O-tree 표현법과 Simulated Annealing 기법을 이용한 효과적인 플로어플랜

박재민*, 허성우**
*동아대학교 컴퓨터공학과
e-mail: *parkings88@dreamwiz.com,**swhur@dau.ac.kr, corresponding author

Effective Floorplan using Otree-Reprentation and Simulated Annealing Technique

Jae-Min Park*, Sung-Woo Hur**
*,**Dept. of Comuter Engineering, Dong-A University

요 약

O-tree 표현법을 이용한 기존의 플로어플랜 알고리즘은 결정적 기법에 기반한 것으로써, 회로의 각 모듈을 차례대로 삭제한 후 가장 좋은 다른 위치에 삽입하는 과정을 함으로써 해 공간을 검색해 간다. 이는 모듈을 처리하는 순서에 따라 결과가 결정되는 단점이 있다. 이런 단점을 해결하기 위해 본 논문에서는 Simulated Annealing 프레임을 이용하여 해 공간을 효과적으로 검색하는 방법을 제시한다. 이웃 해를 탐색하기 위한 플로어 플랜의 변형은 매우 단순하면서도 효과적인 두 가지 방법을 사용한다. 첫째 방법은 한 쌍의 모듈을 선택하여 상호 위치를 맞바꾸는 방법이고, 둘째는 임의의 한 모듈을 선택하여 삭제한 후 삽입 가능한 모든 위치 중 임의의 한곳에 삽입하는 연산을 사용한다. 실험 결과는 매우 고무적이다.

1. 서론

플로어플랜이란 복잡한 회로를 구성하는 수많은 모듈들이 회로 내부에서 물리적으로 어떤 위치에 배치될 것인가에 대해 고려하는 회로 디자인 과정중 하나이다. 이 과정은 회로의 최종 형태와 크기에 영향을 미치며, 내부적으로 회로의 성능에도 직접 혹은 간접적으로 영향을 미친다. 플로어플랜의 목적은 전체 회로 크기의 최소화와 내부 모듈간 배선길이의 최소화로 나뉜다. 전체 회로 크기를 작게 만드는 것의 효과는 회로가 이용되는 제품의 소형화를 돕거나 제한된 공간에서 좀더 많은 회로를 직접 할 수 있게 한다. 회로 내부 모듈간 배선길이를 최소화하는 것의 효과는 발열량을 줄이고, 전력소모를 낮추며, 회로의 성능을 향상시킬 수도 있다. 현재 복잡한 전자 회로를 만들어 내는 공정에서 플로어플랜은 중요하게 사용되고 있으며, 플로어플랜의 연구는 상당히 많이 이루어져있다.

O-tree표현법을 이용한 방법[1], SP(Sequence-Pair) 표현법을 이용한 방법[2], BSG(Bounded-Sliceling-Grid)구조를 이용한 방법[3], CBL(Corner-Block-List)를 이용한 방법[4] 등 많은 방법들이 연구되었으며, 각각의 방법들을 개선시킨 다양한 연구가 이루어졌다. 최근엔 기존의 O-tree표현법을 이용한 방법으로는 검색하지 못하는 해 공간을 탐색할 수 있도록 하는 개선된 방법[5]이 발표된 바 있다. 하지만 이 방법 역시 기존의 방법과 유사하게 결정적 알고리즘에 기반한 것으로써, 회로의 각 모듈을 차례대로 삭제한 후 가장 좋은 다른 위치에 삽입하는 과정을 함으로써 해 공간을 검색해 간다. 이는 모듈을 처리하는 순서에 따라 결과가 결정되는 단점이 있다.

이런 단점을 해결하기 위해 본 논문에서는 Simulated Annealing 프레임을 이용하여 해 공간을 효과적으로 검색하는 방법을 제시한다. 이웃 해를 탐색하기 위한 플로어 플랜의 변형은 매우 단순하면서도 효과적인 두 가지 방법을 사용한다. 첫째 방법은 한 쌍의 모듈을 선택하여 상호 위치를 맞바꾸는 방법이고, 둘째는 임의의 한 모듈을 선택하여 삭제한 후 삽입 가능한 모든 위치 중 임의의 한곳에 삽입하는 연산을 사용한다.

본 논문의 구성은 다음과 같다. 2장에서는 플로어플랜을 위한 O-tree 표현법에 대해 간단하게 소개하며, 3장에서는 본 논문에서 새로 제안하는 방법에 대해 설명한다. 4장에서 실험 결과를 보인다.

2. O-tree표현법을 이용 플로어플랜

2.1 O-tree Encoding

O-tree 표현법을 이용한 플로어플랜[1]은 플로어플랜을 표현하기 위해 O-tree라고 하는 구조를 사용한다. n-노드 O-tree라는 것은 n+1개의 노드들을 가지는 순서화된 tree를 말하며 T와 π를 이용하여 표현될 수 있다.

T는 O-tree 노드들의 구조를 나타내며 π는 O-tree 노

드들의 인덱스와 순서를 나타낸다. T는 2n개의 2진수 배열로 이루어져 있으며, π는 n개의 배열로 이루어져 있다. (π는 노드의 인덱스를 저장해야 하므로 정수형 배열을 사용하면 무난하다.)

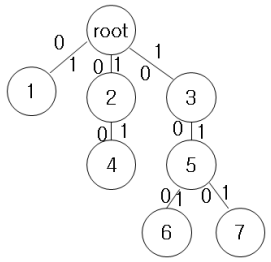
O-tree를 T와 π의 정보로 나타내는 과정을 O-tree Encoding이라 하며 그 방법은 다음과 같다.

Tree의 root부터 DFS(Depth-First-Search)순서로 모든 노드를 검색한다. 검색하는 과정에서 노드와 노드사이의 관계가 부모노드에서 자식노드로 갈 때 T에는 0이 들어가고, 자식노드에서 부모노드로 갈 때 T에는 1이 들어가게 된다. 그리고 모든 노드의 인덱스를 최초로 방문한 순서대로 π에 기입하게 된다. π은 단순하게 방문 순서대로 모든 노드의 인덱스가 하나씩 저장하면 되기 때문에 π는 n-노드 O-tree에서 항상 n개의 정보를 갖고 있다.

T는 root에서 시작된 검색이 다시 root로 돌아오기 위해 약간의 법칙이 존재한다. 최종검색위치가 다시 root로 돌아오기 위해서는 O-tree를 구성하는 T의 0과 1의 갯수가 동일해야 한다. T에서 0이 더 많게 된다면 최종 검색위치가 root가 아닌 그 자손의 노드 어딘가에 머물러 있다는 뜻이 되며, T에서 1이 더 많게 된다면 최종 검색 위치를 root보다 상위에 있는 노드를 의미하는 것이기 때문에 T의 0과 1의 개수가 틀리다면 O-tree를 제대로 표현하지 못한 것이라고 할 수 있다. T에서 표현하는 이진수는 결국 노드와 노드사이의 간선을 표현한 것이고, 간선은 각각 상향할 때와 하향할 때 한번씩 만 사용된다.

O-tree에서 노드들 사이에 간선은 모두 하나만 존재하기 때문에 T 배열은 n-노드 O-tree에서 항상 2n개의 정보를 갖고 있다.

(그림 1)은 7-노드 O-tree가 T와 π로 encoding되는 것을 보여준다.

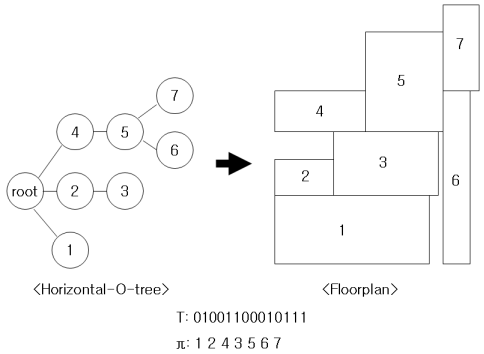


(그림 1) 7-노드 O-tree

2.2 Horizontal O-tree

Horizontal O-tree는 O-tree를 시계 반대 방향으로 90도 회전 시킨 모양이다. Root는 가장 좌측에 위치하게 되며 기존의 O-tree를 회전시킨 것 빼고는 모든 것이 동일하다. Horizontal O-tree는 O-tree를 플로어플랜으로 나타내기 위해 사용한다.

(그림 2)에서 기존의 O-tree(그림 1)를 Horizontal O-tree로 만들고 DFS순서로 노드를 선택하여 플로어플랜을 완성시킨다.



(그림 2) Horizontal O-tree와 플로어플랜

Horizontal O-tree에서 플로어플랜을 완성하는 과정은 다음과 같다.

π의 순서에 따라 모듈을 하나씩 선택한다. 선택된 모듈은 가능한 좌측하단에 위치하게 된다.

순서에 맞게 선택된 노드는 자신의 부모가 되는 노드의 우측에 위치한다. 모든 노드가 선택되어 좌표가 결정되면 최적화 작업을 진행시킨다.

최적화 작업은 배치된 모든 모듈들이 좌측으로 혹은 하단으로 서로 겹치지 않고 수평, 수직 이동할 수 있다면 이동을 시킨다. 좌측끝부분과 하단 끝부분에 위치해 있는 모듈들은 움직이지 않는다.

2.3 결정적 알고리즘과 개선된 결정적 알고리즘

O-tree 표현법에서 이용되는 결정적 알고리즘은 전체 회로에서 하나의 모듈을 빼내고 다른 위치에 이동시키는 것의 반복으로 이루어진다[1]. 모든 모듈 각각을 한 번씩 빼내어 넣을 수 있는 가능한 위치에 차례로 넣어본 후 가장 좋은 위치에 배치시키는 것이 결정적 알고리즘의 아이디어이다. 결정적 알고리즘의 구체적인 동작은 다음과 같다.

π에서 빼낼 모듈을 삭제하고 나머지 모듈들의 인덱스로만 π를 정렬 한다. T에서도 마찬가지로 빼낼 모듈에 관계된 0과 1을 빼어낸 후 나머지 모듈들과 관련된 0과 1들로만 정렬 한다. 이렇게 모듈 하나를 들어낸 후, O-tree에서 단말의 위치에 넣을 수 있는 모든 가능한 위치에 삽입해 본 후 결과가 좋은 위치에 배치시킨다. [1]에서 제안한 기법에 따르면 새로 삽입되는 모듈은 O-tree에서 단말의 노드로만 존재할 수 있게 된다. 이 제약은 해 공간의 일부를 전혀 탐색해 볼 수 없게 만들어 좋은 해를 찾을 확률이 줄어 드는 단점이 있다. 이런 단점을 보완한 개선된 결정적 알고리즘[5]은 새로운 모듈이 단말이 되지 않는 위치에도 삽입이 가능하다. 기존 O-tree 노드와 노드 사이에 삽

입 될 수 있지만 이들의 원래 O-tree에서 조상-자손관계가 여전히 변하지 않는 위치에만 삽입될 수 있다.

예를 들어, 8개의 모듈로 구성된 플로어플랜에서 모듈 0을 삭제한 후 형성된 T가 01001100010111이고, π는 1243567이라고 가정하자. π에서 각 모듈은 고유번호로 나타내었다. 모듈 0을 삽입하여 새로운 T와 π를 구해야 하는데, 모듈 0가 π에서 가장 앞에 삽입되었다고 하자. 즉, 새로 구한 π′은 {0}1243567이 되는데 (새로 변경된 것을 눈에 띄게 하기 위해 { }로 표시함), 여기에 대응하는 새로운 T′은 다음과 같다.

[1]에 기반한 플로어플랜 결정적 알고리즘을 이용할 경우 T′은 {01}01001100010111 로 한 가지만 허용된다. 반면 [5]에 기반한 플로어플랜 결정적 알고리즘을 이용할 경우 T′은 {01}01001100010111, {0}01{1}001100010111, {0}010011{1}00010111 또는 {0}0100110001011{1} 이 가능하고, 이중 가장 최적이 되는 것을 선택하게 된다.

물론 모듈 0이 π에서 임의의 위치에 삽입되는 각 경우에 대해서 앞에서 설명한 것처럼 대응되는 T가 알고리즘에 따라 다르며, [1]과 [5]에서 제안한 두 기법 모두 모듈 0이 가장 최적의 위치에 삽입되는 것을 선택한다.

3. 제안 알고리즘

[1]에서 제시한 방법과, 이를 개선한 [5]의 방법은 모두 결정적 알고리즘에 기반한 것으로써, 회로의 각 모듈을 차례대로 삭제한 후 가장 좋은 다른 위치에 삽입하는 과정을 함으로써 해 공간을 검색해 간다. 이는 모듈을 처리하는 순서에 따라 결과가 결정되는 단점이 있다.

이런 단점을 해결하기 위해 본 논문에서는 Simulated Annealing 프레임을 이용하여 해 공간을 효과적으로 검색하는 방법을 제시한다. Simulated Annealing 기법을 사용할 경우 가장 핵심적인 것은 효과적으로 이웃 해를 탐색할 수 있는 방법을 찾는 것이다. 즉, 이웃 해를 탐색하기 위한 플로어 플랜의 변형은 매우 단순하면서도 효과적이어야 한다. 본 논문에선 두 가지 방법을 제시한다.

첫째 방법은 한 쌍의 모듈을 선택하여 상호 위치를 맞바꾸는 방법이고, 둘째는 임의의 한 모듈을 선택하여 삭제한 후 삽입 가능한 모든 위치 중 가장 최적의 곳에 삽입하는 연산을 사용한다.

첫째 방법, 즉 한 쌍의 모듈을 선택하여 상호 위치를 맞바꿀 경우는 O-tree 표현법에서 T는 그대로 둔 채, π의 두 모듈을 맞바꾸는 것에 해당된다. 두 모듈 각각에 대해 서로 다른 방향 즉, 90도 회전하여 배치하는 경우와 회전하지 않는 경우 2가지를 고려한다. 그럴 경우 한 쌍의 모듈의 위치를 맞바꿀 경우 4가지의 서로 다른 배치가 존재한다. 그 중 가장 좋은 것을 선택하여 그에 대응하는 T와 π로 기억한다.

둘째 방법, 즉 한 모듈을 삭제한 후 삽입 가능한 모든 위치 중 최적의 곳에 삽입하는 연산을 한다. 이 때, [5]에서 제시한 방법과 다른 점은 삽입 가능한 위치를 찾을 때 순수하게 T의 속성 즉, T의 열에서 임의의 전반부까지 고려할 때 그때까지의 0의 개수는 1의 개수보다 같거나 많아야한다는 점만 고려하여 가능한 위치를 고려한다. 앞 절에서 제시한 예에서 보인 것처럼, 8개의 모듈로 구성된 플로어플랜에서 모듈 0을 삭제한 후 형성된 T가 01001100010111이고, π는 1243567이라고 가정하자. 모듈 0을 삽입하여 새로운 T와 π를 구해야 하는데, 모듈 0가 π에서 가장 앞에 삽입되었다고 하자. 즉, 새로 구한 π′은 {0}1243567이 되는데, 본 논문에서 제시한 방법으로 삽입이 허용될 경우 여기에 대응하는 새로운 T′은 다음과 같다. {01}01001100010111, {0}01{1}001100010111, {0}01001100010111, {0}010{1}01100010111, {0}0100110{1}0010111, {0}01001100{1}010111, 또는 {0}01001100010{1}111 이다.

이웃 해를 찾기 위해 첫째 방법, 즉 한 쌍의 모듈의 위치를 맞바꿀 경우 O-tree의 구조는 그대로 유지되는 반면 둘째 방법을 사용할 경우엔 O-tree의 구조가 크게 변형이 된다. 다시 말해 기존의 플로어플랜에 가해지는 변형이 크게 된다.

Simulated Annealing 프레임 안에서 이웃 해 탐색을 위한 두 가지 기법은 실행 시 파라메터로 제시한 값에 따라 선택 비율이 결정되는데, 본 논문에선 첫째 방법이 95%, 둘째 방법이 5%의 비율로 적용되었다.

4. 실험 결과

<표 1> 실험 결과

회로	방법	면적(mm^2)			개선율 (%)	평균 시간(초)
		최대	최소	평균		
ami49	[1]	41.46	39.13	40.54		5.799
	[5]	40.25	38.07	39.10	3.54	87.069
	본논문	39.13	36.74	37.56	3.94	26.012
ami33	[1]	1.36	1.29	1.33		1.894
	[5]	1.32	1.24	1.28	3.88	19.686
	본논문	1.28	1.20	1.22	4.69	14.270
hp	[1]	13.78	9.20	11.60		0.140
	[5]	13.02	9.22	11.18	3.62	0.576
	본논문	9.90	9.11	9.35	16.37	2.918
xerox	[1]	22.46	20.31	21.43		0.114
	[5]	21.88	19.91	20.91	2.42	0.453
	본논문	21.07	19.91	20.29	2.97	2.336
apte	[1]	51.69	47.31	49.19		0.089
	[5]	50.82	47.30	48.39	1.63	0.309
	본논문	50.07	46.92	47.86	1.10	1.989

제시된 알고리즘은 C로 구현되어 MS Window XP SP2, AMD 2800+(2.0Ghz) 1GB의 시스템에서 실험하였다. 플로어플랜의 목적함수는 배선 길이를 제외하고 회로의 면적에 대해서만 고려하였다.

MCNC 밴치마크 회로의 각각에 대해 [1]에서 제시한

방법, [5]에서 제시한 방법, 그리고 본 논문에서 제시한 방법으로 각각 100번씩 결과를 구하여, 최대, 최소, 평균 면적과 실행시간을 보였다. <표 1>의 실험결과에서 개선율은 평균 면적에 대한 개선율이며, [5]에서의 개선율은 [1]에서 구한 결과에 대한 개선율이며, [본 논문]란에 보인 개선율은 [5]의 방법을 기준으로 한 개선율이다.

결과가 보여 주듯이 기존의 두 방법[1,5]에 비해 본 논문에서 제안된 방법으로 구한 결과는 매우 우수하다. 특히 [5]의 결과와 비교할 때, 크기가 큰 회로에 대해 실행시간 면에서도 매우 좋은 결과를 보여주고 있다.

5. 결론

본 논문에서 제안한 알고리즘은 O-tree 표현법으로 나타낸 플로어플랜에 기존의 결정적 알고리즘이 가지는 단점을 해결하기 위해 Simulated Annealing 기법을 적용하여 해 공간을 효율적으로 탐색할 수 있는 기법을 제시하였다. 이웃 해를 찾기 위한 두 가지 플로어플랜 변형 함수는 단순하면서도 매우 효과적임이 실험을 통해 입증되었다. 실험 결과는 기존의 O-tree 표현법을 이용한 결과에 비해 매우 우수하다.

참고문헌

[1] P. -N. Guo, C. -K. Cheng, T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," Proc. 36th ACM/IEEE Design Automation Conf. ,pp. 268-273,June 1999.

[2] H. Murata and E. Kuh, Sequence-pair based placement method for hard/soft/pre-placed modules, Proc. ISPD, pp.167-172, 1998.

[3] S. Nakatake, H. Murata, K. Fujiyoshi, Y. Kajitani, "Module placement on BSG-structure and IC layout application" in: Proc. of International Conference on Computer Aided Design, 484-490, 1996.

[4] X. Hong et al. Corner block list: An effective and efficient topological representation of non-slicing floorplan, Proc. ICCAD, pp.8-12, 2000.

[5] 박재민. 허성우. "O-tree 표현법을 이용한 개선된 플로어플랜 알고리즘" 정보처리학회 학술발표논문집. 제34권 제1호 B, pp.482-486, 2007.

[6] Y. Pang, C.K. Cheng T. Yoshimura, "An Enhanced Perturbing algorithm for Floorplan DesignUsing the O-tree Representation", ACM Proc. ISPD, pp. 168-173, 2000.