation time, the clustering has been known as an effective method to reduce the number of nodes [3, 6, 19]. But it is difficult to apply conventional clustering algorithms for ICs to the MCM system partitioning problem, since we must consider interconnection delays between nodes for the performance constraint, and the chip area and the number of chip I/O pins for the physical constraints.

We propose a clustering algorithm considering such three constraints. Firstly, we cluster all pairs of the nodes that cause a timing constraint violation if a pair of nodes are assigned to different chips. In other words, a pair of nodes $v_l, v_m \in V$ generate timing violation only if the maximum allowable delay d_{lm} between nodes v_l and v_m is less than the minimum interconnection delay between chips, i.e., $d_{lm} < D_{min}(= min_{1 \leq i \neq j \leq J} D_{ij})$. So, we cluster such all pairs of nodes. This clustering process is repeated until all pairs of cluster nodes have the maximum allowable delay $d_{lm} \geq D_{min}$ [10]. After phase 1, all pairs of nodes, which are connected each other, have $d_{lm} \geq D_{min}$. However, if the area of a cluster node is larger than the chip area constraint, there is no feasible solution and terminate with reporting an error. The algorithm of phase 1 is shown below.

[Phase 1 : Initial Clustering under Timing Constraint]
Step 1.1 Find a pair of nodes $v_l, v_m \in V$ with $d_{lm} < D_{min}$;
If there is no such a pair of nodes, then go to **Step 1.4**;
Step 1.2 Cluster nodes v_l and v_m to v_l;

Step 1.3 If all pairs of cluster nodes have the maximum allowable delay $d_{lm} \geq D_{min}$, then terminate;

Step 1.4 If there is a pair of nodes with $d_{lm} < D_{min}$ and $a_l + a_m > min_{1 \leq i \leq J} A_i$ then report that there is no feasible solution and terminate, else go to **Step 1.1**;

Since the algorithm checks all pairs of nodes, the computation time is $O(|V|^2)$, where $|V|$ is the number of nodes.

C. *Phase 2 : Clustering under the Three Constraints*

We propose a new clustering method considering the timing, the area, and the chip I/O pins constraints, and reduce the number of nodes so as to solve the large MCM partitioning problem within a practical computation time. The delay is said to be *critical* if $d_{lm} - D_{min} < preset\_value$. In the proposed method, we cluster a pair of nodes $v_l, v_m \in V$, for which the maximum allowable delay d_{lm} is critical, the area is $a_l + a_m > min_{1 \leq i \leq J} A_i$, and the connectivity ω_{lm} is large.

In general, it is practically difficult to satisfy the chip I/O pin constraints before determining a chip assignment of nodes. Therefore we cluster nodes considering the connectivity ω_{lm} of nodes in order to satisfy the I/O pin constraint. Given a pair of nodes $v_l, v_m \in V$, we define the cost function $CF1(l,m)$ as,

$$\begin{aligned} CF1(l,m) &= \alpha \times \omega_{lm} + \beta \times \frac{A_{average}}{a_l + a_m} \\ &+ \frac{1}{d_{lm} - D_{min} + 1}, \end{aligned}$$

where, α, β are constants, and ω_{lm} and $A_{average}$ are the connectivity between nodes v_l, v_m and the average of chip areas, respectively. Larger the value of the first term becomes, higher the possibility of clustering the pair of the nodes becomes. Since the second term represents the ratio of the average of chip areas $A_{average}$ to the sum of areas a_l and a_m of nodes v_l and v_m, larger the value of the second term becomes, higher the possibility of clustering the pair of nodes becomes. In the third term, $(d_{lm} - D_{min})$ represents the degree of criticality of the delay between nodes v_l and v_m. As the value of the third term becomes smaller, the possibility of clustering the pair of the nodes becomes higher.

When we compute the cost function $CF1(l,m)$, we are able to get the degrees of the clustering priority between nodes v_l and v_m under the three constraints. Therefore we cluster a pair of nodes v_l and v_m whose value of the cost function $CF1(l,m)$ is maximum. This process is repeated while the number of cluster nodes is larger than a pre-defined number. The algorithm of phase 2 is shown as follows.

[Phase 2 : Clustering under the Three Constraints]

Step 2.1: Compute the cost $CF1(l,m)$ of all pairs of nodes $v_l, v_m \in V$;

Step 2.2: Clustering a pair of nodes v_l and v_m whose cost $CF1(l,m)$ is maximum and satisfies the three constraints;

Step 2.3: If the number of cluster nodes is smaller than a predefined number, then terminate; else go to **Step 2.1**;

Since the algorithm computes costs of all pairs of nodes, the computation time is $O(|V|^2)$, where n is the number of nodes.

In phases 1 and 2, the clustered nodes $v_l, v_m \in V$ must be assigned to the same chip. Therefore the interconnection delay between the nodes v_l and v_m is much less than the interconnection delay between nodes assigned to different chips. We can update the maximum allowable delay d_{lm} between the clustered nodes v_l and v_m so as to redistribute the maximum allowable delay to the path to which the timing constraint is given. Even if we update the maximum allowable delay between the nodes on the path, the maximum allowable delay between flip flops D_{allow_τ} is still satisfied. In the following, we define the set of cluster nodes in phases 1 and 2 as $v_i \in CV$ and call a cluster node a node for short.

D. *Phase 3 : Initial Partitioning Based on 0-1 Integer Linear Programming*

In phase 3, we get an initial partition with 0-1 integer linear programming. The objective function is the total wire length under the chip area and timing constraints. It is very difficult to consider the I/O pin constraint in phase 3 because the number of I/O pins of chips is unable to be represented in a linear expression generally. Simirally the number of cuts can not be represented by a linear expression generally. This is the reason why we consider the total wire length as the objective function. Since we can not consider all constraints in the 0-1 ILP, there may exist constraint violations in an initial partitioning obtained in phase 3. We remove the constraint violations by an iterative improvement in phase 4. The 0-1 ILP formulation of the initial partitioning problem is as follows.

[The 0-1 ILP Formulation of the Initial Partitioning Problem]

Input: MCM System $M = (G, S)$

Output: A chip assignment $F1 : CN \rightarrow S$ so as to minimize the objective function

Objective: *the total wire length*

$$\sum_{l=1}^{I'} \sum_{m=1}^{I'} Dc_{pq}(x_{lp} + x_{mq})$$

Constraints: (1) the timing constraint

$$D_{pq}(x_{lp} + x_{mq} - 1) \leq d_{lm}$$

(2) the chip area constraint

$$\sum_{i=1}^{I'} a_i x_{ij} \leq A_j \quad (1 \leq j \leq J)$$

(3) the chip assignment constraint

$$\sum_{j=1}^{J} x_{ij} = 1 \quad (1 \leq i \leq I')$$

(4) $x_{ij} = \begin{cases} 1 & F1(n_i) = s_j \\ 0 & otherwise \end{cases}$

Practically, the timing constraint should be satisfied when $x_{lp} = x_{mq} = 1$. Since the timing constraint is always satisfied when x_{lp} or x_{mp} is 0, the timing constraint formulated above is equivalent to the original timing constraint. Since all nodes are assigned to any one of chips necessarily, $\sum_{j=1}^{J} x_{ij} = 1$.

E. Iterative Improvement Based on 0-1 Integer Linear Programming

1. Outline of Phase 4

If more than one nodes are connected each other and they move from a chip to another chip at one time, the number of cuts can not be represented with a linear expression. Therefore, we select a set of nodes not connected each other, i.e., a maximal independent set of nodes, to formulate the partitioning problem with linear expressions. Firstly, we compute oe_{ij} of the initial partition obtained in phase 3, where oe_{ij} is the number of edges of node v_i going out from chip s_j. Next, we select a maximal independent set and formulate the problem to the 0-1 ILP problem. In order to obtain a solution in a practical computation time, we convert the 0-1 ILP problem into a linear programming(LP) problem and solve it. If a solution has a variable whose value is an integer (0 or 1), we presume this variable as a constant and apply the LP again. The process is repeated until all variables of the solution become integer constants (0 or 1). The improvement is repeated while the number of cuts is decreased and three constraints (timing, chip area, and I/O pins of chips) are satisfied. The algorithm of phase 4 is shown below.

[Phase 4 : Iterative Improvement based on 0-1 ILP Formulation]

Step 4.1: Select a maximal independent set;

Step 4.2: Formulate a partitioning problem as a LP problem and solve it;

Step 4.3: If a variable x_{ij} is an integer (0 or 1), then set x_{ij} = 0 or 1 and regard it as a constant; else if a variable x_{ij} is $0 < x_{ij} \leq 0.01$ or $0.99 \leq x_{ij} < 1$, then set x_{ij} = 0 or 1, and regard it as a constant;

Step 4.4: If all variables x_{ij} are integer constants (0 or 1), then go to **Step 4.5**; else go to **Step 4.2**;

Step 4.5: If the number of cuts is not improved for a predefined successive time, then terminate; else go to **Step 4.1**;

In **Step 4.3**, if no variable is regarded as an integer constant, we change the variable with a minimum constraint violation into an integer constant (0 or 1). Therefore, we may get a solution with constraint violations. Since we repeat **Step 4.2** $\sim$ **4.4** until all variables become integer constants (0 or 1) and at least one of variables becomes an integer constant in one iteration, the computation time for one loop is $O(T(LP) \cdot |CV|)$ in the worse case, where $T(LP)$ is the computation time of a LP method and $|CV|$ is the number of cluster nodes.

2. Selection of a Maximal Independent Set

As mentioned in 1., the number of cuts can not be represented with a linear expression when moving nodes. If we apply quadratic programming to the partitioning problem, we are able to get a solution for only a small size of problem. Because the number of nodes for the MCM system partitioning problem is very large, it is not practical to get a solution by quadratic programming. Therefore, firstly, we select a maximal independent set so that the number of cuts can be represented with a linear expression exactly. Next, we apply 0-1 ILP to the nodes in the maximal independent set to determine which nodes are to be exchanged to improve a partition. The number of cuts can be represented with a linear expression exactly if only nodes in a maximal independent set were moved, because the nodes in the maximal independent set do not connect with each other. In order to avoid selecting only the same nodes successively and to improve the partition effectively, we introduce the cost function $CF2(v_i)$ of a node v_i defined as

$$CF2(v_i) = degree(v_i) + \gamma \cdot select\_count(v_i),$$

where $degree(v_i)$ is the degree of a node v_i ($v_i \in CV$), $select\_count(v_i)$ shows how many times a node v_i ($v_i \in CV$) is selected as an element of a maximal independent set, and γ is a constant.

In the proposed method, we sort nodes in increasing order of their costs and check if a node is able to be selected for a maximal independent set in that order. Larger the count of a node selected as an element of a maximal independent set becomes, smaller the cost becomes. In our simulation experiments, we set γ to the number of average degree of nodes so as to avoid selecting the nodes repeatedly. Figure 4 shows an example of a selected maximal independent set. The nodes v_2, v_4 and v_8 are selected for a maximal independent set.

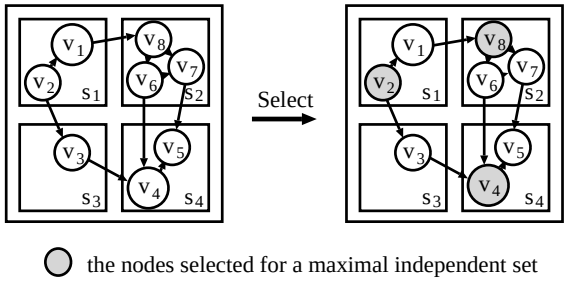


Fig. 4. Selection of a maximal independent set.

3. The 0-1 ILP Problem Formulation

After selecting the nodes in a maximal independent set, we are able to represent the number of cuts with a linear expression exactly. We formulate the partitioning problem in phase 4 based on the 0-1 ILP as follows.

[The 0-1 ILP Formulation of the Partitioning Problem]

Inputs: (1) a MCM System $M = (G, S)$, (2) an initial partitioning $F1 : CN \rightarrow S$

Output: a chip assignment $F2 : CN \rightarrow S$ so as to minimize the objective function

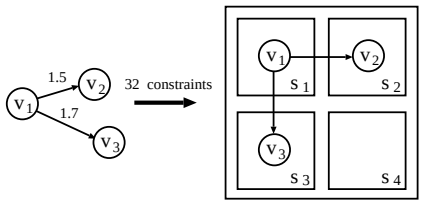
Objective: the number of cuts $\sum_{i=1}^{I'} \sum_{j=1}^{J} oe_{ij} x_{ij}$

Constraints: (1) the timing constraint

$$D_{pq}(x_{lp} + x_{mq} - 1) \leq d_{lm}$$

(2) the I/O pin constraint

$$\sum_{i=1}^{I'} oe_{ij} x_{ij} \leq IO_j \quad (1 \leq j \leq J')$$



(a)

$2(x_{11} + x_{24} - 1) \le 1.5$
$2(x_{11} + x_{34} - 1) \le 1.7$
$2(x_{12} + x_{23} - 1) \le 1.5$
$2(x_{12} + x_{33} - 1) \le 1.7$
$2(x_{13} + x_{22} - 1) \le 1.5$
$2(x_{13} + x_{32} - 1) \le 1.7$
$2(x_{14} + x_{21} - 1) \le 1.5$
$2(x_{14} + x_{31} - 1) \le 1.7$

$x_{11} + x_{24} \le 1$
$x_{11} + x_{34} \le 1$
$x_{12} + x_{23} \le 1$
$x_{12} + x_{33} \le 1$
$x_{13} + x_{22} \le 1$
$x_{13} + x_{32} \le 1$
$x_{14} + x_{21} \le 1$
$x_{14} + x_{31} \le 1$

$2x_{11} + x_{24} + x_{34} \le 2$
$2x_{12} + x_{23} + x_{33} \le 2$
$2x_{13} + x_{22} + x_{32} \le 2$
$2x_{14} + x_{21} + x_{31} \le 2$

(b) 8 constraints (c) 8 constraints (d) 4 constraints

Fig. 5. Reduction of timing constraints.

(3) the chip area constraint

$$\sum_{i=1}^{I'} a_i x_{ij} \le A_j \quad (1 \le j \le J')$$

(4) the chip assignment constraint

$$\sum_{j=1}^{J} x_{ij} = 1 \quad (1 \le i \le I')$$

(5) $x_{ij} = \begin{cases} 1 & F1(n_i) = s_j \\ 0 & otherwise \end{cases}$

4. Reduction of Timing Constraint

The timing constraint is formulated as, $D_{pq}(x_{lp}+x_{mq}-1) \le d_{lm}$ $(1 \le l,m \le I')$ $(1 \le p,q \le J)$, where D_{pq} represents the interconnection delay between chips s_p and s_q, and d_{lm} represents the maximum allowable delay between nodes v_l and v_m. But, if we formulate all timing constraints with the inequality, we must consider all combinations of chips and nodes, and the number of the inequalities of the timing constraints is $I'^2 \times J^2$ in the worst case, where I' and J is the number of nodes and chips, respectively. Even if we construct 100 clusters in phase 2, the number of the inequality of the timing constraints is 640,000 when nodes are assigned to 8 chips! It is impossible to solve such a large size problem in a practical computation time. However we can reduce the number of these constraints, while keeping the mathematical equivalence of original constraints and resulting constraints, because there exist redundant constraints in the original timing constraints.

For example in Figure 5 (a), the maximum allowable delay between nodes v_1 and v_2, and nodes v_1 and v_3 are 1.5 D_{min} and 1.7 D_{min}, respectively. When the nodes are assigned to 4 chips, we need 32 inequalities under the timing constraint. Now, consider the constraints that satisfy $D_{pq}(x_{lp} + x_{mq} - 1) \le d_{lm}$ when $x_{lq} = x_{mq} = 1$. The constraints satisfy $D_{pq}(x_{lp}+x_{mq}-1) \le d_{lm}$ even if x_{lq} and x_{mq} are any value (0 or 1). Therefore we can remove such constraints from the timing constraints. Consequently, the number of the timing constraints becomes 8 as shown in Figure 5 (b). Since these constraints generate the timing violation only when $x_{lq} = x_{mq} = 1$, we are able to convert Figure 5 (b) into (c) keeping this relation. When a source node $v_l \in CN$ is assigned to a chip s_p, let P_m $(1 \le m \le M)$ be the number of chip assignments generating timing violations for every sink node v_m $(1 \le m \le M)$ where M is a number of sink nodes. Then the number of chip assignments generating timing violations, denoted P, becomes $M \times P_m$ for all sink nodes v_m $(1 \le m \le M)$. When we convert these constraints in one constraint, we get the following constraint.

$$P \times x_{lq} + \sum_{m=1}^{M} \sum_{i=1}^{P_m} x_{mq_i} \le P \qquad (2)$$

This new timing constraint can be satisfied *iff* the old timing constraint is satisfied. Finally, we get the result as shown in Figure 5 (d). The number of timing constraints becomes 4, where P = 2. We can derive the following two theorems. Due to the space limitation, we omitted proofs of these theorems. Detail of proofs are found in Ref. [8].

Theorem 1 *When a source node $v_l \in CN$ is assigned to a chip s_p, we presume that the number of chip assignments with timing violations is P for all sink nodes v_m $(1 \le m \le M)$. Then the following constraints,*

$$P \times x_{lq} + \sum_{m=1}^{M} \sum_{i=1}^{P_m} x_{mq_i} \le P \ (1 \le l \le I')(1 \le p,q \le J) \quad (3)$$

are satisfied iff the original constraints,

$$D_{pq}(x_{lp} + x_{mq} - 1) \le d_{lm} \ (1 \le l \le I')(1 \le p,q \le J) \quad (4)$$

are satisfied.

Theorem 2 *The number of the timing constraints of the partitioning problem is at most $I' \times J$, where I' and J are the number of nodes and chips, respectively.*

From these theorems, we are able to reduce the number of timing constraints. For example, when we construct 100 cluster nodes in phase 2 and assign the nodes to 8 chips, the number of timing constraints is not 640,000 but 800!, and we can easily obtain a solution within a practical computation time.

IV Experimental Results

We have implemented the proposed method by C language. All experiments were tested on a SPARC server 1000 (135MIPS, 256Mbytes) of Sun Microsystems, Inc. Table I shows characteristics of the data and the constraints used in experiments. All tested data are MCNC benchmark data. For "primaryGA1" and "primaryGA2", the I/O pin constraint is the same as ones used in Ref. [15]. For "avq.small" and "avq.large", we set the I/O pin constraint to 500. We also set the chip area constraint to $\frac{\sum_{n_i \in N} a_i}{\#chips} \times 1.1$, where a_i and $\#chips$ are the area of node v_i and the number of chips, respectively. In

TABLE I

CHARACTERISTICS OF BENCHMARK DATA.

data	$\#nodes$	$\#nets$	IO_j	A_j	$\#chips$
primaryGA1	833	904	90	3.85	8
primaryGA2	3014	3029	335	7.19	8
avq.small	21854	22081	500	5.77	8
avq.large	25114	25341	500	6.39	8

IO_j : the I/O pin constraint

A_j : the chip area constraint $[mm^2]$

TABLE II

RESULTS OF SHIH AND KUH'S METHOD [14] UNDER THE I/O PIN AND THE CHIP AREA CONSTRAINTS.

data	#cuts	#I/Os	area	#vio	time (s)
primaryGA1	334	90	–	–	–
primaryGA2	992	335	–	–	–

#cuts : the number of cuts

#I/Os : the largest number of I/O pins

area : the largest chip area $[mm^2]$

#vio : the number of timing violations

time : the computation time [s]

other words, every chip may have 10 % extra areas. For "primaryGA1" and "primaryGA2", the maximum allowable delay between flip flops, D_{allow_τ}, is set to the value obtained by multiplying the sum of the internal delays of nodes between flip flops and 11.0. For "avq.small" and "avq.large", we assigned delays ($0 \times D_{min} \sim 5 \times D_{min}$) to pairs of nodes randomly. Note that the delay more than $5 \times D_{min}$ is meaningless in 8-way partitioning since $5 \times D_{min}$ is the delay from one corner to the another corner, that it is the maximum.

We compared the proposed method with the method proposed in Ref. [15]. They partition nodes into eight sets under only two constraints, that is, the I/O pin and the chip area constraints. The I/O pin constraint is set by the value given in Ref. [15]. Table II shows the results described in Ref. [15]. The chip area and the computation time were not shown in Ref. [15]. We partitioned "primaryGA1" and "primaryGA2" into eight chips as the same as in Ref. [15], respectively. Tables III and IV show the results of the proposed method. We set α = 60, β = 0.05, γ = 200,000, and $\#clusters$ = 300 for data "primaryGA1", and α = 90, β = 0.01, γ = 400,000, and $\#clusters$ = 1,000 for data "primaryGA2".

Table III shows the results of the initial partitioning in phase 3 and the final partitioning in phase 4 under two constraints, i.e., the I/O pin and the chip area constraints. In phase 3, the I/O pin constraint is not satisfied, but the chip area constraint is satisfied. In phase 4, the I/O pin and the chip area constraints are satisfied, but the timing constraint is not satisfied since we do not consider the timing constraint in this experiment. Therefore the third term of the cost function $CF1(l, m)$ in phase 2, which represents the criticality of delays, is considered as 0. The number of cuts in phase 4 is improved by a 47 % on an average over phase 3, and the number of the timing violations in phase 4 also decreases over phase 3. The proposed method improves the number of cuts by a 27 % on an average over Ref. [15].

Table IV shows the results of the initial partitioning in phase 3 and the final partitioning in phase 4 under the three constraints, that is, the timing constraints, the I/O pin, and the chip area constraints. In phase 3, only the timing constraint is satisfied for data "primaryGA1", and no constraints are satisfied for data "primaryGA2". The number of cuts in phase 4 is improved by a 41 % on an average over phase 3. In phase 4, all constraints are satisfied. The proposed method can obtain a partition satisfying three constraints with only a 5 % of the number of cuts increasing compared with Ref. [15].

TABLE III

RESULTS OF THE PROPOSED METHOD UNDER THE I/O PIN AND THE CHIP AREA CONSTRAINTS.

data	phase	#cuts	#I/Os	area	#vio	time (s)
primaryGA1	3	390	239	3.78	30	5
	4	233	86	3.81	19	160
primaryGA2	3	1607	816	7.19	279	66
	4	751	325	7.15	182	2038

α = 60, β = 0.05, γ = 200,000, $\#clusters$ = 300

TABLE IV

RESULTS OF THE PROPOSED METHOD UNDER THE THREE CONSTRAINTS.

data	phase	#cuts	#I/Os	area	#vio	time (s)
primaryGA1	3	519	267	7.70	0	7
	4	286	87	3.79	0	318
primaryGA2	3	1942	873	8.10	1	97
	4	1231	333	7.19	0	3968

α = 90, β = 0.01, γ = 400,000, $\#clusters$ = 1,000

Table V shows the results of the proposed method under the two and the three constraints for large data, respectively. The parameters of the cost function in phase 2 are set α = 10, β = 0.05, γ = 100,000, and $\#clusters$ = 1,500 so that the second term of $CF1(l, m)$, which represents the ratio of the average of chip areas to the sum of node areas, becomes the largest among the values of three terms. In Table V, "constraints" represents the constraint that we adopt in experiments, and "IO_j + A_j" and "ALL" represent the two constraints (the chip area and the I/O pin constraints) and three constraints (the timing, the chip area, and the I/O pin constraints), respectively. Note that the results under the three constraints are improved by an 8 % on average over the results under the two constraint against our expectation. This may be explained that initial solutions may be different under the two and the three constraints. The number of cuts in the initial partitioning for data "avq.small" is 3,090 and 3,053 under the two and the three constraints, respectively, and that for data "avq.large" is 3,078 and 3,198 under the two and the three constraints, respectively. As expected, the computation time under the three constraints becomes longer than that under the two constraints.

In Ref. [15], they do not show the chip area constraint. Therefore the chip area constraint might be too loose if every chip has a 10 % extra area. Thus, we set the area constraint A_j as $\frac{\sum_{v_i \in V} a_i}{\#chips} + max_{v_i \in v} a_i$ and experimented so as to show the effectiveness of the proposed method, where a_i, $\#chips$, and $max_{v_i \in v} a_i$ are the node area, the number of chips, and the largest node area, respectively. In other words, each chip has the minimum extra area which is the largest node area. Note that this is the tightest chip area constraint. We can say

TABLE V

THE EXPERIMENTAL RESULTS FOR LARGE DATA UNDER THE THREE CONSTRAINTS.

data	constraints	#cuts	#I/Os	area	#vio	time (s)
avq.small	$IO_j + A_j$	1456	413	5.77	114	11774
	ALL	1230	425	5.76	0	38012
avq.large	$IO_j + A_j$	1264	483	6.39	60	24366
	ALL	1246	429	6.38	0	41133

α = 10, β = 0.05, γ = 100,000, $\#cluster$ = 1,500

$IO_j + A_j$: the I/O pin and the chip area constraints

ALL : the timing, the chip area, and the I/O pin constraints

TABLE VI

RESULTS OF 8-WAY PARTITIONING UNDER THE I/O PIN AND THE CHIP AREA CONSTRAINTS.

data	#cuts	#I/Os	area	#vio	time (s)
primaryGA1	250	86	3.67	21	116
primaryGA2	883	315	6.71	154	3213

that the proposed method is more effective than the method in Ref. [15] if we can get a better solution that the number of cuts is smaller and it satisfies the constraints, under the two constraints. For data "primaryGA1" and "primaryGA2", we set the chip area constraint to 3.68 and 6.72 [mm^2], respectively. For data "primaryGA1" and "primaryGA2", α = 50, β = 0.05, γ = 300,000, and $\#clusters$ = 300, and α = 90, β = 0.05, γ = 500,000, and $\#clusters$ = 1,000 in phase 2, respectively. Table VI shows the results in 8-way partitioning under the I/O pin and the chip area constraints obtained by the proposed method. From Tables II and VI, the proposed method improves the number of cuts by an 18 % over Ref. [15] on an average. The I/O pin and the chip area constraints are satisfied for both methods. Table VII shows the results in 8-way partitioning under the three constraints for proposed method. Although the number of cuts increases by a 3 % over Ref. [15] on an average, the proposed method produces the results satisfying the three constraints within a practical computation time. Those results show that the proposed method is much more effective than the method in Ref. [15].

V CONCLUSIONS

In this paper, we proposed an MCM system partitioning method under the timing, the chip I/O pins, and the chip area constraints. Experimental results show that the proposed method improves the number of cuts by a 27 % on an average over Ref [15]. We showed that the proposed method can get a solution for large size data up to about 20,000 blocks within a practical computation time. As far as we know, the proposed method is the first partitioning method under the three constraints mentioned above. Future research includes the development of an MCM Routing method under performance and physical constraints.

ACKNOWLEDGEMENTS

The authors thank Mr. H. Oohira who has helped in the implementation of programs.

TABLE VII

RESULTS OF THE PROPOSED METHOD FOR 8-WAY PARTITIONING UNDER THE THREE CONSTRAINTS.

data	#cuts	#I/Os	area	#vio	time (s)
primaryGA1	295	84	3.65	0	272
primaryGA2	1172	334	6.72	0	4308

REFERENCES

[1] T. N. Bui and B. R. Moon: "A fast and stable hybrid genetic algorithm for the ratio-cut partitioning problem on hypergraphs," Proc. of ACM/IEEE 31st Design Automation Conference, pp. 664–669 (1994).

[2] D. A. Doane and P. D. Franzon: "Multichip Module Technologies and Alternatives," Van Nostrand Reinhold (1993).

[3] J. Garbers, H. J. Promel and A. Steger: "Finding clusters in VLSI circuits," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 520–523 (1990).

[4] G. L. Ginsberg and D. P. Schnorr: "Multichip Modules and Related Technologies," McGraw-Hill, Inc. (1993).

[5] L. Hagen and A. B. Kahng: "Fast spectral methods for ratio cut partitioning and clustering," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 10–13 (1991).

[6] L. Hagen and A. B. Kahng: "A new approach to effective circuit clustering," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 422–426 (1992).

[7] P. S. Hauge, R. Nair and E. J. Yoffa: "Circuit placement for predictable performance," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 88–91 (1987).

[8] Y. Katsura, T. Koide, S. Wakabayashi and N. Yoshida: "A new system partitioning method under performance and physical constraints for multi-chip modules," Technical Report Tech. Rep. No.95-01, Faculty of Engineering, Hiroshima University (1995).

[9] E. S. Kuh and M. Shih: "Recent advances in timing-driven physical design," Proc. of Asia-Pacific Conference on Circuits and Systems, pp. 23–28 (1992).

[10] E. S. Kuh, M. Shih and R.-S. Tsay: "Performance-driven system partitioning on multi-chip modules," Proc. of ACM/IEEE 29th Design Automation Conference, pp. 53–56 (1992).

[11] E. S. Kuh, M. Shih and R.-S. Tsay: "Integer programming techniques for multi-way system partitioning under timing and capacity constraints," Proc. European Conference on Design Automation with the European Event in ASIC Design, pp. 294–298 (1993).

[12] B. M. Riess, K. Doll and F. M. Johannes: "Partitioning very large circuits using analytical placement techniques," Proc. of ACM/IEEE 31st Design Automation Conference, pp. 646–651 (1994).

[13] K. Roy and C. Sechen: "A timing driven n-way chip and multi-chip partitioner," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 240–247 (1993).

[14] M. Shih and E. S. Kuh: "Quadratic Boolean programming for performance-driven system partitioning," Proc. of ACM/IEEE 30th Design Automation Conference, pp. 761–765 (1993).

[15] M. Shih and E. S. Kuh: "Circuit partitioning under capacity and I/O constraints," Proc. of IEEE Custom Integrated Circuits Conference, pp. 659–662 (1994).

[16] Y.-C. Wei and C.-K. Cheng: "Towards efficient hierarchical designs by ratio cut partitioning," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 298–301 (1989).

[17] Y.-C. Wei and C.-K. Cheng: "Ratio cut partition for hierarchical designs," Proc. of IEEE International Conference on Comput.-Aided Des., 10, 7, pp. 911–921 (1991).

[18] N.-S. Woo and J. Kim: "An efficient method of partitioning circuits for multiple-FPGA implementation," Proc. of ACM/IEEE 30th Design Automation Conference, pp. 202–207 (1993).

[19] C.-W. Yeh, C.-K. Cheng and T.-T. Y. Lin: "A probabilistic multicommodity-flow solution to circuit clustering problems," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 428–431 (1992).