BSG 구조에서 압축을 이용한 플로어플랜 기법

성영태○ 허성우
동아대학교 컴퓨터공학과
saint○@donga.ac.kr, swhur@dau.ac.kr

Floorplan Technique Using Compaction on BSG-Structure

YoungTae Sung○ SungWoo Hur
Computer Engineering of University Dong-A

요 약

BSG(Bounded Sliceline Grid)를 이용한 플로어플랜 기법은 매우 빠르고 효과적이나 모듈 사이에 빈 공간이 존재하여 필요 이상으로 면적을 넓게 차지하는데도 불구하고 그 점을 무시한 채 배치 면적을 구하는 문제점이 있다. 본 논문에서는 BSG 구조를 이용한 플로어플랜 과정 중 빈 공간이 생기는 문제점을 해결하기 위해 모듈들을 좌측 또는 아래로 옮길 수 있는데 까지 옮기는 압축 기법을 추가하여 필요한 면적이 최소가 되도록 하였다. 실험 결과는 압축 기법을 사용하는 것이 사용하지 않을 때보다 최소 면적과 평균 면적 면에서 모두 개선되는 것을 보여 준다.

1. 서 론

VLSI 설계 과정에서 중요한 단계 중의 하나가 플로어플랜 설계 단계이다. 플로어플랜의 목표는 평면상에서 직사각형 모듈들을 중첩되지 않게 배치하면서 원하는 목적함수를 최적화 시키는 것이다. 목적함수로는 플로어플랜의 총 면적, 배선길이 또는 면적과 배선길이에 가중치를 부여한 합 등이다. 플로어플랜 기법은 크게 두 가지로 분류될 수 있다. 즉, 분할구조(slicing structure)와 비분할구조(non-slicing structure)에 근거한 기법으로 나눌 수 있다.

분할에 의한 플로어플랜은 VLSI 영역을 수직 혹은 수평으로 재귀적으로 반복하여 분할함으로써 얻을 수 있다. 분할 플로어플랜은 해 공간(solution space)이 제한되는 단점이 있으나 이진트리 또는 Polish 수식을 이용하여 효과적으로 표현할 수 있는 장점이 있다. 분할 트리를 이용할 경우 해 공간의 크기는 $O(n!\,2^{3n-3}/n^{1.5})$이며, 어떤 트리로부터 플로어플랜을 $O(n)$ 시간에 구할 수 있다[1].

비분할 구조를 이용할 경우 문제는 훨씬 복잡해지는데, 이를 위한 좋은 모델로는 SP를 이용한 것[2,3], BSG(Bounded-Sliceline Grid) 구조를 이용한 것[4,8,9], O-tree를 이용한 것[5,6], CBL(Corner Block List)을 이용한 기법[7] 등이 있다.

SP 모델에서는 n개의 모듈의 위상적 관계를 나타내기 위해 두 순열을 이용하기 때문에 해 공간의 크기는 $O((n!)^2)$이 된다. 한 쌍의 SP로부터 대응하는 배치를 얻는 시간은 $O(n^2)$이 걸린다[4]. BSG 구조에서는 $n\times n$ 격자(grid)를 이용하여 모듈을 배치하며, 배치를 얻는데 걸리는 시간은 $O(n^2)$이고, 해 공간의 크기는 $O(n!C(n^2,n))$이다. BSG를 이용한 배치는 T-모양과 L-모양의 블록을 효과적으로 배치하도록 확장되었으며[8], 최적의 라우팅 공간을 찾도록 개선되었다[9]. O-tree 모델에서는 해 공간이 $O(n!\,2^{2n-2}/n^{1.5})$으로서 상대적으로 적으며, 한 트리로부터 배치를 얻는 시간은 $O(n)$이다. CBL 모델을 사용하는 기법의 해 공간의 크기는 $O(n!\,2^{3n-3}/n^{1.5})$이며 주어진 CBL로부터 배치를 구하는데 필요한 시간은 $O(n)$이다. O-tree나 CBL 모델에서는 해 공간의 크기가 제한되어 있기 때문에 최적의 배치를 찾지 못할 수도 있다.

BSG 구조를 사용할 경우 해가 중복되는 단점이 있으나 SA 휴리스틱을 적용하는 과정에서 모든 변형된 해가 배치가능한 해라는 장점이 있다. BSG 구조의 또 다른 문제는 격자의 경계 간에 미리 설정된 상관관계 때문에 BSG 구조의 격자 내에 할당된 모듈들 사이에 불필요한 빈 공간이 있게 되는 문제점이 있다. 본 논문에서는 BSG 구조에서 이런 불필요한 빈 공간을 포함하지 않도록 SA 휴리스틱을 적용하는 과정에서 효과적인 압축 기법을 적용하여 배치 영역을 가능한 적게 하도록 한다. 이렇게 함으로써 더 좋은 해를 찾을 수 있게 하였다.

본 논문의 구성은 다음과 같다. 2장에서 일반적인 BSG 구조를 설명하고 3장에서 제안하는 압축 기법을 설

명하며, 4장에서는 실험 결과를 보인다.

2. BSG(Bounded-Sliceline Grid) 구조를 이용한 플로어플랜

2.1 BSG 구조와 방향성 그래프

BSG 구조는 평면 상에서 위상적으로 정의된 격자이다. (x,y)-좌표 시스템에서 열린 수직 선분 $H_{i,j}$와 수평 선분 $V_{i,j}$ $(i,j$: 정수)는 다음과 같이 정의된다.

$H_{i,j} = (x,y) | i-1 < x < i+1,\ y = j$

$V_{i,j} = (x,y) | x = i,\ j-1 < y < j+1$

여기서 i,j는 단위 선분의 중심을 의미한다. BSG는 아래와 같은 선분들의 집합 U_{BSG}로 구성된 격자 시스템이다.

$$U_{BSG} = \{ V_{i,j} | i,j : 정수, i+j : 짝수 \} \cup \{ H_{i,j} | i,j : 정수, i+j : 홀수 \}$$

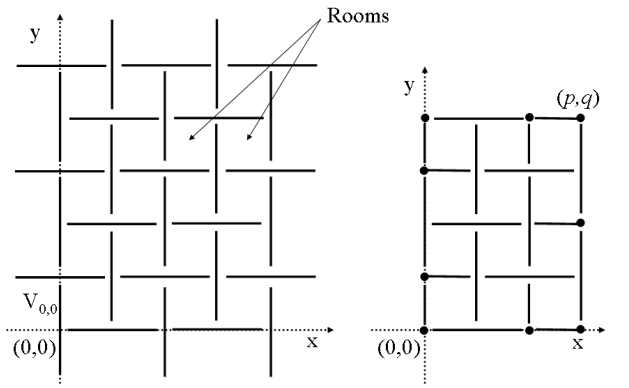


그림 1 BSG와 Domain $BSG_{p\times q}$

그림 1이 BSG의 예를 보여 준다. $|i'-i|=1$이고 $|j'-j|=1$일 때 두 수직 선분 $V_{i,j}$와 $V_{i',j'}$는 인접하다고 말한다. 만약 $i'=i-1$이고 $|j'-j|=1$이면 $V_{i,j}$는 $V_{i',j'}$ "우측에 있다"고 말한다. "우측에 있다"는 성질은 이행성(transitivity)이 만족된다. 즉, 어떤 수직 성분 V_a가 다른 수직 선분 V_b에 있고, V_b가 V_c 우측에 있다면 V_a는 V_c 우측에 있다. 수평성분에 대해서 "인접하다"는 성질과 "위에 있다"는 성질은 유사하게 정의될 수 있다. 수직선분고 수평선분에 의해 둘러싸인 공간을 방(room)이라 부른다.

BSG는 무한 격자이지만 플로어플랜을 위해 p 개의 행과 q 개의 열로 제한된 격자를 사용하고, 그런 유한 격자를 $BSG_{p\times q}$로 나타내며, 하단 좌측 모서리가 원점(0,0)이 되며 상단 우측 모서리의 좌표는 (p, q)가 된다. 이런 관계를 그림 1(B)가 보여준다.

$BSG_{p\times q}$가 주어질 때, 이 격자 상에서 두 개의 방향성 그래프가 다음과 같이 정의된다. 한 개는 수평 인접 그래프라고 불리는 $G_h = (V_h, E_h)$인데 여기서 $V_h = \{s_h, t_h\} \bigcup \{u_{i,j}\}$이다. $u_{i,j}$는 $H_{i,j}(i+j : odd)$에 대응되는 정점이다. E_h는 다음과 같이 정의된다. 모든 하단 수평 선분 즉, $H_{1,0}, H_{3,0}, \ldots, H_{2i+1,0}$ (여기서 $i=(p-1)/2$)에 대해 방향성 에지 $(s_v, H_{2i+1,0})$가 존재한다. 또한 모든 상단 수평 성분 즉, $H_{1,q}, H_{3,q}, \ldots, H_{2i+1,q}$ (여기서 $i=(p-1)/2$)에 대해 방향성 에지 $(H_{2i+1,q}, t_h)$가 존재한다. 그리고 임의의 수평 선분 $H_{i,j}$에 인접하고 위에 있는 선분 $H_{i',j'}$사이에 방향성 에지 $(u_{i,j}, u_{i',j'})$가 존재한다.

수직 인접 그래프라 불리는 $G_v = (V_v, E_v)$도 유사하게 정의된다. 그림 2에서 수평/수직 인접 그래프의 예를 보였다.

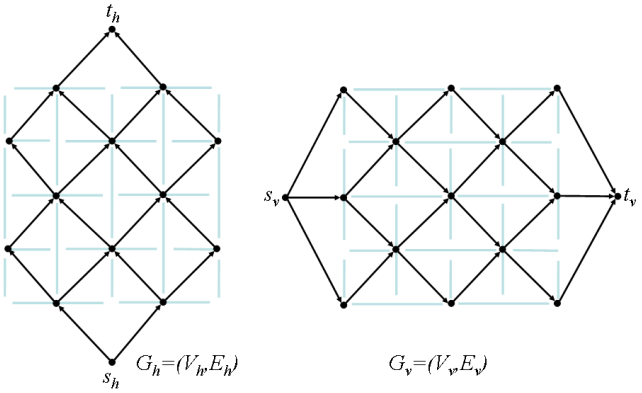


그림 2 수평 및 수직 인접 그래프의 예

2.2 BSG 구조로부터 모듈의 좌표 구하기

n개의 모듈로 구성된 집합 M이 주어졌을 때 (즉, $|M|=n$) $p\times q \geq n$이라고 가정하면 $BSG_{p\times q}$에 M을 할당하는 것은 n개의 모듈을 $BSG_{p\times q}$상의 방에 매핑하는 것이다. 어떤 모듈도 할당되지 않은 방을 빈방이라 부른다. n개의 모듈을 $BSG_{p\times q}$상의 방에 할당한 후 대응되는 수직/수평 인접 그래프의 에지에 가중치를 부여하게 되는데 그 원칙은 다음과 같다. 만약 $e \in E_h$이고, e에 대응하는 방에 어떤 모듈이 놓여 있다면 $w(e)$는 그 모듈의 높이와 같도록 부여한다. 유사하게 만약 $e \in E_v$이고, e에 대응하는 방에 어떤 모듈이 놓여 있다면 $w(e)$는 그 모듈의 너비와 같게 한다. 빈 방에 대응되는 각 에지의

가중치는 0이다.

예를 들어 5개의 모듈의 크기(너비,높이)가 다음과 같다고 하자. ⓐ:(8,3), ⓑ:(6,9), ⓒ:(11,4), ⓓ:(9,8), ⓔ:(10,8). 이 모듈이 그림 3(A)에서 보인 것과 같이 방에 할당되었다고 하면 이 예로부터 구한 가중치가 부여된 수직/수평 인접 그래프는 그림 3(B)와 같다. 모든 정점 $u \in V_h$에 대해 $l_h(u)$는 정점 s_h로부터 정점 u까지의 경로 중 가장 긴 경로의 길이를 나타낸다. 유사하게 모든 정점 $u \in V_v$에 대해 $l_v(u)$는 정점 s_v로부터 정점 u까지의 경로 중 가장 긴 경로의 길이를 나타낸다. 수직/수평 인접 그래프에 있는 모든 정점 u에 대해 $l_h(u)$및 $l_v(u)$는 잘 알려진 최장경로길이를 구하는 알고리즘을 이용하면 쉽게 구할 수 있다. 수직/수평 인접 그래프의 각 정점에 대해 최장경로를 구하는 이유는 각 정점에 대응되는 모듈의 위치를 결정하기 위함이다. 즉, 수직/수평 인접 그래프를 이용하여 각 모듈의 x/y 좌표를 결정할 수 있다. 그림 3에서 보인 예로부터 구한 각 모듈의 x, y 좌표는 다음과 같다. ⓐ:(0,0), ⓑ:(0,12) ⓒ:(8,0), ⓓ:(18,4), ⓔ:(8,4).

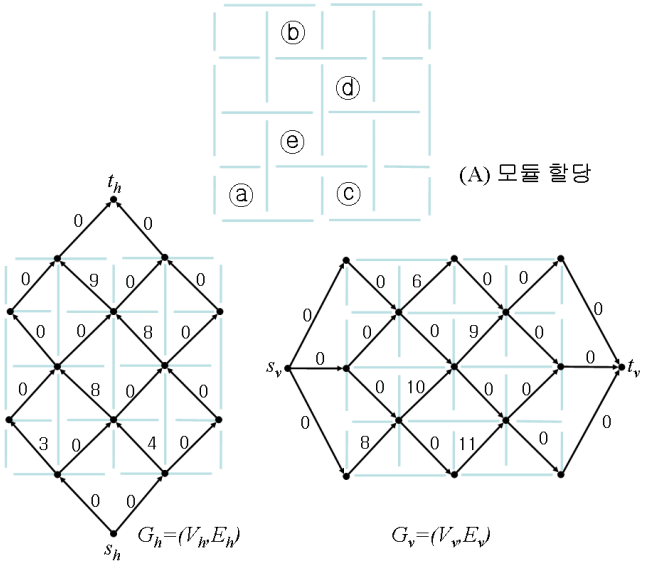


그림 3 격자에 모듈 할당과 이로부터 구한 가중치가 부여된 인접그래프의 예

3. BSG 구조의 문제점을 개선한 기법

3.1 BSG 구조를 이용한 플로어플랜의 문제점

그림 3에서 보인 것과 같은 플로어플랜에서 모든 모듈을 포함하는 최소 직사각형의 너비와 높이는 각각 27과 21이 되어 직사각형의 총 넓이는 567이 된다. 이 관계를 그림 4가 보여준다.

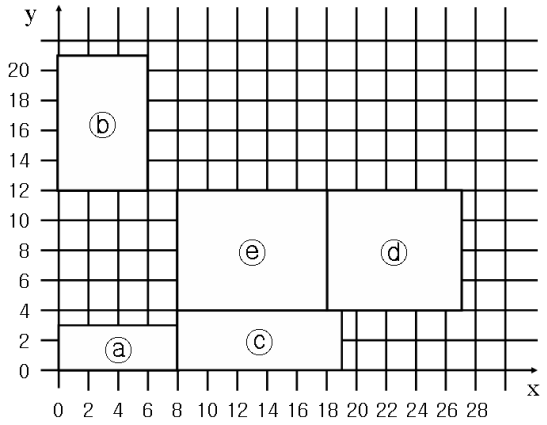


그림 4 그림 3(A)에 대응하는 배치

BSG 구조의 격자 상에 모듈을 할당할 경우 격자의 각 방 관계의 상관관계(즉, 위측 또는 우측)가 미리 정의 되어 있어 그림 4에서 볼 수 있듯이 모듈 간에 빈 공간이 많이 있음에도 불구하고 이를 고려하지 않은 채 주어진 할당으로부터 최장경로 알고리즘을 사용하기 때문에 최종적으로 구한 필요 면적이 필요 이상으로 커지는 문제점이 있다.

3.2 압축을 이용한 개선된 기법

그림 4에서 보였듯이 주어진 모듈을 BSG 구조의 격자에 할당했을 때의 문제는 모듈 사이에 불필요한 공간이 있다는 점이다. 격자 상에 할당된 모듈들을 '아래로' 또는 '좌측으로' 이동을 반복하여 더 이상 모듈들을 옮길 수 없을 때 까지 옮겨 빈 공간을 없앤 후 모듈들을 둘러싸는 최소 면적을 구하고자 한다. 이처럼 빈 공간을 없애기 위해 모듈을 옮기는 과정을 "압축한다" 라고 표현한다. 모듈들을 아래로 이동하여 압축하기 위한 알고리즘은 다음과 같다.

algorithm CompactDown
1. 모듈들의 y-좌표에 따라 오름차순으로 정렬한다. 동일한 y-좌표를 가지는 모듈들은 x-좌표에 따라 오름차순으로 정렬한다.
2. 정렬된 순서대로 모듈을 아래로 이동할 수 있는 만큼 이동한다.

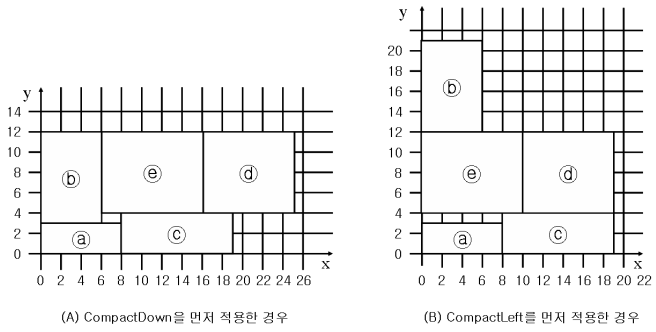
[5]에서 설명된 윤곽(contour) 자료 구조를 이용하면 알고리즘 CompactDown을 효과적으로 구현할 수 있다. 단계 1에서 정렬하는데 $O(nlogn)$ 시간이 필요하며, 단

계 2는 이미 모듈들이 정렬되어 있기 때문에 $O(nlogn)$ 시간에 실행가능하다. 모듈을 좌측으로 이동하여 압축하는 알고리즘 CompactLeft도 유사하게 기술할 수 있다.

BSG 격자 구조에 할당된 모듈들 사이의 빈 공간을 없애기 위해 CompactDown과 CompactLeft를 번갈아 가면서 적용하면 된다. 그런데 CompactDown을 먼저 적용할 때와 CompactLeft를 먼저 적용할 때의 결과가 다르기 때문에 다음에 제시한 알고리즘 Compact에선 두 가지를 모두 적용한 후 좋은 결과를 반환한다.

```
algorithm Compact
1. Do {
      Apply CompactDown;
      Apply CompactLeft;
   } while(any module moves)
2. ResDownFirst = 모듈을 둘러싸는 직사각형의 면적
3. Do {
      Apply CompactLeft;
      Apply CompactDown;
   } while(any module moves)
4. ResLeftFirst = 모듈을 둘러싸는 직사각형의 면적
5. Return min(ResDownFirst, ResLeftFirst)
```

그림 5(A) 에서는 그림 4에서 보인 배치에서 CompactDown을 먼저 적용하여 구한 배치, 즉 알고리즘 Compact의 단계 2에서 구한 ResDownFirst의 결과를 보여준다. 이 경우 ResDownFirst의 값은 $25\times12=300$이 된다. 그림 5(B)는 ResLeftFirst의 결과를 보여주는데 이 값은 $19\times21=398$이 되어 알고리즘 Compact는 300을 반환하게 된다.



(A) CompactDown을 먼저 적용한 경우 (B) CompactLeft를 먼저 적용한 경우

그림 5 그림 4에서 보인 배치에 알고리즘 Compact를 적용한 예

3.3 Simulated Annealing 기법을 이용한 플로어 플랜

BSG 구조의 격자 상에서 임의로 모듈을 배치한 후 앞 절에서 설명한 알고리즘 Compact를 이용하여 그에 대응하는 면적을 구한다. 그 면적을 현재의 해로 둔다. 그리고 격자 상에 있는 임의의 두 방을 선택한다. 두 방 모두 빈방이면 다시 방을 선택한다. 빈 방이 아닌 방이 선택될 경우 그 방에 있는 모듈을 서로 맞바꾼다.

만약 한 방은 빈 방이고 다른 한 방엔 모듈이 있는 경우라면 이는 그 모듈을 격자 상에서 다른 위치로 옮기는 것과 같다. 모듈의 위치를 바꿀 때 모듈의 방향을 90도 돌리는 것도 같이 고려하여 가능한 모든 변화 각각에 대해 Compact 알고리즘을 이용하여 면적을 구한다. 가능한 모든 변화를 다 고찰한 후 면적이 가장 작게 되는 변화에 대응하는 배치의 결과를 새로운 해로 둔다.

새로운 해가 현재의 해보다 좋으면 새로운 해를 현재의 해로 둔다. 새로운 해가 현재의 해보다 좋지 않아도 simulated annealing 기법에선 나빠진 정도와 현재의 온도에 근거해 나빠진 해를 새로운 해로 채택하기도 한다. 나빠진 해를 현재의 해로 채택할 확률은 온도에 의해 조절되며, 온도는 점진적으로 낮아지고 최종적으로 해를 찾는 반복은 중지하게 되고, 그 때의 해가 최종 결과로 반환된다.

4. 실험 및 고찰

본 논문에서 제안한 알고리즘은 Pentium IV 2G/Linux 상에서 C로 구현되었다. 실험은 MCNC 회로에 대상으로 하였고, BSG 구조에서 알고리즘 Compact를 적용한 경우와 적용하지 않은 경우를 비교하였다. 각 회로에 대해 100번 씩 실험을 하였고, 초기 배치는 무작위로 구하였다. 표 1에서 두 가지 경우에 대해 가장 좋은 결과를, 표 2에선 평균 결과를 보였고. 그리고 표 3에선 실행 시간을 보였다.

결과에서 보듯이 알고리즘 Compact를 적용한 경우에 최소면적은 동일한 결과를 보이거나 개선되는 것을 보여준다. 또한 평균 면적의 경우는 Compact를 적용한 경우가 항상 개선되는 것을 보여 준다. 이는 Compact를 사용하는 것이 더 좋은 결과를 안정적으로 얻을 수 있다는 뜻을 내포하기 때문에 Compact를 사용하는 것이 바람직하다고 볼 수 있다. 실행 시간 면에서 예측한대로 Compact를 적용한 경우가 상대적으로 더 많은 시간을 필요로 하지만 이 경우 절대적인 실행 시간은 합리적이라고 판단된다.

표 1 실험결과 (최소 면적)

회로	면적 (최소)	
	without Compact	with Compact
apte	47078460	47078460
xerox	19831084	19831084
hp	9201024	9112824
ami33	1205400	1192464
ami49	37048704	36900136

표 2 실험결과 (평균 면적)

회로	면적 (평균)	
	without Compact	with Compact
apte	50554312	47561624
xerox	20451976	20351250
hp	9422629	9349325
ami33	1235762	1232461
ami49	37909617	37754623

표 3 실험결과 (실행시간)

회로	CPU 시간 (sec)	
	without Compact	with Compact
apte	0.1	0.5
xerox	0.1	0.6
hp	0.1	0.9
ami33	0.7	7.8
ami49	1.2	14.4

5. 결 론

본 논문에서는 [4]에서 제안한 BSG 구조를 이용한 플로어플랜의 단점 즉, 모듈 사이에 없앨 수 있는 빈 공간이 있음에도 불구하고 그것을 그대로 둔 채 면적을 계산하는 문제점을 개선한 기법을 제시하였다. 모듈을 아래로 또는 좌측으로 옮길 수 있을 만큼 옮기는 효과적인 Compact 기법을 이용하여 배치의 총 면적을 가능한 줄인 상태에서 플로어플랜의 면적을 계산하였다.

실험결과에 의하면 최소면적과 평균면적 모두 제시한 기법을 사용하는 경우 개선되는 것을 볼 수 있다. Compact 기법을 사용함으로써 CPU 시간은 더 필요하지만 절대적인 시간 면에서 Compact 기법을 사용해도 충분히 용납할 만하기 때문에 안정적으로 좋은 결과를 얻기 위해선 제시한 Compact 기법을 사용하는 것이 바람직해 보인다.

또한 측정은 하지 않았지만 Compact 기법을 사용하는 경우 배선 길이 면에서도 더 좋은 결과를 보일 것으로 기대된다.

참고문헌

[1] D.F.Wong and C.L.Liu, "A New Algorithm for Floorplan Design," Proc. of DAC, pp. 101-107, 1986.

[2] H.Murata, K.Fujiyoshi, S.Nakatake, and Y.Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 15, No. 12, pp. 1518-1524, 1996.

[3] X.Tang and D.F.Wong, "FAST-SP: A Fast Algorithm for Block Placement Based on Sequence Pair," Proc. ASP-DAC, pp. 521-526, 2001.

[4] S.Nakatake, K.Fujiyoshi, H.Murata and Y.Kajitani, "Module Placement on BSG-Structure and IC Layout Applications," Proc. of ICCAD, 1996.

[5] P.Guo, C.Cheng and T.Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications," Proc. of DAC, pp.268-273, 1999.

[6] Y.Pang, C.Cheng and T.Yoshimura, "An Enhanced Perturbing Algorithm for Floorplan Design Using the O-tree Representation," Proc. of ISPD, pp. 168-173, 2000.

[7] X.Hong, G.Huang, Y.Cai, J.Gu, S.Dong, C.Cheng and Jun Gu, "Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan," Proc. of ICCAD, pp.8-12, 2000.

[8] M.Kang and W.Dai, "General Floorplanning with L-shaped, T-shaped and Soft Blocks Based on Bounded Slicing Grid Structure," Proc. of ASP-DAC, pp.265-270, 1997.

[9] K.Sakanushi, S.Nakatake and Y.Kajitani, "The multi-BSG: stochastic approach to an optimum packing of convex-rectilinear blocks," Proc. of the IEEE/ACM ICCAD, p.267-274, Nov. 08-12, 1998.