CBL 에 기반한 Contour map 을 이용한 플로플랜 기법

오은경*, 허성우**
*동아대학교 컴퓨터공학과
**동아대학교 컴퓨터공학과
e-mail : ekoh@donga.ac.kr* ,swhur@dau.ac.kr**

A Floorplan Technique Based on CBL using Contour map

Eun-Kyung Oh*, Sung-Woo Hur**
*Dept. of Computer Science, Dong-A University
**Dept. of Computer Engineering, Dong-A University

요 약

CBL[1](Corner Block List)에 기반한 Non-Slicing 플로 플랜 알고리즘은 빈 공간이 없는 Non-Slicing 플로플랜만 나타낼 수 있다. 본 논문에서는 CBL 단점을 보완하고 실제 블록의 크기를 이용하여 최적의 위치에 블록을 배치 하기 위해 contour map 을 이용할 것을 제시한다. 본 알고리즘은 배치시 면적을 최소화 하는 방법을 제시하므로 CBL 의 단점을 해결하고 더불어 최적해를 찾기 위한 실행 시간을 단축 시키는 효과를 기대할 수 있다.

1. 서론

플로플래닝[2]은 VLSI 회로의 물리적 설계에서 아주 중요한 단계이다. 플로플랜의 가장 중요한 요소중의 하나가 표현 구조에 있다. 이것은 최적화 알고리즘의 효율성에 직접적으로 영향을 준다. 왜냐하면 표현 구조는 위상 구조로부터 배치를 만들어 내는 복잡도를 결정하기 때문이다.

플로플랜은 Slicing 구조와 non-Slicing 구조로 나뉜다. Slicing 구조에는 이진 트리 표현법을 사용한다[3]. non-Slicing 구조를 다루는 방법으로는 sequence pair[4] 와 bounded-slicing grid[5]가 이다.

최근, non-slicing 플로플랜을 위해 코너 블럭 리스트 (Corner Block List-CBL) 라 불리는 방법이 제시되었다[1]. 이 방법은 하나의 CBL 이 주어지면 이에 대응하는 블록 배치를 얻기 위해서 O(n) 시간이 필요하고, 그 구현이 용이하여 블록 배치에 많이 사용되고 있다. 그러나 CBL 의 단점은 non-slicing 플로플랜의 일부분만을 표현 할 수 있다는 것이다. 왜냐하면, CBL 은 회로를 n 개의 공간으로 분할하고 각각의 공간에 하나의 블록을 배치하는 방식이기 때문에 빈 공간을 표현 할 수 없다. 이는 제한적인 해공간을 갖게 되고 최종적으로 최적의 결과를 가지지 못할 수 있다.[6]

최근에 이 문제를 해결하기 위한 방법들이 제시되었다. ECBL (Extended CBL)[6] 에서는 크기가 없는 가상의 블록을 일정 배수만큼 리스트 상에 추가하고 함께 블럭 배치를 하여 가상의 블록들이 빈 공간의 역할을 하도록 하였다. 그리고 TBS(Twin Binary Sequence)[7] 에서는 이 가상의 블록을 필요한 만큼만 포함하도록 하는 방법을 제시하였다.

본 논문은 새로운 배치 방법을 제시한다. 제안된 방법은 가상의 블록 정보 없이 배치 블록의 경계인 contour map 정보를 이용하는 방법을 제시한다. 블록의 삽입 시 contour map 을 이용하여 최적의 위치를 찾으므로 빈 공간의 면적과 발생 빈도를 최대한으로 줄이는 것이 본 논문에서 제시한 방법의 최대 목표가 된다. 최적의 위치에 블록을 배치함으로 CBL 이 변경이 되기 때문에 simulated annealing 실행 시간을 단축시켜 빠른 시간에 최적의 해에 도달 할 수 있게 된다. 또한 CBL 의 위상정보를 포함하는 constraint 그래프를 사용하지 않으므로 메모리 절약을 기대할 수 있다.

서론에 이어 다음 2 장에서는 CBL 에 대한 정의와 블록배치 방법을 설명하며, 3 장에서는 본 논문에서 제안한 블록 배치 기법에 대해 소개하고 4 장에서 결론을 맺는다.

2. CBL(Corner Block List)

2.1 모자익 플로 플랜

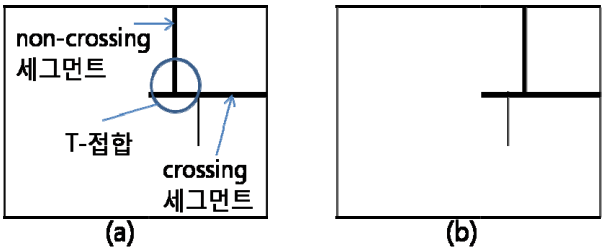
CBL 은 모자익 플로플랜이라는 일반적인 플로플랜의 간략화된 형태를 모델로 하고 있다. 모자익 플로플랜에 대한 정의는 다음과 같다.

(1) 플로플랜 상에는 빈 공간이 없다.
(2)T-접합의 non-crossing 세그먼트를 crossing 세그먼트를 따라 이동한 결과는 이동 전과 동일한 위상으로 정의한다. (그림 1)
(3) 두 T-접합은 같은 점에서 만나지 않는다.

CBL 은 모자익 플로플랜에서 블록들의 위상적인

** Corresponding Auther

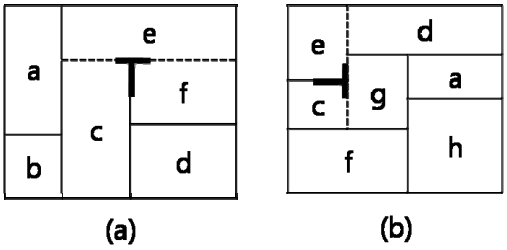
관계를 (S, L, T)의 세 개의 리스트로 정의하며, 이 CBL 에 따라서 칩 영역은 사각 공간들로 분할되고 각 공간에는 단 하나의 블록이 배치된다
그림 1 의 (a)와 (b)는 동등한 위상을 가진다.



(그림 1) 모자익 플로플랜의 동등한 위상

2.2 CBL 정의

코너블록은 오른쪽 위 모서리에 있는 공간에 배치되는 블록을 말한다. 그림 2 의 (a)에서는 e 블록, (b)에서는 d 블록이 코너블록이 된다.



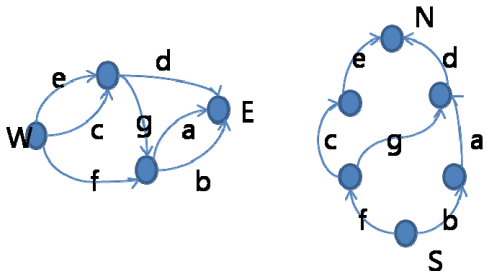
(그림 2) CBL 을 이용한 플로플랜의 예

2.2.1 Constraint 그래프

그림 2 에서 보듯이 오른쪽 위 모서리에 해당하는 정보와 T-접합을 이용한 블럭 배치를 위한 위상 구조로 constraint 그래프를 사용한다. 그래프는 G= (V, E)로 나타내고 여기서 V 는 세그먼트, E 는 배치 공간의 블록을 나타낸다.

Constraint 그래프는 수평 constraint 그래프(HCG) 와 수직 constraint 그래프(VCG) 로 나타낸다.

그림 3 은 그림 2 의 (b)배치에 대한 constraint 그래프를 나타낸다.



(그림 3) constraint 그래프

수평 constraint 그래프에서 E 로 가는 간선과 수직 constriant 그래프에서 N 로 가는 간선에 공통으로 있는 간선이 코너 간선이 되고 이 코너 간선이 코너블록이 된다.

2.2.2. 코너 블록의 방향

코너 블록의 방향은 블록의 왼쪽과 아래쪽의 세그먼트의 결합에 따라 정의한다. 그림 2-(a) 의 경우 블록 e 의 방향은 crossing 세그먼트가 아래쪽에 있고 "T"가 90 도 회전된 모양의 접합을 가지므로 수직 방향이 되고 "0"으로 표기하고, 그림 2-(b) 에서 g 블록 의 경우 접합부의 모양이 180 도의 "T" 이고 왼쪽에 crossing 세그먼트가 있으므로 수평방향이 되고 "1"로 표기한다.

2.2.3 CBL 구성

CBL 은 반복적인 코너블럭의 삭제로부터 생성된다. 각 구성 요소는 다음과 같다.
(a) S : 삭제되는 블럭 네임의 순서
(b) L : 삭제시 코너블럭의 방향정보
(c) T : 삭제시 crossing 세그먼트의 개수

2.3 CBL 와 플로플랜간의 전환을 위한 알고리즘

다음은 코너 블럭 리스트와 플로플랜가 서로 전환되는 알고리즘을 설명한다. 알고리즘 1 은 플로 플랜으로 부터 코너블럭을 생성하는 알고리즘이다.
알고리즘 2 는 코너 블럭 리스트로 부터 플로플랜을 얻은 과정을 나타낸 것이다. 알고리즘 2 의 결과는 해가 존재한다면 모자익이 된다.

```
알고리즘 1.
while 코너블럭이 사용 가능할 동안
{
  (a)코너 블럭의 삭제
  (b)코너 블럭이 마지막이 아닐 경우 CBL 기록
     마지막일 경우 블럭 이름만 추가. 삭제 순서
     의 역순으로 리스트 작성
}
```

```
알고리즘 2.
(a) S[1] 의 블록으로 초기화
(b) for i =2 to n do
     블록 S[i]를 L[i]의 방향에 따라 crossing 세그먼
     트를 T[i]에 해당 되도록 삽입
     플로 플랜이 만약 T[i]의 조건을 만족 시키지
     못하면 exit
      report error
```

2.4 플로플랜과 블록 배치 알고리즘

CBL 을 이용한 플로 플랜 알고리즘은 simulated anealing 에 기초한다. 온도와 최적해를 초기화 하고, 평형상태가 될 때까지 해의 이동을 반복하면서 최적해를 찾는다. 각 이동으로 얻은 이웃해는 블록 배치를 통해서 면적 비용을 계산하고 계산된 비용과 현재 온도를 참고하여 받아들일지 여부를 결정하게 된다.

2.4.1 이웃 해를 얻는 방법

CBL 의 정보중의 하나를 변형하여 이웃해를 구한

다. 정보를 변형하는 방법은 아래와 같다.

(1) 임의의 i,j 에 대하여 S[i] 와 S[j]을 교환
(2) T[i]의 1 의 비트를 0 으로 또는 1 비트를 추가
(3) 블록회전 시키기
(4) 블럭 방향을 변경하기

2.5 CBL 의 문제점

알고리즘 상에 나타나는 문제점은 2 가지가 있다.

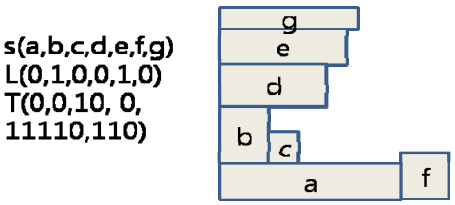
첫째는 2.3 에서 소개된 알고리즘 중 알고리즘 2(b)의 프로그램 중단이다. 이웃해를 얻기 위해 CBL 을 변경 했을 때 crossing 세그먼트의 조건이 맞지 않으면 이웃해를 구하지 못하는 결과가 발생한다. 이는 실행시간을 낭비하는 결과를 초래하게 된다.

두번째는 알고리즘 2 의 결과가 모자익이라는 것이다. 이것이 의미하는 바는 실제적인 블록의 크기를 고려한 최적화가 아닌 것을 뜻한다. 그림 4 는 두번째 단점에서 발생할 수 있는 결과를 보여준다.

f 블럭 삽입시 수평삽입이고, T 정보가 11110 이므로 a 블럭 바로 옆 위치에 삽입된다. 실제로 빈 공간이 많아서 개선의 여지가 많아 보인다.

본 논문에서는 이런 경우 특히 블럭의 크기가 많이 차이가 나는 경우에 발생하는 빈 공간을 효율적으로 줄이기 위해 contour map 을 이용하는 방법을 제안한다. CBL 의 알고리즘을 그대로 사용하면서 플로 플랜으로 전환할 때 블록 배치 방법만 제안된 논문의 알고리즘으로 적용시킨다.

제안된 배치 알고리즘은 다음 3 장에서 설명하도록 한다.



(그림 4) CBL 을 플로플랜으로 전환

3. 제안기법

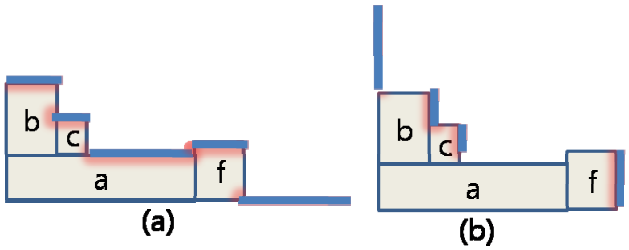
3.1 Contour Map

본 논문에서는 블록의 위상 정보의 보관을 위해 수직 contour map 과 수평 contour map 을 정의 한다.

수직 contour map 은 플로플랜시 위에서 투사했을때 시야내에 들어오는 블럭들의 위치와 정보를 보관한다. 동일한 방법으로 수평 contour map 은 블럭의 오른쪽에서 투사했을 때 시야내에 들어 오는 블럭들의 위치와 정보를 보관한다.

수직 contour map 의 자료 구조는 더블 링크 리스트를 사용하며 정렬 기준은 블럭의 너비 위치가 끝나는 x 값의 오름 차순이다. 동일 한 방법으로 수평 contour map 의 저장 순서는 블럭의 높이의 위치가 높은 y 축 값의 오름 차순으로 저장한다.

그림 5(a)는 수직 contour map, 5(b)는 수평 contour map 을 나타낸다.



(그림 5) contour map

그림 5 의 경우는 수직 contour map 의 저장을 위한 자료구조는 다음과 같다.

3.2 CBL 을 플로플랜으로 전환

3.2.1 코너 블럭 설정방법

제안 기법에서는 코너 블럭의 개념을 다르게 정의한다. 코너 블럭을 설정하는 방법은 아래의 규칙을 따른다.

가정 1 : S[i]의 블럭의 수직 방향 삽입시
1. S[i-1]의 블럭이 수직 contour map 일 경우
   코너 블럭 → S[i-1]
2. S[i-1]의 블럭이 수직 contour map 이 아닐 경우
   코너 블럭 → 수직 contour map 에서 가장 오른쪽 끝에 있는 블럭

가정 2 ; S[i]의 블럭의 수평 방향 삽입시
1. S[i-1]의 블럭이 수직 contour map 일 경우
   코너블럭 →S[i-1]
2. S[i-1] 의 블럭이 수직 contour map 이 아닌 경우
   S[i-1]이 수평 블럭일 경우
   코너블럭 →S[i-1]
   S[i-1]이 수평 블럭이 아닌 경우
   코너블럭 → 수직 contour map 에서 가장 오르쪽 끝에 있는 블록

위의 방법을 이용하여 코너 블럭이 결정되면 다음 절에서 설명하는 방법으로 블록을 배치한다.

3.2.2. 수직 블럭 삽입 방법

S[i] 블럭의 수직 삽입 방법은 알고리즘 3 에 따른다.

알고리즘 3.

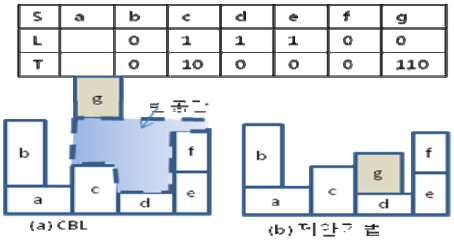
while(T[i] 의 값이 유효할 동안 반복)
{
  코너 블럭으로 부터 T[i]의 조건을 만족 시키는 블럭 을 찾을 동안 한 단계식 수직 contour map 왼쪽으로 이동한다.
  이때 각 contour map 에서의 S[i]의 블럭을 수직 삽입하여 최고의 비용(최소면적)을 갖는 위치를 기

억한다
If (crossing 세그먼트의 수<T[i]의 1 의비트의 수)
Break;
}
최고의 위치로 블럭을 배치하고 T[i]의 정보를 변경하여 저장한다.

알고리즘 3 의 방법대로 수직 배치를 할 경우 접합 부분의 세그먼트의 수는 다소 변경이 된다. 그러나, 플로 플랜 과정에서 알고리즘 2-(b) 와 같이 과정 자체를 중단하는 현상은 발생하지 않으며, 오히려 배치 과정 중 가장 최적의 위치로 배치가 되는 효과를 가져 올 수 있다. 실제로 최적의 위치로의 배치는 simulated annealing 을 거쳐야 가능하지만 제시한 알고리즘은 배치 과정 중에 최적의 위치를 찾고, 모자익의 특성상 발생하는 빈 공간의 발생 면적을 최소로 줄일 수 있다.

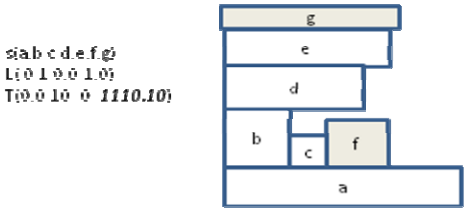
그림 6 은 CBL 과 제안 기법과의 배치 결과를 보여준다. g 블록을 배치 할 때 접합의 개수에 따라 CBL 에서 가능한 배치는 (a)의 결과로 빈 공간이 생성된다. 이 빈 공간은 simulated annealing 의 결과 사라질 가능성도 있다. 그러나 제안 기법에서는 T 를 고려한 배치 과정 중 최적의 위치로 (b)와 같은 배치 결과를 얻을 수 있다. 이 배치 결과는 g 를 수직 삽입 했을 때 얻을 수 있는 최적의 위치임을 알 수 있다.

T[7]의 정보를 배치 후 110 에서 10 으로 수정한다.



(그림 6) 블록 배치의 비교

동일한 방법으로 그림 7 을 수평 삽입시의 배치결과를 보여준다. CBL 은 그림 4 의 CBL 을 참조한다. 제시한 기법의 결과 블록 f 의 위치가 블록 a 옆이 아닌 c 옆에 수평 삽입 된 것을 확인 할 수 있다. 블록 f 위치 역시 T 접합 정보를 최대한 보존하면서 배치 가능한 최적의 위치가 된다. 그림 7 에서 블록 g 의 배치 결과가 CBL 배치 결과와 다르다는 것을 확인 할 수 있다. T[6]의 값이 11110 →1110 으로 변경되었고, 더불어 블록 e 의 T 값도 110 →10 으로 변경되었다.



(그림 7) 그림 4 의 블록 배치의 비교

4. 결론

제시한 알고리즘은 CBL 을 이용하여 블록 배치시에 발생하는 빈공간을 최대한 줄이고자 고안한 알고리즘이다. Constraint 그래프 없이 contour map 만 유지하더라도 수직/수평 방향으로 삽입 시 수직/수평 compact(압축 – 빈공간이 없도록 배치)된다. 따라서 빈공간의 발생을 최대한으로 줄일 수 있다.

참고문헌

[1] Hong Xianlong, Huang Gang et al. “Corner Block List : An Effective and Efficient Topological Representation of Non-slicing Floorplan” ICCAD’2000.
[2] Ralph H. J. M. Otten “What is a floorplan”, ISPD’2000.
[3] D. F. Wong, C. L. Liu, “A new algorithm for floorplan design”, in Prec.of 23rd ACM/IEEE DAC, 101-107, 1986.
[4] S. Nakatake, H. Murata, K. Fujiyoshi and Y. Kajitani, “Block Placement on BSG-structure and IC layout application”, in Proc. Of International Conference on Computer Aided Design, pp 484-49-, 1996.
[5] Hiroshi Murata, K. Fujiyoshi ,S. Nakatake and Y. Kajitani, “VLSI Block Placement Based on Rectangle-Packing by the Sequence Pair” in IEEE Trans. On CAD, vol. 15, pp 1518-1524, 1996.
[6] Shu Zhou, Shequin Dong, Yici Cai, Chung-Kuan Cheng, Jun Gu, “ECBL : An Extended Corner Block List with Solution space including Optimum placement”, in Proc. IEEE/ACM International Symposium on Physical Design, 2001.
[7] F.Y. Young, C. N. Chu and Zion Cien Shen, “Twin Binary Sequence: A Non-Redundant Representation for General Non-Slicing Floorplan”, in Proc. IEEE/ACM International Symposium on Physical Design, pp. April 2001.