論文99-36C-9-4

다중칩 모듈 설계를 위한 Gridless 배선기

(A Gridless Area Router for Multichip Module Design)

李 台 先 *, 林 種 錫 *

(Tae-Sun Lee and Chong suck Rim)

요 약

본 논문에서는 다중칩 모듈 설계를 위한 gridless 배선 방법을 제안한다. 제안하는 배선 방법에서는 배선 상태를 표현하기 위해 배선격자 (grid) 대신 corner stitching 자료구조를 사용함으로써 핀들의 위치 제한을 없애고, 네트의 특성에 맞는 와이어 폭을 선택할 수 있다. 또한 비아보다 그 폭이 작은 와이어들로 인해 남는 공간을 다른 네트의 배선에 이용하여 배선영역을 최대한 낭비 없이 이용한다. 배선격자를 사용하지 않는 배선 방법은 배선격자를 기반으로 하는 배선 방법에 비해 복잡하고 어렵다. 더구나 다중칩 모듈과 같이 배선 영역이 크고 배선층의 갯수가 많은 배선 문제일 경우 배선 속도는 배선 가능성을 결정하는 중요한 요소가 된다. 그러나 제안하는 배선 방법은 빠른 속도가 증명된 SEGRA의 배선 알고리즘과 효율적인 자료구조의 특성을 적절히 이용함으로써 배선격자를 기반으로 하는 배선 방법 보다 빠른 속도와 그리 떨어지지 않는 배선 결과를 보인다.

Abstract

In this paper, we present a gridless router for MultiChip Modules (MCM). Because our router uses corner stitching data structures, not a routing grid, to represent the routing status, it allows arbitrary location of pins, and routes variable-width wires, without a considerable waste of area from bulky vias. A routing speed is a very important factor because a gridless routing approach is known its computation is hard and complex, and MCM routing problem has so large routing area and layers. Our router completes the routing faster than the most of previously reported grid-based routers, with comparable routing result, by using SEGRAs routing algorithm whose very fast speed is proved, and the characteristics of the effective data structure.

I. 서 론

반도체 제조 기술의 발전으로 소자 (device)의 크기가 매우 작아졌을 뿐 아니라 소자 간의 최소 유지 거리도 가까워져, 그 결과로 집적회로 (IC) 내에서의 연결지연 (interconnection delay)이 매우 줄었다. 그러나 이런 성능 향상이 시스템 전체의 성능 향상으로 이어지지 않고 있는데, 이것은 P/I (Packaging and Interconnection) 기술의 발전 속도가 반도체 제조 기술의 발전 속도를 따라가고 있지 못하기 때문이다[3,9,15]. 이러한 P/I에서 비롯되는 한계를 줄여 시스템 성능을 높이고자 하는 노력 중 하나가 다중칩 모듈 (MultiChip Module) 기술이다. 이것은 여러 층 (layer)으로 구성된 기판 (substrate)에 패키징되지 않은 칩들을 2개 이상 올려 놓고 기판을 통해 칩들을 연결한 후 하나로 패키징하는 기술이다. 칩들을 개별적으로 패키징하지 않으므로 기판에 보다 가깝게 놓여질 수 있고 연결에 있어

* 正會員, 西江大學校 컴퓨터學科
(Dept. of Computer Science and Engineering, Sogang University)

※ 이 연구는 1999년도 서강대학교 교내 연구비 지원에 의하여 이루어졌음

接受日字 : 1999年1月18日, 수정완료일 : 1999年8月6日

서도 한 단계가 줄기 때문에, 패키징 밀도는 높아지고 연결 지연은 작아진다[6, 16].

이렇듯 많은 칩이 작은 공간을 사용하여 하나로 패키징 된다는 특징으로 인해 MCM 기술이 고성능 시스템 설계에 많이 사용되지만, 같은 이유로 IC나 인쇄배선기판 (Printed Circuit Board)에 비해 그 배선이 매우 어렵고 복잡한 것으로 알려져 있다. 이것은 IC에 비해 훨씬 많은 수의 배선층이 사용되고, 배선영역이 매우 넓고 3차원 영역 전체를 직접 사용해야 하기 때문이다. 또한 PCB보다 훨씬 조밀하게 배선하여야 하므로 기존의 PCB배선 툴이 사용될 수 없다[10-12]. 이와 같이 MCM 배선 문제는 기존의 IC나 PCB의 배선 문제보다 어렵고, 기존의 배선기로는 적용하기 힘든 형태를 갖고 있기 때문에, 적합한 형태의 배선기 개발이 필요하다.

이러한 MCM 배선 문제를 해결하기 위해 3차원 미로 배선 방법(3-D maze routing)[8,13], SURF[5], SLICE[11], V4R[12], MCG[7], SEGRA[4]와 같은 알고리즘들이 제안되었다. 이중 비교적 최근에 발표된 MCG는 cross-talk을 고려하면서 비교적 적은 배선층을 사용하고, SEGRA는 V4R과 비슷한 성능을 보이면서도 매우 빠른 속도로 배선한다.

이상의 MCM 배선 알고리즘들은 대부분 배선격자를 기반으로 배선한다. 배선격자를 기반으로 하는 배선 모델은 배선영역이 맨하탄 격자로 이루어지고, 반드시 그 격자의 선을 따라 배선할 것을 가정한다. 배선격자를 가정한 접근 방법은 계산을 쉽게 하지만 다음과 같은 제약을 갖는다[16].

1. 배선격자의 피치가 worst-case spacing rule로 결정되므로 배선영역의 낭비가 많다.
2. 와이어 폭이 일률적으로 부여되어, 네트 특성에 따라 와이어 폭을 선택할 수 없다.
3. 핀들이 격자점 위에 놓여야 하므로 칩의 크기가 커지고, 설계가 어려워질 수 있다.

보통 비아 피치가 와이어 피치보다 크고, 모든 격자점에서 비아를 만들 수 있어야 하므로 비아 피치가 격자의 크기를 결정한다. 그런데 표1에서 알 수 있듯이 MCM 제조 기술에 따라 비아의 최소 피치가 와이어의 최소 피치에 비해 아주 클 수 있다[6]. 배선격자에 기반하여 배선한다면 비아와 와이어 폭의 크기 차가 클 경우 상대적으로 작은 와이어 폭에 의해 낭비되는 공간의 면적이 커진다. 만일 설계규칙 (design rule)만을 고려하여 배선한다면 그림 1과 같이 공간의 낭비 없이 배선할 수 있다.

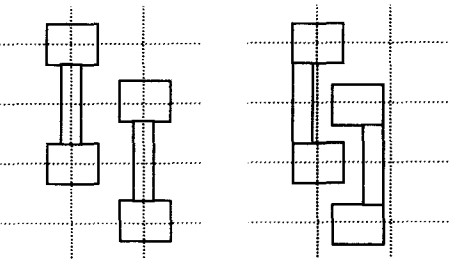

그림 1. 배선격자기반 배선과 GRIDLESS 배선
Fig. 1. GRID-BASED vs. GRIDLESS.

또한 요즘처럼 높은 주파수로 동작할 것이 요구되는 상황에서 클락과 같이 많은 전류를 흘려 보 내야 하는 네트들은 다른 네트보다 더 굵은 와이어가 요구된다. 그러나 배선격자를 기반으로 하는 배선기들은 일률적인 와이어 폭만을 사용한다. 만일 더 넓어진 와이어 피치가 배선격자의 피치를 초과한다면, 배선격자를 그 와이어 피치에 맞추어 다시 배선하거나 와이어의 최대 폭에 한계를 주어야 한다. 따라서 네트별로 다른 와이어의 폭을 부여할 수 있으면서도 가능한 조밀하게 배선할 수 있는 배선방법이 요구된다.

배선격자를 기반으로 하는 배선기는 핀들이 격자점 위에 놓여져야 배선이 가능한데, 이러한 제약은 칩의 설계를 어렵게 할 뿐 아니라 칩의 크기를 필요 이상으로 크게 하는 원인이 된다. 더구나 과거에 설계된 칩은 현재의 기술로 결정된 피치에 맞지 않아 다시 설계하거나 피치에 맞도록 칩의 크기를 크게 해야 할 가능성이 높다. 만일 배선기가 핀의 위치에 관계 없이 배선할 수 있다면 설계시의 어려움을 크게 덜 수 있을 것이고 칩이 필요 이상으로 커지는 일도 막을 수 있다.

이러한 제약들로부터 완전히 자유로워지기 위해서는 배선격자를 포기하고 연속적인 공간을 배선영역으로 하여야 한다. 그러나 배선격자가 계산을 보다 쉽고 빠르게 하기 위해 사용되었던 만큼 배선격자를 사용하지 않는 배선은 그 계산이 복잡하고 어렵다. 더구나 다중칩 모듈과 같이 배선영역이 크고 배선층의 수가 많은 배선 문제일 경우 배선 속도는 배선 가능성을 결정하는 중요한 요소가 된다.

표 1. 일반적인 최소 라인 피치
Table 1. Typical Minimum Line Pitches.

TECHNOLOGY	TECHNOLOGY	TECHNOLOGY
Cofired Ceramic MCM	150 μm-250 μm	250 μm
Thin film MCM	20 μm-40 μm	20 μm-200 μm
Laminate MCM	100 μm-250 μm	1000 μm-2500 μm
PCB	250 μm-300 μm	1000 μm-2500 μm

본 논문에서는 배선격자를 사용하지 않고 다중칩 모듈에서의 배선에 사용할 수 있는 배선 방법을 제안한다. 제안하는 배선 방법은 매우 빠른 속도를 보인 SEGRA의 배선 알고리즘을 응용함으로써 비교적 빠른 속도로 배선한다. 배선격자를 사용하지 않고 배선하므로 배선격자를 기반으로 배선하는 배선방법들이 갖는 제약들로부터 자유로우면서 기존에 발표된 배선 방법에 비해 그리 떨어지지 않는 배선 결과를 보인다. 앞으로 이 논문에서는 제안하는 배선기를 GMR (Gridless MCM Router)이라 부르기로 한다.

서론에 이어 2장에서는 먼저 SEGRA의 배선 알고리즘을 개략적으로 설명하고, 제안하는 배선 방법을 SEGRA와 비교하여 설명한다. 3장에서는 사용하는 자료구조와 기본 모듈에 대해 설명하고, 4장에서 배선 알고리즘에 대해 SEGRA와 다른 점을 중심으로 자세히 설명한다. 실험결과와 기존의 결과와의 비교를 5장에서 보이고, 6장에서 결론을 맺는다.

II. SEGRA 와 GMR의 알고리즘 개요

1. SEGRA[4]

SEGRA는 위에서부터 아래로 한 번에 2개의 배선층을 선택하여, 하나는 수평 방향, 다른 하나는 수직 방향의 배선에 사용한다. 현재 배선중인 층에서 최대한 많은 네트를 배선하고 배선에 성공하지 못한 네트들은 다음 배선 층으로 전달하여 배선을 수행하는데, 이런 과정을 모든 네트가 배선을 마칠 때까지 반복한다.

각 배선 층에서는 배선 영역 좌우 양끝에 두 개의 수직선을 설정하고 이들을 중심을 향해 수평으로 진행시키면서 각 네트의 배선을 수행한다. 이때 왼쪽에서 오른쪽으로 이동하는 진행선을 *좌진행선* 이라 하고, 반대로 오른쪽에서 왼쪽으로 이동하는 진행선을 *우진행선* 이라 한다. 진행선들이 각각 서로를 향해 수평으로 배선 격자를 따라 이동하는 중 배선이 완료되지 않은 터미널을 만나면 그 터미널로부터 *진행점*이 만들어지고, 이 진행점은 연결해야 할 터미널인 *목표 터미널*을 향해, 또는 목표 터미널로부터 만들어진 진행점을 향해 진행선과 함께 이동한다. 만일 진행점의 진행 방향으로 가까운 거리 내에 장애물이 있거나 목표점과 만나기에 적합한 수평트랙을 발견하면 수직으로 이동하는데 이를 진행점의 *수직이동* 이라 한다.

진행선과 함께 이동하고 있는 진행점들 가운데 어느 것이 먼저 수직이동을 시도하느냐를 결정하기 위해 진행점들의 진행 방향으로 가장 가까운 장애물까지의 거리에 따라 1부터 6까지 모두 여섯 가지의 *우선순위*를 부여하고, 높은 우선순위를 갖는 진행점부터 먼저 수직이동을 시도한다. 수직이동은 몇 가지 규칙에 의해 이루어지며, 한번 수직이동한 진행점은 바로 수평으로 연결될 수 있는 위치가 아니면 수직이동을 하지 못하도록 함으로써 가능한 비아의 개수가 적은 모양의 배선이 이루어질 수 있도록 한다. 만일 진행점이 수직이동에 실패하고 장애물로 인해 더 이상 수평으로 이동하는 것이 불가능해지거나, 좌우 진행선이 서로 만날 때까지 배선을 완료하지 못하면 지금까지의 중간 배선 결과는 무시되고 다음 배선 층에서 처음부터 다시 배선을 수행한다.

배선 알고리즘에서 가장 빈번히 수행되는 작업은 수직 또는 수평으로 주어진 구간 내에 주어진 방향으로 가장 가까운 장애물을 찾아내는 작업이다. 이것은 전체 배선 속도에 큰 영향으로 미치는 요소이다. SEGRA는 이러한 작업을 효율적으로 수행하기 위하여 알고리즘의 초기에 모든 터미널들을 그들의 위치에 따라 정렬하여 2개의 배열에 저장하는데, 하나는 행 우선으로 정렬된 배열이고, 다른 하나는 열 우선으로 정렬된 배열이다. 이들 두 터미널 배열의 임의의 열이나 행으로의 접근을 수행하기 위해 그림 2와 같이 각 열과 행의 처음을 가리키는 *열 배열 (column array)*과 *행 배열 (row array)*을 사용한다.

이러한 효율적인 자료구조와 우선순위 체계를 사용함으로써 SEGRA는 배선 속도가 매우 빠르고, 네트의 배선 형태에 제한을 두어 불필요한 비아 사용을 최대한 억제함으로써 우수한 배선 결과를 얻는다. 그러나 SEGRA 역시 서론에서 설명한 배선 격자 기반 배선 방법이 갖는 제약을 갖는다. 또한 비아 사용을 가능한

억제하기 위해 배선 형태를 엄격히 제한하므로 현재 배선 층에서의 배선을 포기하게 되는 네트들이 많다. 이것은 결과적으로 배선영역의 낭비로 이어질 수 있다.

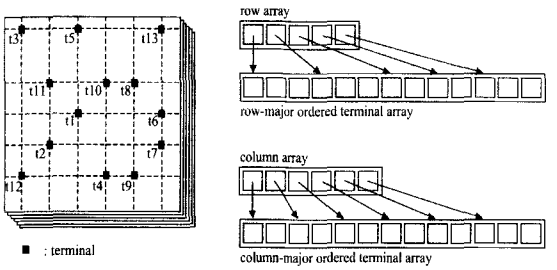

그림 2. SEGRA의 자료구조
Fig. 2. Data Structures used in SEGRA.

SEGRA는 배선 알고리즘을 적용하기 전에 3개 이상의 터미널을 갖는 다중 터미널 (multi-terminal) 네트에 대해서는 Prim의 최소 스패닝 트리 (minimum spanning tree)[1]를 구한 후 두 개의 터미널을 갖는 네트로 분할하여 재구성한다. 따라서 어떤 터미널에 대해서는 그림 3과 같이 목표 터미널이 2개 이상이 될 수 있다. SEGRA는 목표 터미널이 2개 이상인 터미널로부터 진행점을 만들 때 그 중 하나만을 목표로 하게 하고, 목표점과의 배선이 완료되면 그 다음 목표 터미널을 목표로 하게 하는 방식으로 배선하는데, 이러한 방식은 배선 길이가 길어지게 할 가능성이 높다.

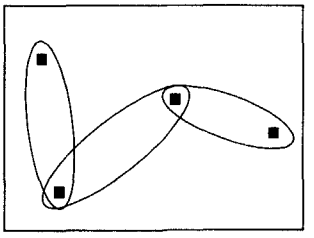

그림 3. 다중터미널이 2-터미널로 재구성된 예
Fig. 3. The decomposing example of a multi-terminal net to two-terminal nets.

2. GMR

GMR은 배선격자를 사용하지 않고 SEGRA의 배선 알고리즘을 적용함으로써 배선 격자의 사용으로 인한 여러 가지 제약을 피하면서 빠르게 배선 하고, 또한 SEGRA보다는 배선 형태의 제한을 비교적 완화함으로써 가능한 주어진 배선 층에서 많은 네트가 배선 될 수 있도록 하는 것을 목표로 한다. 따라서 배선 알고리즘의 기본 생각은 SEGRA와 같으나 다음과 같은 몇 가지 면에서 SEGRA와 다른 배선 전략이 요구된다.

우선 제안하는 배선 방법은 설계규칙에 의존하여 배선하므로 제조 기술에 맞는 설계규칙이 입력되어야 한다. 설계규칙은 파일로 입력 받는다. 첫 줄에 모든 네트에 적용할 비아 크기, 와이어 폭, 다른 네트와 유지되어야 할 최소 간격을 지정하고, 다른 비아 크기, 와이어 폭, 최소 간격을 지정해야 하는 네트가 있으면 그 다음 줄에 네트 번호와 함께 값을 지정한다. 단 비아의 모양은 정사각형이고, 층별 설계규칙의 차이는 없다고 가정한다.

배선 격자 기반 배선 방법들은 와이어는 격자선에, 터미널이나 비아는 격자점에 놓인다고 가정한다. 그러나 GMR과 같은 gridless 배선 방법에서 와이어나 터미널, 비아는 실제적인 배선 형태에 보다 가까운 블록(block)들의 집합으로 표현된다. GMR이 목표로 하는 배선 결과는 이러한 블록들이 설계규칙에 어긋나지 않으면서 최대한 가깝게 놓이도록 하는 것이다. GMR은 이를 위해 새로운 블록을 만들 때 이미 만들어진 블록이 장애물이 되면 설계규칙이 허용하는 한도 내에서 장애물이 되는 블록을 간신히 피하는 위치에 블록을 만드는 전략을 사용한다. 이것은 GMR이 배선격자를 대신할 수 있도록 사용하는 기본적인 전략으로 진행선의 다음 위치를 결정할 때, 진행점이 수직이동할 다음 위치를 선택할 때, 그리고 수직, 수평와이어에 대한 비아의 위치를 결정할 때 사용된다. 이에 대해 4장에서 자세히 설명한다.

GRM의 또 다른 배선 목표 중 하나가 SEGRA보다는 배선 형태의 제한을 비교적 완화함으로써 가능한 주어진 배선 층에서 많은 네트가 배선 될 수 있도록 하는 것이다. GMR은 이를 위해 SEGRA의 배선 방법과 몇 가지 면에서 다른 방법을 사용한다. 먼저 다중 터미널의 경우 목표 터미널의 수 만큼 진행점을 만들어 각각에게 다른 목표를 부여한다. 이 진행점들은 처음엔 같은 와이어를 공유하며 함께 진행하다가 각각의 목표와 만날 수 있는 위치를 찾아 각각 수직이동한다. 이러한 방법은 하나의 네트가 차지하는 배선영역을 줄이고, 따라서 주어진 배선영역에서 보다 많은 네트가 배선이 이루어지도록 한다. 또한 SEGRA에서 수직이동의 기회가 주어진 진행점은 주어진 구간 내에 수직이동에 적합한 위치가 있으면 다른 네트에 관여치 않고 그 위치

로 이동한다. 그 진행점의 수직이동이 이루어진 구간 내에 다른 진행점들이 있다면 그 진행점들에게 현재 진행선의 위치에서의 수직이동 기회는 없다. GMR에서도 기본적으로는 같은 방법을 취한다. 그러나 장애물이 매우 가까이에 있어 빨리 수직이동하지 않으면 현재의 층에서의 배선을 포기하여야 하는 진행점들에 대해서는 가능한 다른 진행점들의 수직이동으로 인해 수직이동의 기회를 잃지 않도록 수직이동이 가능한 구간을 제한한다. 또한 장애물까지의 거리 구분을 보다 세분화하여 보다 위급한 진행점들이 빨리 수직이동을 할 수 있도록 우선 순위 체계를 7가지로 늘려서 사용하다.

SEGRA의 배선 속도를 빠르게 하는 것은 무엇보다 효율적인 자료구조의 사용으로, 주어진 구간 내에서 주어진 방향으로 가장 가까운 블록을 찾는 작업이 매우 빠르게 이루어질 수 있도록 한다. 이 자료구조는 배선 격자를 기반으로 구성된 배열의 형태이므로 GMR에서 사용할 수 없다. 따라서 GMR에서는 빠른 배선을 가능하게 하는 새로운 자료구조의 도입이 절대적으로 필요하다.

Ⅲ. Gridless 배선을 위한 자료구조

1. Corner Stitching 자료구조

네트마다 다른 크기의 비아와 와이어 폭을 2차원의 연속적인 공간에 적절하게 표현할 수 있으며 그 위치 제한이 없고, 배선 속도에 큰 영향을 미치는 장애물 찾기 작업이 매우 빠르게 이루어져야 한다는 조건을 만족시키기 위해 우리의 배선기는 corner stitching 자료구조를 기본으로 사용한다[14]. 이 자료구조는 배선상태를 여러 개의 타일 (tile)들로 표현하는데, 사용되고 있는 영역 (blocked tile)과 비어 있는 영역 (vacant tile)을 동시에 나타내고, 주어진 타일에 이웃하는 타일을 찾는데 효율적이다. 각 타일은 그림 4에서 보듯이 직사각형의 왼쪽 위 좌표에 해당하는 (x_1, y_1), 오른쪽 아래 좌표에 해당하는 (x_2, y_2)와, rt, tr, bl, lb의 4개의 포인터를 갖는데, 이 포인터들을 corner stitches라고 부르고, 모든 타일들은 이 포인터들로 연결된다.

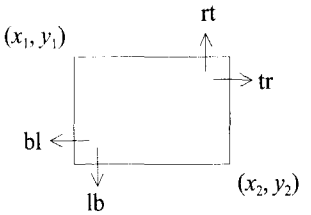

그림 4. Corner Stitches
Fig. 4. Corner Stitches.

그림 5는 타일들이 어떻게 연결되는지를 보여주는 한 예이다. 회색 타일은 block tile을, 흰색 타일은 vacant tile을 나타낸다. 타일 A의 rt 포인터는 타일 B를 가리키고, tr 포인터는 타일 C를, bl 포인터는 타일 F를, lb 포인터는 타일 E를 가리킨다.

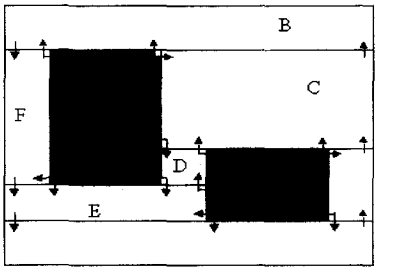

그림 5. Corner Stitches 자료구조를 사용한 예
Fig. 5. An example of corner stitching data structure.

Corner stitching 자료구조는 vacant tile을 나누어 나가는 방향에 따라 *maximal horizontal strip (H)* 모델과, *maximal vertical strip (V)* 모델 두 종류로 구분한다[14,15]. 그림 5는 H모델로 구성된 경우이다. H모델은 수평방향의 블록 찾기에, 그리고 V모델은 수직방향의 블록 찾기에 유리하다. 또한 상대적으로 수직으로 긴 모양의 블록의 수가 많을 때는 V모델의 경우가 H모델보다 적은 수의 타일을 사용한다. 따라서 우리의 배선기는 수평와이어가 주로 놓이는 배선층은 H모델, 수직와이어를 위한 배선층은 V모델에 기반하여 구성함으로써 배선 속도와 메모리 사용이 최소가 되도록 한다.

실제로 본 논문에서 사용되는 실험 데이터 중 하나로 3385×3385의 격자와 14659개의 핀을 갖는 것이 있는데, 격자 기반 배선 방법과 같은 조건이 되도록 설계규칙을 부여하여 배선한 결과 초기의 타일 수는 62656개, 배선 중 타일의 수가 가장 많았을 때도 63645개에 불과했다. 이것은 2개의 배선층을 구성하는데 사용된 타일들을 모두 합한 것으로, 총 핀의 개수의 약 4배에 해당하는 수이다. 이때 타일 개수의 초기값과 최대값의 차가 적은 것은 이미 배선이 끝난 진행선의 왼쪽 부분의 배선 결과를 기록한 후 블록 타일들을 지우기 때문

이다. SEGRA를 제외한 어떤 배선 방법보다 빠른 속도를 보임으로써 corner stitching자료구조가 우리 알고리즘에 적합함을 알 수 있다.

2. 기본 작업

제안하는 알고리즘에서는

블록 찾기 : 최대한 길게 배선할 수 있는 수직, 수평와이어의 길이를 계산하기 위해 주어진 방향으로 가장 가까이에 놓인 블록 타일을 찾아내는 작업과,

영역 탐색 : 비아를 만들 수 있는지를 확인하기 위해 주어진 영역에 장애물, 즉 블록 타일이 있는지를 확인하는 작업

이 수시로 요구된다.

블록 찾기는 수직 방향 블록 찾기와 수평 방향 블록 찾기로 나눌 수 있다. 수직 방향 블록 찾기는 수직배선층에서, 수평 방향 블록 찾기는 수평배선층에서 이루어진다. 블록 찾기 작업은 배선할 와이어 폭에 유지되어야 할 최소 거리를 감안한 범위 내에서 이루어진다. 그림 6에서 터미널 블록 T의 네트 번호가 n이고, 네트 n의 와이어 폭이 ω_n, 그리고 다른 네트와 유지해야 할 최소 거리가 μ_n일 때, v좌표를 중심으로 수평으로 배선하는데 있어서 오른쪽으로 가장 가까운 장애물을 찾는다고 하면, 수평 블록 찾기 모듈은 터미널 블록 T보다 오른쪽에 있으면서 수직구간 $[v-\omega_n/2-\mu_n, v+\omega_n/2+\mu_n]$내에 있는 블록들을 찾는다. 즉, 회색의 직사각형이 수평와이어가 만들어질 영역이고, 검은색 점선이 수평와이어 블록에 대해 다른 네트와 유지되어야 할 최소한의 거리를 감안한 영역이므로, 터미널 블록 타일 T로부터 v좌표를 중심으로 오른쪽으로 수평 배선하는데 있어서 장애물이 되는 것은 검은색 점선 안에 있는 블록 타일들이다. 마찬가지로 수직 방향 블록 찾기에서도 배선 될 수직와이어 블록의 좌변에서 μ_n을 뺀 좌표로부터 우변에 μ_n을 더한 좌표까지를 대상구간으로 하여 주어진 수직 방향에 있는 블록들을 찾는다.

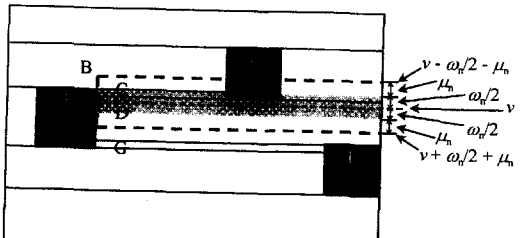

그림 6. 수평 방향 블록 찾기가 이루어지는 범위
Fig. 6. The search range for blocks in horizontal direction.

v좌표를 중심으로 오른쪽으로 수평 배선하는데 있어서 장애물은 타일 E이다. 그러나 만일 그림 7과 같이 v'좌표를 중심으로 배선한다면 타일 F가 장애물 타일이 되고, 타일 F까지의 거리가 타일 E까지의 거리보다 더 크다. 이와 같이 수평와이어 블록이 만들어질 중심 좌표의 위치에 따라 수평으로 배선 가능한 거리가 달라질 수 있다.

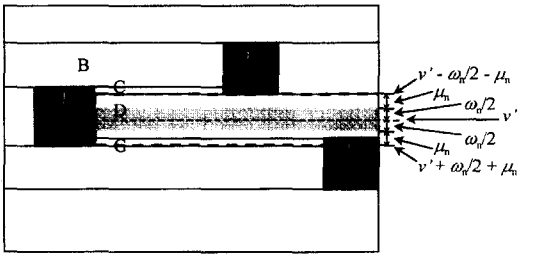

그림 7. 진행점의 y좌표 계산하는 예
Fig. 7. An example of calculating the y-coordinate of active point.

GMR은 터미널을 진행점으로 포함할 때 진행점의 y좌표를 결정한다. 이때 모든 수평와이어는 진행점의 y좌표를 중심으로 하여 만들어지므로 가능한 가장 멀리까지 배선할 수 있는 좌표를 진행점의 y좌표로 하는 것이 더 유리하다. 따라서 다음과 같이 수평와이어의 중심좌표로서 가능한 좌표들을 모두 찾아 그 중 가장 멀리까지 수평 배선할 수 있는 좌표를 진행점의 y좌표로 한다.

터미널 블록 T로부터 설계규칙을 어기지 않고 배선 가능한 수평와이어의 중심 좌표의 범위는 $[T\cdot y_1+\omega_n/2, T\cdot y_2-\omega_n/2]$이다. 진행점의 y좌표의 초기값은 구간의 최소값인 $(T\cdot y_1+\omega_n/2)$으로 한다. 그림 6을 보면 v좌표가 진행점의 y좌표의 초기값에 해당하고, 수평 블록 찾기의 대상이 되는 수직구간의 y_1좌표가 가장 가까운 장애물인 타일 E의 y_2좌표와 같아지도록 내려오기 전까지는 가장 가까운 장애물이 계속 타일 E임을 알 수 있다. 따라서 타일 E가 아닌 다른 타일이 가장 가까운 장애물이 되도록 하기 위해서는 수평와이

어의 중심좌표가 $(E \cdot y_2 + (E \cdot y_2 + \omega_n + 2\mu_n))/2$ 이상인 값을 가져야 한다. 좌표 v를 중심으로 수평으로 배선할 때의 가장 가까운 장애물인 타일 E를 피할 수 있는 수평와이어의 중심좌표인 이 값을 v'라 하고, 다시 장애물 타일 F를 피하도록 수평 블록 찾기의 대상이 되는 수직구간을 아래로 내린다. 이때 수평와이어 블록의 중심좌표가 배선 가능한 수평와이어의 중심 좌표 범위를 벗어나게 되므로 GMR은 터미널 블록 T로부터 오른쪽으로 배선 되는 수평와이어의 가능한 중심좌표로서 찾아진 v와 v' 가운데 장애물까지의 거리가 더 먼 v'을 터미널 T로부터 만들어지는 진행점의 y 좌표로 한다.

영역탐색은 주어진 영역 내에 블록 타일이 있는지를 확인하는 작업이다. 배선의 결과로 새로운 블록을 만들 때 설계규칙에 어긋나지 않음을 확인하기 위해 블록이 만들어질 위치에 최소 유지 거리가 감안된 영역 내에 다른 블록은 없는지를 확인하여야 한다. 영역탐색 작업도 블록에 대한 최소유지거리를 감안한 범위를 대상으로 이루어진다. 즉 만들어질 블록의 왼쪽 위 좌표가 (x_1, y_1), 오른쪽 아래 좌표가 (x_2, y_2)일 때, 블록이 속한 네트의 번호가 n이고, 다른 네트의 블록과 유지해야 할 최소 거리를 μ_n이라 하면, 블록 찾기나 영역탐색 작업이 이루어질 범위는 $(x_1 - \mu_n, y_1 - \mu_n)$부터 $(x_2 - \mu_n, y_2 - \mu_n)$에 이르는 직사각형 영역이다. 본 논문에서는 이 직사각형 영역을 *대상영역*이라 부르고, $x_1 - \mu_n, y_1 - \mu_n, x_2 + \mu_n, y_2 + \mu_n$에 해당하는 좌표를 각각 X_1, Y_1, X_2, Y_2로 표현하기로 한다.

3. 연결 리스트 자료구조

진행선의 다음 위치 계산을 위해서는 블록 타일들의 순차적 검색이 필요하다. Corner stitching 자료구조만으로 블록 타일들만을, 그것도 x_1, x_2 좌표의 오름차순으로 접근하는 일은 쉽지 않다. 또한 vacant 타일과는 달리 블록들은 각각이 속한 네트에 대한 많은 정보를 갖고 있어야 한다. 따라서 우리의 배선 알고리즘은 보다 효율적인 블록 검색과 메모리 관리를 위해 블록 타일에 대한 정보만으로 구성된 연결 리스트 (linked list)를 사용한다. 연결 리스트를 구성하는 노드들은 해당하는 블록 타일의 주소와 블록이 속한 네트의 정보, 블록 타일의 x_1, x_2 좌표값을 기준으로 정렬하기 위한 포인터를 갖는다.

Ⅳ. 배선 알고리즘

1. 진행선의 구성 및 이동

진행선은 배선영역의 맨 왼쪽 끝에서 만들어져 오른쪽 끝으로 이동하면서 배선이 완료되지 않은 터미널들을 만나면 그 터미널들에 대해 각각 진행점을 만들어 진행선에 포함한다. 단, 진행선이 맨 왼쪽 끝에 있을 때는 주어진 거리 내에 있으면서 왼쪽 방향으로 배선하는데 있어서 아무런 장애물이 없는 터미널에 한하여 진행점으로 포함시키는 것을 허용한다. 이것은 특정 터미널들의 우회(detour)를 허용하는 것으로, 왼쪽 끝의 비어 있는 영역을 배선 영역으로 활용하여 배선율을 높이고자 하는 것이다. 또한 재구성된 다중터미널 중 목표터미널이 2개 이상인 터미널에 대해서는 목표 터미널의 수 만큼 진행점을 만들어, 같은 비아와 와이어를 공유하며 최대한 함께 진행하다가 각자의 목표 터미널들을 향해 수직이동하도록 한다. 이 연결방법은 다중 터미널들이 완전하지는 않으나 스타이너 트리(steiner tree)에 가까운 배선형태가 되도록 한다.

주어진 위치에서 진행점이 수직이동을 하면 그 결과로 수직와이어와 비아가 만들어지는데, 진행선은 이러한 블록들의 위치를 결정하는 기준이 된다. 특히 수직와이어는 수직와이어 블록의 x_1 좌표가 진행선의 x 좌표에 일치한다는 가정하에 수직이동이 시도된다. 만일 현재 진행선의 위치에서 어떤 진행점도 설계규칙을 어기지 않고 수직와이어를 배선할 수 없는데 진행점들이 수직이동을 시도한다면 시간의 낭비를 초래하여 전체 배선 시간이 길어진다. 반대로 진행선의 이동 간격이 너무 커 수직와이어들간의 거리가 네트간 유지되어야 할 최소거리보다 크다면 배선공간이 낭비되었다고 할 수 있다. 따라서 진행선의 위치는 이미 만들어진 수직 배선층의 블록들에 최대한 가까우면서, 하나 이상의 진행점에 대해 수직이동이 가능해야 한다.

이러한 진행선의 특성과 역할이 다음 위치 선정에 감안 되어져야 한다. 즉, 빠뜨리고 지나치는 터미널이 없어야 하며, 진행선의 위치에서 최대한 수행시간과 배선영역을 낭비하지 않으면서 수직 와이어와 비아가 만들어질 수 있어야 한다. 따라서 다음의 C_1, C_2를 먼저 구한 후 둘 중 작은 값을 진행선의 다음 위치로 한다.

C_1 : 진행선에서 가장 가까운 터미널 블록 타일의 x_1

좌표.

C_2 : 수직배선층의 블록 타일들 중 x_2 좌표와 해당하는 네트의 최소 유지 거리와의 합이 진행선의 x 좌표보다 큰 값들 중 가장 작은 값.

그림 8의 (a)에서 진행선의 다음 위치를 구할 때 $C1$ 은 아직 진행점이 되지 않은 터미널들의 x_1 좌표값들 중 최소값이고, $C2$ 는 수직배선층의 블록 타일들 중 x_2 좌표와 해당하는 네트의 최소 유지 거리와의 합이 진행선의 x 좌표보다 큰 값들 중 가장 작은 값이다. $C2'$ 은 현재의 진행선의 x 좌표보다 그 값이 작으므로 고려하지 않는다. $C2$ 와 $C1$ 을 비교하면 $C2$ 가 그 값이 작으므로 $C2$ 가 진행선의 다음 위치가 된다. 그림 8의 (b)는 (a)에서 결정된 위치로 이동하여 수직이동 모듈이 수행된 후 다시 진행선의 다음 위치를 계산하는 그림이다. (b)의 $C1$ 은 (a)에서와 같다. 그러나 $C2$ 는 (a)에서 $C2$ 의 기준이 되었던 블록보다 x_2 좌표가 큰 블록, 즉 진행선의 현재 위치에서 새로 만들어진 수직와이어를 기준으로 다시 계산된다. 이번에는 $C1$ 이 $C2$ 보다 작기 때문에 진행선의 다음 위치는 $C1$ 이 되고, $C1$ 의 기준이 되었던 터미널을 진행점으로 포함한다.

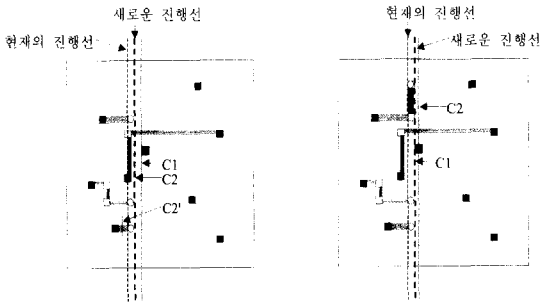

그림 8. 진행선의 다음 위치 계산의 예
Fig. 8. An example of calculating the next position of sweep line.

진행선이 새로운 위치로 이동한 후 $C2$ 의 기준이 되었던 블록의 x_1 좌표값을 기준으로 왼쪽에 있는 블록들은 배선 결과를 기록한 후 삭제함으로써 전체 배선상황을 표현하기 위해 사용되는 타일의 수를 가능한 작게 한다.

2. 우선순위 결정

진행점들은 수직이동을 통하여 각자의 목표 터미널과 만나기에 가장 좋은 위치를 차지하려고 경쟁한다. 이러한 최적의 위치를 찾으려는 시도를 다른 진행점보다 먼저 할수록 선택의 폭이 넓어지고, 따라서 원하는 위치로 이동할 가능성이 높아진다. 우선순위는 이러한 진행점들간의 수직이동을 시도할 순서를 결정하는 것으로 그 값이 클수록 보다 빨리 수직이동을 시도할 수 있다. 제안하는 알고리즘은 수직이동 모듈을 수행하기 전에 모든 진행점들에 대해 현재 위치에서 오른쪽으로 가장 가까운 장애물을 찾고, 그 장애물까지의 거리에 따라 1부터 7까지의 우선순위를 다음의 기준에 따라 부여한다.

1. 진행선의 우측으로 배선영역의 끝까지 장애물이 없거나 가장 가까이에 있는 장애물이 목표 터미널인 경우는 가장 낮은 우선순위 1을 부여한다.
2. 진행선의 우측으로 가장 가까운 장애물이 진행점의 현재 위치와 장애물 사이에 비아를 6개 이상 만들 수 있는 위치에 있으면 장애물까지의 거리가 어느 정도 있어 수직 이동의 기회가 다른 진행점들에 비해 많으므로 우선순위 2를 부여한다. 만일 목표 터미널이 진행점이 되어 함께 진행하고 있다면 수직으로 연결하기만 하면 배선이 완료되므로 우선순위를 3으로 부여하여 빨리 배선을 완료할 수 있도록 한다.
3. 진행선의 우측으로 가장 가까운 장애물이 진행점의 현재 위치와 장애물 사이에 비아를 3개 이상, 6개 미만을 만들 수 있는 위치에 있으면 우선순위 4를 부여한다. 목표 터미널이 진행점이 되어 함께 진행하고 있다면 우선순위를 5로 부여한다.
4. 진행선의 우측으로 가장 가까운 장애물과의 사이에 비아를 2개 만들 수 있으면 우선순위 6을, 목표 터미널이 진행점이 되어 함께 진행하고 있다면 우선순위 7을 부여한다
5. 진행선의 우측으로 가장 가까운 장애물과의 사이에 비아를 1개 밖에 만들 수 없다면 우선순위를 7로 한다. 이 때는 수직이동의 기회가 단 한 번뿐이므로 진행점과의 연결을 목표로 하는 경우와 차등을 두지 않고 같은 우선순위를 부여한다.

3. 수직이동

진행점들에게 우선순위가 부여되면 몇 가지 규칙에 따라 수직이동을 시도한다. 우선 가능한 다른 진행점의 배선 진로를 빼앗지 않는 것을 기본으로 한다. 정해진

조건에 맞을 때만 수직이동하고, 일단 조건에 의한 수직이동이 이루어진 후에는 목표점과 바로 연결될 수 있는 위치가 아니면 수직이동하지 않는다. 수직와이어는 수직배선층에 배선하는 것이 원칙이지만 만일 수직와이어의 길이가 너무 짧아 수평배선층에 배선하여도 다른 진행점의 진로를 방해하지 않는다면 비아 대신 bend를 이용하여 수평배선층에 배선한다.

그림 9에 수직이동 과정을 보인다. 매개변수 P는 우선순위이고, 매개변수 D와 U는 대상이 되는 진행점들의 index의 하한과 상한을 의미한다. P의 초기값은 최고 우선순위인 7이고, D는 0, U는 (진행점들의 수 −1)을 초기값으로 갖는다. 만일, 어떤 진행점이 선택되어 이의 수직이동이 Y_d에서 $Y_u(Y_d < Y_u)$로 이루어졌고, 진행점의 index가 I_d에서 I_u로 바뀌었다면 진행점 (D, I_d-1)과 (I_u+1, U)에 대해 순환적으로 같은 과정을 반복함으로써 모든 진행점이 각자 가지고 있는 우선순위에 따라 수직이동을 시도할 수 있도록 한다.

```
수직이동 (D, U, P) {
  if (U≤ D)            return;
  do {
    U와 D 사이의 진행점 중 우선순위가 P이면서
        수직이동이 시도되지 않은 진행점 I_r을 찾는다;
    진행점 I_r이 시도되었음을 표시한다;
    I_r의 수직이동을 시도한다;
  } while( I_r이 수직이동에 실패 && U와 D사이에 수직이동이 시도되지 않은 진행점이 있다) {
  if( I_r이 y축의 Y_r에서 Y_d로 수직이동하였다 ) {
    I_d = 진행점의 새로운 index( Y_d);
    if( I_r < I_d)          {
        UP = I_d + 1;  DOWN = I_r - 1;}
    else                    {
        UP = I_r + 1;  DOWN = I_d - 1;}
    수직이동(UP, U, P);
    if (P > 2)
        수직이동(D, DOWN, P-1);
  }
  else
    if (P > 2)          수직이동(D, U, P-1);
}
```

그림 9. 수직이동 알고리즘
Fig. 9. An outline of the vertical movement algorithm.

수직이동을 시도하는 진행점은 먼저 자신의 목표 터미널로부터 진행점이 만들어졌는지를 확인하여 만일 자신과 함께 진행선을 따라 진행하고 있는 상태라면 서로 수직으로 연결함으로써 배선을 완료한다. 목표 터미널로부터 아직 진행점이 만들어지지 않았다면 진행점은 목표 터미널과 연결되기에 용이한 위치를 찾아 수직이동을 시도한다.

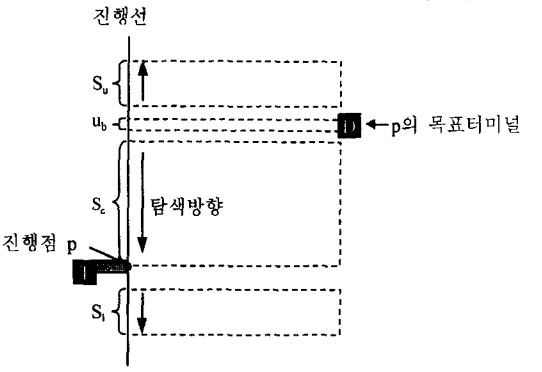

그림 10. 수직이동 탐색 구간
Fig. 10. Search ranges for vertical movement.

수직이동이 시도되는 구간은 그림 10과 같이 4개로 제한된다. 그림 10의 진행점 p의 네트번호가 n이고 네트 n의 와이어 폭이 ω_n, 다른 네트와 유지해야 할 최소 거리가 μ_n이며, 목표터미널 블록을 D라 할 때, 가장 먼저 수직이동이 시도되는 구간인 u_b의 정확한 범위는 $[D\cdot y_1+\omega_n/2,\ D\cdot y_2-\omega_n/2]$로, 이 구간의 의미는 구간 내의 한 점을 중심으로 수평 배선하면 목표 터미널까지 설계규칙을 어기지 않고 배선할 수 있다는 뜻이다. 다음으로 수직이동이 시도되는 구간 S_c의 범위는 $[D\cdot y_2+\omega_n/2,\ p\cdot y]$이고, 두 구간으로의 수직이동이 모두 실패하면 다음으로 S_u와 S_l으로의 수직이동이 시도되는데 이들의 범위는 각각 $[D\cdot y_1-\omega_n/2,\ D\cdot y_1-$전체수직구간의 15%]와 $[T\cdot y_2+\omega_n/2,\ T\cdot y_2+\omega_n/2+$전체수직구간의 15%]이다. 이들 구간의 범위는 목표터미널로부터 가장 가까운 위치에 만들어질 수 있는 수평와이어의 중심좌표와 진행점의 현재 위치에 만들어질 비아로부터 가장 가까이에 만들어질 수 있는 수평와이어의 중심좌표를 기준으로 결정된다. 그러나 만일 이 구간들 내에 장애물이 있다면 그 장애물을 피할 수 있도록 각 구간의 범위를 조정한다. 이렇게 재조

성이 된 구간 내의 점에 대해서는 진행점으로부터의 수직 배선이 가능하므로, 이후로는 구간 내의 점까지 수직 배선 가능성 여부를 확인하지 않아도 된다.

수직이동은 SEGRA와 마찬가지로 우선순위가 가장 높은 진행점부터 위에서부터 아래로 하나씩 선택하여 시도한다. 즉, 같은 우선순위를 갖는 진행점들 가운데 수직방향으로 위에 위치할수록 먼저 수직이동을 시도한다. 따라서 만일 한 진행점이 먼저 수직이동에 성공하면 그 진행점의 수직이동 이전의 위치와 수직이동 이후의 위치 사이에 있는 진행점들은 수직이동의 기회마저 잃게 된다. GMR에서는 이러한 점을 고려하여 우선순위가 6이상인 진행점들, 즉, 장애물이 매우 가까이에 있어 빨리 수직이동을 하지 않으면 현재 층에서의 배선을 포기해야 하는 진행점들에 대해서는 조금 다른 방법을 취한다. 우선순위가 6이상인 한 진행점의 아래 방향으로 수직이동 탐색 구간 내에, 역시 우선순위가 6 이상인 진행점이 있으면 이 진행점을 수직 배선의 장애물로 생각하여 구간을 재조정한다. 즉, 수직이동 가능 구간을 구간 내의 우선순위가 6이상인 진행점의 바로 위까지로 제한함으로써 아래 방향에 위치한 진행점의 수직이동 기회를 보장하자는 것이다. 이러한 방법은 한 번 수직이동에 성공한 진행점은 목표하는 터미널과 연결될 수 있는 위치가 아니면 수직이동할 수 없다는 원칙을 깨게 한다. 따라서 수직이동을 제한하는 원칙에 충실하다면 목표하는 터미널까지 진행점마다 2번 이하의 꺾어짐으로 목표하는 터미널과 연결될 수 있다는 보장도 지켜지지 않도록 한다. 그러나 한편으로는 현재의 배선층에서 보다 많은 네트가 배선할 수 있도록 한다.

실험결과에 의하면 우리의 배선 알고리즘은 SEGRA에 비해 많은 비아를 사용하지만 층별로 배선이 완료된 네트의 비율 면에서는 SEGRA보다 높다.

이렇게 재조정된 구간 내의 점들 가운데 수직이동 조건을 만족하는 한 점을 찾는다. SEGRA는 수직이동 조건을 만족하는 점을 찾기 위해 구간 내의 격자점들을 주어진 탐색방향으로 하나씩 선택하여 확인한다. 그러나 GMR에서는 구간이 연속적인 공간이므로 구간 내에서 임의의 한 점을 선택하여야 한다. GMR은 이러한 수직이동 조건을 확인할 점을 계산에 의한 방법으로 선택하는데, 만일 수직이동 조건을 확인하기 위한 다음 위치가 너무 멀리 떨어져 있다면 조밀한 배선결과를 기대하기 어렵고, 주어진 구간을 너무 촘촘한 간격으로 나누어 수직이동 조건을 확인하면 수행 시간이 길어진다. 탐색해야 할 구간과 탐색 방향이 주어졌을 때의 탐색 방법은 다음과 같다.

우선 탐색 방향에 맞추어 주어진 구간의 두 끝점 중 하나를 수직이동을 시도하는 초기값으로 선택한다. 이 점을 중심으로 수직이동 조건을 확인하고, 조건을 만족하지 못하게 하는 장애물이 발견되면 그 장애물을 간신히 피하여 수평 배선 할 수 있는 점이 다음 수직이동 조건 확인을 위한 점이 된다. 만일 발견된 장애물이 비아나 터미널, 또는 배선이 완료된 수평와이어라면 최소 거리만을 유지할 수 있도록 하고, 함께 진행중인 수평와이어라면 최소 유지 거리에 그 진행점이 수직이동할 때 만들어질 비아를 위한 영역을 고려한 위치가 다음 위치이다. 이러한 방법으로 탐색 구간의 끝까지 탐색해나가는데, 이 방법은 최대한 블록들이 조밀하게 놓일 수 있도록 하며, 가능성이 있는 점만이 조건 검색되도록 하므로 수행 시간이 절약된다.

그림 11은 탐색구간 S_c의 시작점인 $V1$으로의 수직이동에 실패한 경우 다음으로 수직이동을 시도할 위치를 계산하는 방법을 설명하는 그림이다. $V1$중심으로 목표터미널로부터 진행선 방향으로 수평 블록 찾기 모듈을 수행하면 타일 C가 가장 가까운 장애물 블록으로 보고된다. 이 블록은 진행중인 수평와이어이므로, 이 진행점이 수직이동을 할 경우 만들어질 비아를 위한 영역의 y_2좌표와 수평블록찾기 모듈의 수직 탐색 구간에 해당하는 점선의 윗변이 일치하도록 아래로 내려주면, 새로운 탐색 구간의 수직 중앙점이 다음 수직이동을 시도할 위치 $V2$가 된다.

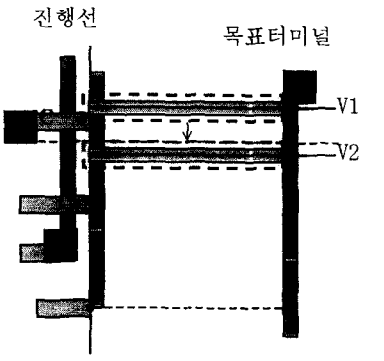

그림 11. 구간 S_c내에서 수직이동을 위한 다음 위치 계산에 대한 예

Fig. 11. An example of calculating the next position for vertical movement in S_c.

진행점의 수직이동 위치가 결정되면 그 진행점의 현재 위치와 새로운 위치에 비아가, 그리고 두 비아를 연결하기 위한 수직 와이어가 만들어지고, 진행선이 이동하면 새로운 위치의 비아와 연결하여 수평와이어가 만들어진다. 이때 만들어지는 수직와이어 블록의 x_1 좌표는 진행선의 x 좌표와 일치하여야 하며, 수평와이어 블록은 언제나 진행점의 y 좌표를 중심으로 만들어진다. 비아는 이미 배선이 끝난 영역의 빈 공간을 최대한 활용하기 위해 그림 12과 같이 수직와이어에 대해 왼쪽으로 치우치는 것을 원칙으로 한다. 비아와 수평와이어와의 관계는 배선 가능한 수평와이어의 수에 영향을 미친다. 진행선이 수평 방향으로 진행하기 때문에 수직와이어와의 관계와는 달리 여러 가지 상황을 고려해야 한다. 우선 수직이동 전의 위치에서 만들어지는 비아는 연결되는 수평와이어의 왼쪽에 있는 비아나 터미널 블록과 같은 y_1, y_2 좌표를 갖도록 한다. 그림 12의 비아 B는 이미 만들어진 터미널 블록 A를 기준으로 하여 위치를 잡는다. 이것은 이미 만들어진 비아나 터미널 블록의 모양을 기준으로 다른 네트들이 배선하기 때문이다. 즉, 주어진 위치에서 벗어나지 않도록 하는 것이 배선율을 가장 높일 수 있는 방법이 된다. 수직이동하려는 위치에 만들어지는 비아는 가능한 해당 네트가 차지하는 면적을 줄일 수 있는 것으로 선택한다. 그림 12의 예제에서는 진행점이 아래에서 위로 수직 이동하였으므로 비아 C가 아래쪽을 향하도록 하였다.

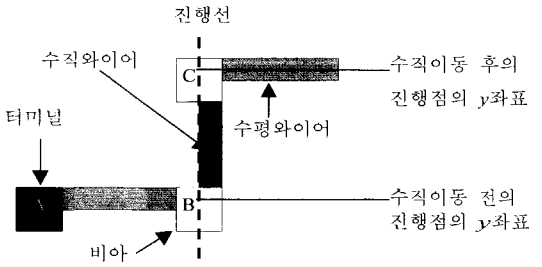

그림 12. 블록들이 만들어지는 위치
Fig. 12. The position for blocks.

주어진 위치에 비아를 만들기 위해서는 수직, 수평 배선층 모두에 대해 비아가 만들어질 영역이 비어 있고, 주위에 설계규칙을 어기게 할만한 블록이 없어야 한다. 이를 확인하고, 만일 주어진 영역 내에 블록이 발견되면 이를 피할 수 있는 새로운 비아 위치를 제안하는 과정이 그림 13과 같다. 먼저 비아 생성 규칙에서 제안하는 비아의 기본 위치에 대해 수평배선층을 탐색하여 장애물이 발견되면 그 장애물을 피할 수 있도록 비아의 위치를 조정하고, 장애물이 발견되지 않으면 그 위치를 고수한다. 수평배선층에서 결정된 위치로 수직배선층을 탐색하여 장애물이 발견되지 않으면 비아를 만들 수 있음을 알리고 종료한다. 만일 장애물이 발견되면 그 장애물을 피할 수 있도록 비아의 위치를 조정한 후, 다시 조정된 위치를 대상으로 수평배선층을 탐색한다. 여기서 장애물이 발견되면 비아 만들기를 포기하고, 장애물이 발견되지 않으면 수직배선층에서 결정된 위치가 만들어질 비아의 위치가 된다. 이렇게 기본 위치뿐 아니라 다른 위치에도 비아가 만들어질 수 있도록 하는 것은 비아의 모양 선택에 유연성을 부여함으로써 배선 영역을 아끼기 위함이다.

비아 위치 결정 순서

1. 수평배선층에서 비아생성규칙에 따른 위치를 기본으로 가능한 비아 위치 결정한다.
2. 1에서 비아 위치 결정에 실패하면 수행 종료한다.
3. 수직배선층에서 1에서 결정된 위치를 기본으로 가능한 비아 위치 결정한다.
4. 3에서 비아 위치 결정에 실패하면 수행 종료한다.
5. 3에서 결정된 비아의 위치와 1에서 결정된 비아의 위치를 비교하여
   a) 두 위치가 같으면 그 위치에 비아 생성이 가능함을 알리고 수행을 종료한다.
   b) 같지 않으면 3에서 결정된 위치를 기준으로 수평배선층에서 영역탐색하여,
      ① 장애물이 발견되면 비아 생성이 불가능함을 알리고 종료한다.
      ② 발견되지 않으면 3에서 결정된 위치에 비아 생성 가능함을 알리고 종료한다.

그림 13. 비아 위치 결정 알고리즘
Fig. 13. An outline of the via position determination algorithm.

그림 14는 이러한 비아의 위치를 결정하는 순서를 예를 들어 설명한다. 그림 14의 (a)를 보면 비아가 기본위치에 만들어질 때 수평배선층에서 블록 A가 장애물이 된다. 블록 A의 수직중심좌표가 대상영역의 수직중심좌표보다 아래쪽에 있으므로 블록 A의 y_1 좌표에 대상영역의 Y_2 좌표가 일치하도록 대상영역을 위로 올

려준다. 새로운 위치에 비아를 만들면 수평배선층에서는 설계규칙에 어긋나지 않으므로 이 위치를 비아의 위치로 지정하고 수직배선층에서 영역탐색 모듈을 수행한다. 그림 14의 (b)에서 볼 수 있듯이 수직배선층에서는 블록 B가 장애물이 되므로 블록 B의 x_2좌표에 대상영역의 X_1좌표가 일치하도록 대상영역을 오른쪽으로 이동한다. 이렇게 조정된 위치에 비아를 만들어도 수직배선층에서는 설계규칙에 어긋나지 않으므로 비아의 위치로 지정하고 다시 한번 수평배선층에서 새로운 위치에 비아를 만든다고 가정하고 영역탐색을 한다. 그림 14의 (c)를 보면 수평배선층에서도 대상 영역 내에

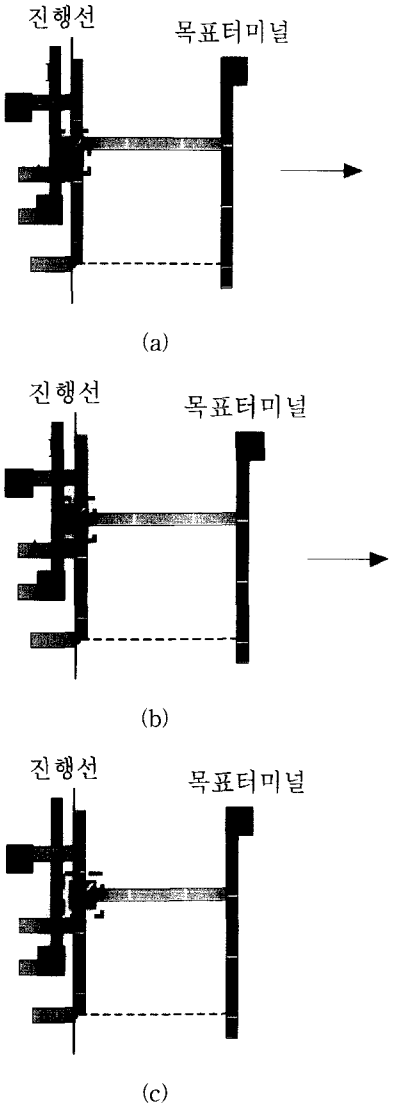

그림 14. 비아의 위치 찾는 예
Fig. 14. An example of the via position determination.

장애물이 발견되지 않으므로 수직배선층에서 결정된 위치가 비아가 만들어질 위치로 확정된다.

진행점의 수직이동 위치가 결정되고 그에 따른 블록들이 만들어지고 나면, 수직이동한 진행점과 함께 진행하던 진행점이 있는지, 있다면 함께 진행하고 있던 진행점들의 목표하는 점이 수직이동할 위치와 같은 방향인지를 확인한다. 만일 함께 진행하고 있던 진행점들의 목표 터미널이 수직이동한 진행점과 같은 방향이면, 그림 15와 같이 함께 수직이동하게 하거나 같은 네트인 수직와이어의 범위 내로 수직이동 하도록 한다. 이것은 진행선상에서 하나의 네트가 차지하는 범위를 작게 함으로써 다른 네트가 충분히 수직이동할 수 있게 한다. 특히 우선순위가 7인 진행점들은 반드시 따라가도록 하는데 이것은 현재 배선층에서 배선을 포기하는 진행점의 수를 적게 한다.

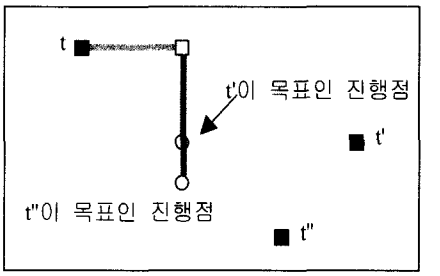

그림 15. 다중터미널인 진행점의 수직이동
Fig. 15. The vertical movement of active points which are mult-terminal net.

V. 실험 결과

GMR은 C언어를 사용하여 MOTIF 환경 하에서 동작하도록 구현하였다. 실험에 사용한 컴퓨터는 ULTRA SPARC II로 512Mb의 기억 용량을 갖는다. Khoo와 Cong[11-12]이 제공한 6개의 회로를 데이터로 사용하였는데 표2에 이 회로들의 특징을 보인다. 이중 test1, test2, test3은 두 개 터미널만을 갖는 네트를 임의로 구성한 것이고, mcc1-75, mcc2-75, mcc2-45는 실제로 다중칩 모듈 생산에 사용되는 것으로 MCC에 의해 제공되었다. 특히, mcc2는 37개의 VHSIC 게이트 어레이를 갖는 슈퍼 컴퓨터이다. mcc2-75, mcc-45는 같은 회로로, 각각의 배선 피치를 75㎛와 45㎛로 갖는다.

실험은 조건을 달리하여 모두 3번 이루어졌다. 먼저 제안하는 배선기를 SEGRA, V4R, SLICE, MCG와 비

교하기 위해 모든 네트에 대해 비아 크기와 와이어의 폭이 같도록 하였고, 크기 또한 최소 spacing 값과 더했을 때 배선 피치와 같아질 수 있도록 지정하였다. 이와 같은 조건의 실험을 실험 1이라 부르기로 한다. 다음으로 비아피치크기가 와이어피치크기의 2배가 되도록 모든 네트의 와이어의 폭을 비아 크기의 반으로 지정한 실험을 실험 2라 부르고, 마지막으로 Gridless 배선 방법의 장점을 살리고자 네트별로 각기 특성에 맞게 다른 와이어 폭을 선택하여 배선하고 이를 실험 3이라 부르기로 한다.

표 2. 테스트 데이터(mcc2-45의 피치 : 45㎛, 나머지 모두 : 75㎛

Table 2. Benchmark circuits(pitch of mcc2-45 : 45㎛, others : 75㎛)

Circuit	Number of Chips	Number Nets	Number of Pins	Size of Substrate(mm^2)	Percentage of Multiterminal Net
test1	4	500	1,000	22.5×22.5	0%
test2	9	966	1,912	30×30	0%
test3	9	1,254	2,508	37.5×37.5	0%
mcc1-75	6	802	2,49	45×45	24.19%
mcc2-75	37	7,118	14,659	152.4×152.4	5.90%
mcc3-45	37	7,118	14,659	152.4×152.4	5.90%

표 3은 제안하는 배선기와 다른 배선기의 모든 네트를 배선하는데 사용한 배선층의 개수를 비교한 것이다. 배선층의 개수는 제조 비용에 비례하므로 되도록 적은 층만으로 배선하는 것이 좋다. GMR은 실험 1의 조건에서는 SEGRA나 V4R과 같은 수의 배선층으로 배선한다. 실험 2의 배선환경에서는 상대적으로 더 가는 와이어로 인해 생기는 공간을 배선에 활용한 결과 mcc2-75 회로의 경우 4개의 배선층만으로 배선을 완료한다.

표 3. 배선층의 개수 비교

Table 3. The mumber of layers used for routing.

Circuit	GMR		MCG	SEGRA	V4R	SLICE
	실험1	실험2				
test1	4	4	자료 없음	4	4	5
test2	4	4		4	4	6
test3	4	4		4	4	6
mcc1-75	4	4	4	4	4	5
mcc2-75	6	4	4	6	6	7
mcc1-45	4	4	4	4	4	-

표 4는 우리의 배선환경인 ULTRA SPARC II를 기준으로 한 GMR의 실험 1과 실험 2에서의 수행시간이다. 실험 2에서 와이어 폭이 실험 1의 와이어 폭에 비해 반으로 줄어듬으로 인해 진행선이 보다 조밀한 간격으로 진행하게 되고, 따라서 수행시간도 2배 가량 더 소요된다.

표 4. GMR의 배선 수행 시간

Table 4. The running time for GMR to route.

Circuit	Running time(실험 1)	Running time(실험 2)
test1	00:00:03	00:00:05
test2	00:00:08	00:00:16
test3	00:00:14	00:00:27
mcc1-75	00:00:13	00:00:25
mcc2-75	00:03:57	00:08:20
mcc1-45	00:05:11	00:11:27

표 5는 다른 배선격자 기반 배선기들과 수행시간을 비교한 것이다. 참고문헌[4]에서는 V4R, SLICE와 비교하기 위해서는 32Mbyte의 메모리를 갖는 SPARC 2에서, MCG와 비교하기 위해서는 120Mbyte의 메모리를 갖는 SPARC 10에서 수행하였다. 표 5는 SEGRA를 ULTRA SPARC II에서 수행하여 SPARC 2와 SPARC 10에서 수행한 결과와의 속도비를 구한 후 GMR의 속도를 추측한 것이다. ULTRA SPARC II가 SPARC 2보다 약 7.04배, SPARC 10보다는 약 3.65배 빠르게 동작한다. GMR은 SEGRA를 제외한 어떤 배선기보다 빠르게 배선한다.

표 5. 실험 1의 수행시간의 비교(h:mm:ss)

Table 5. The running times of the five routers in Exp. 1(h:mm:ss)

Circuit	GMR	SEGRA	V4R	SLICE	GMR	MCG
test1	00:00:18	0:00:02	0:01:00	0:02:00	0:00:11	자료 없음
test2	0:00:57	0:00:05	0:01:00	0:06:00	0:00:31	
test3	0:01:35	0:00:08	0:03:00	0:12:00	0:00:51	
mcc1-75	0:01:25	0:00:10	0:03:00	0:12:00	0:00:47	0:09:00
mcc2-75	0:28:51	0:02:59	1:05:00	8:15:00	0:14:54	3:37:00
mcc1-45	0:35:42	0:04:09	1:37:00		0:18:34	1:48:00

표 6은 배선에 사용한 비아의 수를 비교한 것이다. 비아는 시스템의 성능을 저하시키므로 좋은 배선결과를 위해서 비아의 수가 적어야 한다. 비아의 수는 참고문헌[4]와 마찬가지로 2가지 방법으로 계산하였다. CM1은 모든 인접한 배선층 사이의 비아를 하나의 비아로 계산하는 것이고 CM2는 연속되는 비아들의 집합을 하나의 비아로 계산하는 것이다. GMR은 다른 배선기에 비해 조금 많은 비아를 사용한다. 이것은 4장에서 설명한대로 우선순위가 6 이상인 진행점들끼리는 서로 수직이동의 위치를 양보하기 때문이다. mcc1-75의 경우엔 SEGRA보다 적은 비아를 사용하는데 이것은 이 회로에서 다중 터미널 네트가 차지하는 비율이 높기 때문이며, 따라서 GMR이 SEGRA의 다중터미널 처리 방법의 단점을 완전하지는 않지만 해결하였음을 알 수 있다.

표 6. 실험 1의 비아의 수 비교
Table 6. The number of via used for routing in Exp. 1.

circuit	GMR(실험1)		SEGRA		MCG	V4R	SLICE
	CM1	CM2	CM1	CM2			
test1	2,945	2,281	2,619	2,080	자료 없음	2,250	2,013
test2	6,990	4,676	6,610	4,476		4,493	5,271
test3	9,105	6,055	8,692	5,833		5,855	6,892
mcc1-75	8,264	6,202	9,310	6,773	5,742	6,993	6,386
mcc1-75	52,777	36,839	52,261	36,362	34,300	36,438	47,864
mcc1-45	42,148	36,233	39,911	34,305	33,238	36,473	

표 7은 층별로 배선이 완료된 네트의 비율을 비교한 것이다. SEGRA에 비해 같은 조건 하의 GMR의 층별 배선율이 높은데, 이것은 우선순위가 6이상인 진행점들끼리 수직이동 가능 구간을 서로 조금씩 양보함으로써 서로의 수직이동 기회를 보장하였기 때문이다. 즉, 비아를 조금 많이 쓰는 대신 층별 배선율을 높인 것이다. 또한 실험1에서의 배선율과 실험2에서의 배선율을 비교하면 실험2에서의 배선율이 더 높고, 특히 mcc2-75 회로의 경우 높아진 배선율로 인해 배선층이 2개 줄었음을 알 수 있다. 앞서 사용된 배선층의 비교에서 언급한 바와 마찬가지로 실험 1에 비해 더 가는 와이어를 사용함으로써 생기는 공간을 다른 네트의 배선에 활용하였기 때문이다. 만일 배선격자 기반 배선방법으로 배선하였다면 실험1에서와 같은 배선율로 배선하고 따라서 mcc2-75의 경우에도 배선층의 수를 줄일 수 없을 것이다.

표 7. GMR과 SEGRA의 층별 배선율 비교
Table 7. The comparison of completed routes per two layers.

Circuit	SEGRA			GMR(실험1)			GMR(실험2)	
	1-2	3-4	5-6	1-2	3-4	5-6	1-2	3-4
test1	79.60%	100.00%		87.80%	100.00%		92.20%	100.00%
test2	58.10%	100.00%		58.20%	100.00%		64.37%	100.00%
test3	57.66%	100.00%		58.70%	100.00%		63.00%	100.00%
mcc1-75	75.03%	100.00%		81.48%	100.00%		87.33%	100.00%
mcc1-75	57.89%	91.91%	100.00%	67.00%	99.28%	100.00%	73.46%	100.00%
mcc1-45	85.07%	100.00%		94.89%	100.00%		97.69%	100.00%

Grid를 사용하지 않고 배선하는 배선 방법으로서의 특징 가운데 하나가 비아 크기와 와이어 폭의 크기가 다를 때 배선 면적이 줄어든다는 점이다. 본 실험에 사용되었던 벤치마크들은 모두 배선 격자 기반 배선기들을 위한 회로들이므로 칩들의 위치가 고정되어 있어 배선 면적이 줄어드는 것을 확인하는 것이 불가능하다. 그러나 상대적으로 작은 폭의 와이어로 인해 생기는 공간을 다른 네트의 배선에 활용하고, 따라서 사용하는 배선 층의 수를 줄일 수 있다면 배선 면적이 줄어들었다고 말할 수 있다. 표 7을 보아 실험 1의 배선율에 비해 실험 2의 배선율이 높고, 더욱이 mcc2-75의 경우 사용된 배선 층의 수가 적으므로 배선 면적이 줄었다고 할 수 있다

실험 3에서는 네트별로 와이어의 폭을 다르게 지정하였다. 그림 16은 test 1회로의 실험 3의 환경에서

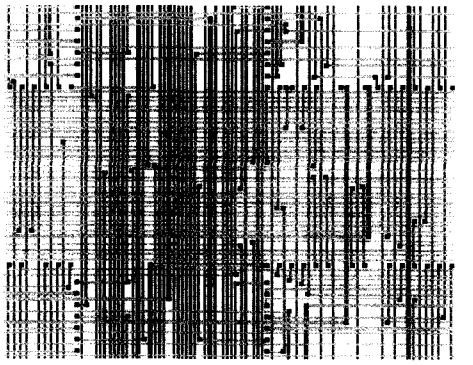

그림 16. 실험 3에서 GMR에 의한 test1 회로의 배선 결과의 일부
Fig. 16. Partial routing result shown here for the "test1" in Exp. 3.

(GMR에 의한 배선결과의 일부로, 내부가 검은 색의 사각형은 터미널, 외곽선이 있는 사각형은 비아, 짙은 회색은 수직 와이어, 옅은 회색은 수평 와이어이다. 네트별로 와이어의 폭이 다름을 볼 수 있다.

VI. 결 론

본 논문에서는 다중칩 모듈 배선을 위한 gridless 배선 알고리즘을 제안하였다. 우리의 배선기는 배선격자를 사용하는 대신 corner stitching 자료구조와 연결 리스트 자료구조를 함께 사용하여 배선영역을 표현하는 방법을 택함으로써 배선격자를 기반으로 배선할 때의 단점을 피한다. 즉, 비아보다 그 폭이 작은 와이어들로 인해 생기는 공간을 다른 네트의 배선에 이용함으로써 배선영역을 최대로 이용하고, 와이어 폭을 네트별로 자유롭게 지정할 수 있으며, 핀들이 반드시 격자점 위에 놓여야 하는 제약을 갖지 않는다. GMR은 배선을 하는 동안 핀의 4~5배 정도에 해당하는 수의 타일만으로 두 배선층의 배선 상태를 표현하고, SEGRA를 제외한 어느 배선격자 기반 배선 방법들보다 짧은 시간 내에 배선을 마친다. 비록 조금 많은 비아를 사용하긴 하지만, 다중 터미널 네트를 비교적 효율적으로 처리하므로, 다중 터미널 네트가 차지하는 비율이 높은 회로에 대해서는 다른 배선 방법에 비해 적은 비아를 사용할 수도 있으리라 예상된다.

참 고 문 헌

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Menlo Park, CA, 1987.

[2] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Menlo Park, CA, 1990.

[3] A. J. Blodgett, "Microelectronic packaging," *Scientific American*, pp. 76-86, July 1983.

[4] Y.-J. Cha, C. S. Rim and K. Nakajima, "A simple and effective greedy multilayer router for MCMs," in *Proc. Int. Symp. Physical Design*, 14-16 April 1997, pp. 67-72.

[5] W. W. Dai, T. Dayan., and D. Staepelaere. Topological Routing in SURF: Generating a Rubber-Band Sketch. *ACM/IEEE 28th Design Automation Conference*, pp. 39-44, June 1991.

[6] D. A. Doane, P. D. Franzon, "*Multichip Module Technologies and Alternatives, The Basics*", Van Nostrand Reinhold, NY, 1993.

[7] T. Hameenanttila, J. D. Carothers and D. Li, "Fast Coupled Noise Estimation for Crosstalk Avoidance in the MCG Multichip Module Autorouter," *IEEE Trans. on Computer-Aided Design*, Vol. 4, No. 3, pp. 356-368, Sept. 1996.

[8] A. Hanafusa, Y. Yamashita, and M. Yasuda, "Three-Dimensional Routing for Multilayer Ceramic Printed Circuit Boards," *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pp. 386-389, Nov. 1990.

[9] D. Herrell, "Multichip module technology at MCC," *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 2099-2103, May 1990.

[10] J. M. Ho, M. Sarrafzadeh, G. Vijayan, and C. K. Wong, "Layer Assignment for Multichip Modules," *IEEE Trans. on Computer-Aided Design*, Vol. 9, pp. 1272-1277, Dec. 1990.

[11] K. Y. Khoo, and J. Cong, "A Fast Multilayer General Area Router for MCM Designs," *IEEE Trans. on Circuits and Systems II*, pp. 841-851, Nov. 1992.

[12] K. Y. Khoo, and J. Cong, "An Efficient Multilayer MCM Router Based on Four-Via Routing," *IEEE Trans. on Computer Aided Design*, Vol. 14, pp. 1277-1290, Oct. 1995.

[13] R. Miracky, et al., "Technology for Rapid Prototyping of Multi-Chip Modules," *IEEE Int'l Conf. on Computer Design*, pp. 588-591, 1991.

[14] J. Ousterhout, "Corner stitching: A data-structuring technique for VLSI layout tools," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 87-100, 1984.

[15] H. B. Richard, P. M. William, and H. Jackson, "MultiChip Modules," *ACM/IEEE 26th Design Automation Conference*, pp. 389-393, 1989.

[16] N. Sherwani, *Algorithms for VLSI Physical*

Design Automation, Kluwer Academic Publishers and Toppan Company (S) Pte Ltd., 1996.

[17] C. C. Tsai, S. J. Chen, and W. S. Feng, "An H-V tile-expansion router," in *Proc. Nat. Computer Symp.*, 1989, pp. 106-115.

[18] C. C. Tsai, S. J. Chen, P. Y. Hsiao, and W. S. Feng, "New iterative construction approach to routing with compacted area," *Proc. Inst. Elec. Eng.*, pt. E, vol. 138, no. 1, pp. 57-71, 1991.

[19] C. C. Tsai, S. J. Chen, and W. S. Feng, "An Alternating Router," in *IEEE Trans. on Computer-Aided Design*, Vol. 11, No. 8, pp. 976-991, Aug. 1992.

저 자 소 개

李 台 先(正會員)
1992년 서강대학교 전자계산학과 학사. 현재 서강대학교 컴퓨터학과 석사과정

林 種 錫(正會員) 第 36卷 C編 第 1號 參照