배선밀집도 드리븐 배치

오 은 경[†] · 허 성 우[††]

요 약

본 논문에서는 주어진 상세배치의 배선밀집도를 예측하고, 배선밀집도가 높은 지역을 효과적으로 해결하는 새로운 알고리즘을 소개한다. 제안된 기법의 특징은 다음과 같다. 첫째, 특정지역을 관통하는 넷이 많아 그 지역의 배선밀집도가 높을 경우 관통하는 넷에 연결된 셀들을 효과적으로 찾아내어 그들의 위치를 조정함으로써 배선밀집도를 완화시킨다. 둘째, 리플이동 (ripple movement) 기법을 이용하여 셀들을 이동함으로써 배선길이를 희생시키지 않으면서 배선밀집도를 완화시킨다. 셋째, 셀의 이동에 따른 배선밀집도의 변화 및 배선길이의 변화를 효과적으로 추적할 수 있도록 증분 자료구조 (incremental data structure)를 사용하였기 때문에 실시간으로 변화되는 상황에 적합한 셀 선택이 이루어지며, 실행시간도 매우 빠른 장점이 있다. 마지막으로, 본 논문에서는 MST 넷 모델을 사용하여 배선밀집도를 예측하지만 제안한 기법은 특정 넷 모델에 상관없이 적용할 수 있다. 특히 실제 라우터로부터 배선정보를 얻을 수 있다면, 배선밀집도는 더욱 효과적으로 해결될 수 있다.

제안된 기법을 이용한 실험결과는 배선길이를 희생하지 않으면서 배선밀집도를 매우 효과적으로 개선할 수 있음을 보여준다.

키워드 : 배선밀집도, 배선길이

Routing Congestion Driven Placement

Eun Kyung Oh[†] · Sung Woo Hur[††]

ABSTRACT

This paper describes a new effective algorithm to estimate routing congestion and to resolve highly congested regions for a given detailed placement. The major features of the proposed technique can be summarized as follows. Firstly, if there are congested regions due to some nets which pass through the regions it can determine which cells affect those congested spots seriously and moves some of them to resolve congestion effectively. Secondly, since the proposed technique uses the ripple movement technique to move cells it resolves congestion without sacrificing wire length. Thirdly, we use an efficient incremental data structure to trace the changes in congestion and wire length as cells move. Hence, selection of cells to move could be very accurate and fast in the course of iteration. Finally, although an MST net model is used to resolve congestion in this paper, proposed technique can be work with any net model. Particularly, if proposed technique can obtain routing information from a real router, congestion can be resolved more effectively.

Experimental results show that the proposed technique can resolve congestion effectively and efficiently without sacrificing wire length.

Key Words : Congestion, Wire Length

1. 서 론

VLSI 기술이 향상됨에 따라 시스템의 복잡도가 증가하게 되고 물리설계의 단계 또한 점점 더 복잡해지고 있다. 특히 셀 배치는 배선길이, 성능(performance) 그리고 배선가능성(routability)과 같은 주요한 설계변수에 큰 영향을 미친다. 특히 배선가능성을 고려하지 않은 배치는 결국 배선 과정에서 심각한 문제를 야기하게 되고 결국은 배치 자체를 다시 해야만 할 경우도 발생시킨다.

전통적인 배치의 목표는 컷 사이즈를 줄이거나 배선길이를 최소화 시키는 것으로써 이에 대한 연구는 오랫동안 되어 왔다. 최소-컷을 찾는 기법을 사용하는 배치 툴로는 Capo[1], Quad[2], FengShui[3], Snap-on[4] 등[5]이 있다. Caldwell[6]이 지적한대로 이 기법을 사용하는 툴들은 대부분 변형된 KL알고리즘[7]이나 FM알고리즘[8]을 이용하고 있으나 최소-컷을 찾는 방법의 구조적인 문제로 인해 셀들을 균등하게 분포시키는 데에는 문제가 있다[9]. 배선길이를 최소화하기 위해 simulated annealing 기법을 사용하는 툴로는 TimeberWolf[10]가 잘 알려져 있으며, 해석적 기법 또는 FD(Force-Directed) 기법을 사용하는 툴들도 있는데, FD 방법을 사용하는 배치기는 그 변화가 매우 다양하다. Goto는 iterative FD 방법을 이용한 독특한 알고리즘을 일찍이 제안

※ 본 논문은 2002학년도 정보통신부 IT학과 장비지원사업의 동아대학교 대응자금에 의해 연구되었음
† 준 회 원 : 동아대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 동아대학교 전기전자컴퓨터공학부 교수
논문접수 : 2005년 10월 7일, 심사완료 : 2006년 2월 9일

하였다[11]. 셀 중복을 가능한 줄이기 위해 인장력뿐만 아니라 척력 (repelling force)을 이용한 기법도 소개되었다[12, 13]. 이 외에도 배선길이를 최소화하기 위해 다양한 복합기법을 사용하는 툴들도 있다[14-16]. 배선길이만을 최소화하기 위한 배치기법에선 배선가능성에 대한 고려가 거의 배제되어 있다. 최소-컷을 찾는 기법이나 배선길이를 최소화하기 위한 기법은 거시적인 관점에서 볼 때 배선의 밀집도를 줄일 수 있으나 국부적으로 발생되는 높은 배선밀집도를 예방하지는 못하는 문제가 있다.

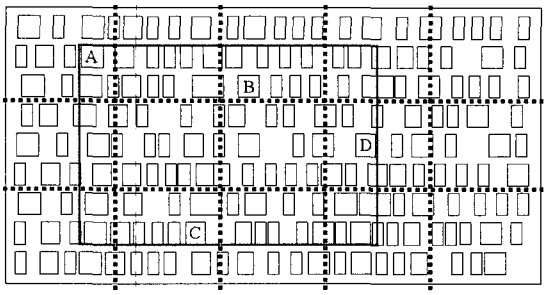
배선밀집도를 어떻게 추정하고 이를 어떻게 줄일 것인가에 대한 연구는 배선길이를 줄이고자 하는 연구에 비해 상대적으로 많이 되어 있지 않다. Meixner[17]는 다단계 분할을 이용한 방법을 제안했는데, 여기서는 배선밀집도를 줄이기 위해 미리 구성된 Steiner 트리로부터 실제로 계산된 배선밀집도를 사용했다. 하지만 여기선 과도한 계산의 부하를 피하기 위해 분할횟수가 제한되어 있다. Wang 등[18-20]은 요구/공급(demand/supply) 관계를 이용하여 일관된 배선 모델을 제안하였다. 그러나 실험결과에 의하면 배선밀집도 만을 고려한 배치는 매우 나빴다. 그래서 우선 배선길이를 목표로 배치를 구하고, 나중에 배선밀집도를 고려하여 배치를 수정하였다. 배선길이 최소화 작업을 하는 이유는 앞에서 이미 언급한대로 거시적인 관점에서 볼 때 배선밀집도를 개선시키기 때문이다[18, 21]. Wang 등이 사용한 바운딩 박스에 의한 배선밀집도 추정은 너무 단순하여 실제 배선과는 큰 차이가 나는 문제가 있다. 배선밀집도와 배선길이는 거시적으로 밀접한 관계가 있기 때문에 Jason 등[22, 23]은 증분 배치(incremental placement)와 더불어 국부적으로 배선밀집도 문제를 해결할 것을 제안하였다. 하지만 이럴 경우에는 배선밀집도 개선을 위해 배치를 변형할 때 배선길이도 같이 변하는데 이런 변화의 추적이 쉽지 않은 문제점이 있다. Parakh[25]는 영역확장(region expanding) 기법을 사용하여 배선밀집도를 해결하는 기법을 제안하였다. 하지만 이 기법은 둘 이상의 영역에 대해 밀집도가 진동하기 쉬운 문제점이 있다. Wang[19]은 플로우-기반(flow-based) 셀-중심(cell-centric) 알고리즘을 사용하여 셀을 이동함으로써 밀집도를 최소화하였는데 이는 라우터에 너무 심하게 의존하는 문제점이 있다. [24]와 [26]에서는 배선밀집도가 높다고 판단된 영역에 있는 셀들에 대해선 크기를 가상적으로 키움으로 그 영역에 있는 셀들의 개수를 상대적으로 줄이는 방법으로 배선밀집도를 해결하였다. 이 방법에선 그 영역을 관통하는 넷에 의해 배선밀집도가 높아질 경우 정확하게 처리하지 못하는 문제점이 있다. 배선을 위해 어떤 넷 모델링을 사용할 것인가 하는 것도 배선밀집도와 배선길이에 큰 영향을 미친다. [27]에서는 배선을 위해 RMST(Rectilinear Minimum Spanning Tree) 모델을 사용하여 확률에 근거한 배선 요구를 예측하였는데 이는 RSMT(Rectilinear Steiner Minimum Tree) 모델을 이용하는 것과 실제적으로 큰 차이가 없음을 보였다. [28, 29]에서는 배선을 위해 흔히 사용되는 RSMT 넷 모델에서 넷 길이를 빨리 계산할 수 있는 효과적인 방법을 제시하였다.

본 논문에서는 [18-20, 30, 31]에서처럼 주어진 상세배치를 바탕으로 배선밀집도를 예측하고 배선밀집도가 높은 지역을 국부적으로 해결하는 새로운 기법을 제안한다. 배선밀집도 예측을 위해서는 MST 넷 모델을 사용하였는데 이는 계산 속도가 빠르면서 실제적인 라우터의 배선과 큰 차이가 없기 때문이다. 제안된 기법의 특징 중 하나는 특정지역을 관통하는 넷이 많아 그 지역의 배선밀집도가 높아질 경우 관통하는 넷에 연결된 셀들을 효과적으로 찾아내어 그들의 위치를 조정함으로써 그 특정지역의 배선밀집도를 완화시킬 수 있다는 것이다. 배선밀집도에 기반하여 셀들을 이동한 후, Mongrel[16]에서 제안한 것처럼 배선길이와 타이밍에 기반한 리플이동(ripple movement) 기법을 이용하여 셀들을 국부적으로 다시 이동시킨다. 이렇게 함으로써 변화된 배치에서 셀의 중첩을 최소화시키고 따라서 결과적으로 얻은 광역배치를 상세배치로 변환하였을 때 배선길이가 최소한으로 희생되는 효과가 있으며, 주어진 입력배치보다 타이밍도 더 나빠지지 않는 효과가 있다. 또 다른 특징은 셀의 이동에 따른 배선밀집도의 변화 및 배선길이의 변화를 효과적으로 추적할 수 있도록 증분 자료구조 (incremental data structure)의 사용이다. 제안된 증분 자료구조는 배선밀집도가 높은 영역에 대해 이를 해결하는데 초점을 맞추어 설계되었기 때문에 매우 효과적이며, 또한 실행시간을 감안하여 모든 넷에 대한 정보를 저장하는 대신 배선밀집도를 개선하는데 효과가 크다고 판단되는 넷들에 대해서만 정보를 저장함으로써 매우 빠르게 실행되는 장점이 있다.

2장에서는 본 논문에서 사용되는 용어를 정의하고 간략히 설명한다. 3장에서는 배선밀집도를 추정하기 위한 계산 방법과 사용하는 넷 모델링에 대해 설명한다. 4장에서는 배선밀집도를 개선할 수 있는 기법을 자세히 설명한다. 5장에서는 제안된 기법에 의해 행해진 실험결과를 보이며, 6장에서는 결론을 맺는다.

2. 용어 정의 및 설명

회로는 하이퍼그래프 $G=(V,E)$로 나타내는데, 여기서 V는 셀들의 집합을 의미하며, E는 넷들의 집합을 말한다. 하이퍼그래프에서 넷 $e\in E$는 2개 이상의 셀들로 구성된 V의 부분집합 즉, $e\subseteq V$이다. 배치(placement)란 일반적으로 직사각형으로 된 칩 영역 내에서 모든 셀들을 위한 위치의 집합으로서, 광역배치(global placement)는 셀들의 중첩을 허용하지만 상세배치(detailed placement)에서는 셀들 간에 중첩이 허용되지 않는다. 광역배선(global routing)이란 한 넷에 연결된 셀들을 추정된 경로에 의하여 최적으로 연결하는 것을 의미한다. 상세배치를 개선하는 과정에서 광역배선을 미리 예측함으로써 배선이 가능한 배치를 얻을 수 있는데, 이때 일반적으로 칩 영역 (또는 코어 영역이라고도 부름)은 $n\times m$ 격자로 나누어진다. 격자의 각 영역을 빈(bin)이라 부른다.



(그림 1) 3×5의 격자에서 빈과 셀 그리고 넷 $e=\{A,B,C,D\}$의 바운딩박스

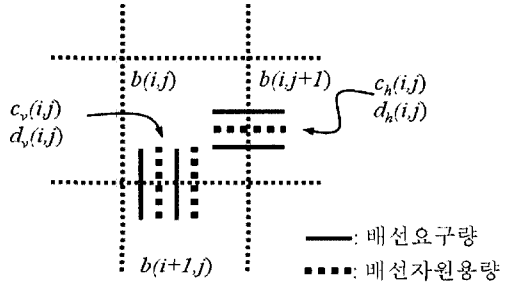
어떤 넷 e의 바운딩박스 (bounding box)란 e에 속한 모든 셀들을 포함하는 가장 작은 직사각형을 나타낸다. 넷 e의 바운딩박스 둘레길이의 1/2을 HP(e)로 나타내는데 여기서 HP는 half perimeter의 약자로, 이를 이용하여 일반적으로 넷 e의 길이를 표현한다. 배치 P의 배선길이는 다음과 같이 계산된다.

$$len(P)=\sum_{e\in E} HP(e)$$

(그림 1)은 3×5의 격자로 분할된 칩 영역에서 상세배치의 한 예를 보여준다. 점선은 각 빈의 경계선을 나타낸다. 네 개의 셀로 구성된 넷 $e=\{A,B,C,D\}$의 바운딩박스는 실선으로 보였다.

배선밀집도를 추정하기 위해선 어떤 영역을 지나가는 배선이 얼마나 될 것인지를 예측해야 하는데 이는 주어진 배치에서 넷에 있는 셀들이 어떻게 연결될 것인가와 깊은 관계가 있다. 배선이 지나가는 개수를 측정하기 위해선 기준 지점이 있어야 하는데 이를 위해 빈 경계선을 이용한다. 즉, 빈 경계선을 통과하는 배선의 개수를 추정하고, 이를 '배선요구량'(routing demand)이라고 부른다. 배선을 위해 사용가능한 자원은 칩의 설계 시에 다른 요인들에 의해 이미 정의되어 있는데, 이를 '배선자원용량'(routing resource capacity)이라고 부른다.

$b(i,j)$는 i번째 행과 j번째 열에 있는 빈을 나타낸다. 빈 $b(i,j)$와 $b(i,j+1)$의 경계선을 통과하는 배선은 수평선이며, 이에 대응하는 배선요구량 및 배선자원용량은 각각 $d_h(i,j)$ 및 $c_h(i,j)$로 나타낸다. 유사하게 빈 $b(i,j)$와 $b(i+1,j)$의 경계선을 통과하는 배선은 수직선이며, 이에 대응하는 배선



(그림 2) 빈 $b(i,j)$의 배선요구량 및 배선자원용량

요구량 및 배선자원용량은 각각 $d_v(i,j)$ 및 $c_v(i,j)$로 나타낸다. (그림 2)는 빈과 배선요구량 및 배선자원용량 관계를 나타낸다. 특정 넷 e에 의해 발생되는 배선요구량의 수평 및 수직성분은 각각 $d_{h_e}(i,j)$, $d_{v_e}(i,j)$로 나타낸다. 그러면 $d_h(i,j)=\sum_{e\in E} d_{h_e}(i,j)$, $d_v(i,j)=\sum_{e\in E} d_{v_e}(i,j)$ 가 된다.

$b(i,j)$에서 수직성분의 배선밀집도(routing congestion) $rc_v(i,j)$와 수평성분의 배선밀집도 $rc_h(i,j)$는 다음과 같이 각각 정의한다.

$$rc_v(i,j)=d_v(i,j)/c_v(i,j)$$
$$rc_h(i,j)=d_h(i,j)/c_h(i,j)$$

그리고 빈 $b(i,j)$의 배선밀집도 $rc(i,j)$는 $rc_v(i,j)$나 $rc_h(i,j)$중 큰 값으로 둔다. 즉,

$$rc(i,j)=\begin{cases} rc_v(i,j), & \text{if } rc_v(i,j)>rc_h(i,j) \\ rc_h(i,j), & otherwise \end{cases}$$

3. 배치에서 배선밀집도 추정

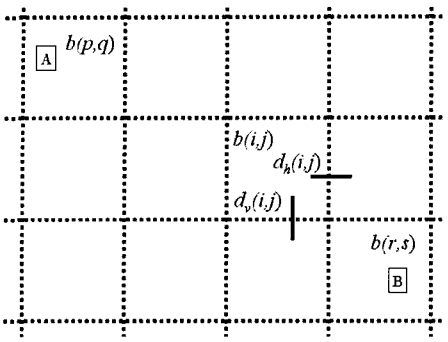
일반적으로 물리설계 과정에서는 배치 후 배선을 하게 되는데 배선과정에서 배선밀집도가 1을 넘는 경우는 우회배선을 해야 하고, 우회배선을 해야 하는 경우가 많으면 타이밍이 아주 나빠지거나 또는 배선 불가능한 경우를 만나게 된다. 이런 경우엔 배치 또는 이전 단계의 처리를 전면적으로 다시 해야 하는데 이는 굉장한 비용을 요구하게 된다. 그래서 배치 과정에서 배선밀집도를 미리 추정하여 밀집도가 높은 곳은 배선밀집도를 낮출 수 있도록 셀의 위치를 다시 조정함으로써 배선과정에서 배선이 불가능한 경우가 발생되지 않도록 미리 예방하고자 하는 많은 연구가 행해져 왔고, 본 연구의 목적도 배치 과정에서 배선을 미리 예측하여 배선밀집도가 높은 곳은 완화시켜 줌으로써 최종배치가 배선가능할 뿐 아니라 필요에 따라 다른 목적함수(예를 들어, 배선길이, 타이밍 등)도 동시에 최적화 시키는 것이다.

3.1 배선요구량 계산

본 절에서는 연결이 필요한 두 셀에 의해 파생되는 배선요구량을 어떻게 계산하는지를 설명한다. 어떤 넷에 연결된 셀이 3개 이상일 때 이 셀들이 어떻게 연결될 것인지에 대한 설명은 3.2절에서 다룬다.

주어진 회로와 칩 파라미터에 따라 각 빈 $b(i,j)$ $(1\le i\le n,\ 1\le j\le m)$에 대해 배선자원용량 $c_v(i,j)$와 $c_h(i,j)$는 미리 결정되어 있다. 배선밀집도를 계산하기 위해선 각 빈의 경계선을 통과하는 넷이 얼마나 되는가를 추정할 필요가 있다. 즉, $1\le i\le n$, $1\le j\le m$에 대해 $d_v(i,j)$와 $d_h(i,j)$를 추정할 필요가 있다.

넷 $e=\{A,B\}$라고 가정하자. 당연히 셀 A와 B를 연결해야 하고, 이 두 셀이 서로 다른 빈에 있다면 이들을 연결



(그림 3) 배선요구량 계산 방법을 보이는 예

하는 방법은 여러 가지가 있다. 특히 배선장애로 인해 우회(detour)하여 연결하는 것까지 고려한다면 그 두 셀을 연결하기 위한 가능한 경로는 무한히 많다. 본 논문에서는 우회연결은 계산의 복잡도 이유로 인해 고려하지 않는다. 실제 라우터도 배선이 불가능한 경우에만 우회연결을 고려하며, 배치 과정에서 그런 우회가능성까지 고려하면서 배선밀집도를 추정한다는 것은 현실성이 없어 보인다.

셀 A는 $b(p,q)$에, 셀 B는 $b(r,s)$에 있고, $1\le p<r\le n$, $1\le q<s\le m$이 만족된다고 가정한다(이 가정은 일반성을 잃지 않는다). 이 두 셀을 연결하는 서로 다른 단조경로(monotone path)의 수는 $\binom{r-p+s-q}{r-p}$이며, 두 빈 $b(p,q)$와 $b(r,s)$ 사이에 있는 모든 빈 $b(i,j)$ $(p\le i\le r,\ q\le j\le s)$에서 두 셀 A와 B를 연결하기 위해 필요한 배선요구량 $d_{v_e}(i,j)$와 $d_{h_e}(i,j)$는 확률에 근거하여 (그림 3)에서 보인 것처럼 다음과 같이 계산할 수 있다. 즉, 셀 A와 B를 연결하는 경로가 $b(i,j)$의 우측 경계선을 통과하기 위해선 빈 $b(p,q)$,..., $b(i,j)$, $b(i,j+1)$,..., $b(r,s)$를 지나가야 한다. 마찬가지로 셀 A와 B를 연결하는 경로가 $b(i,j)$의 아래쪽 경계선을 통과하기 위해선 빈 $b(p,q)$,..., $b(i,j)$, $b(i+1,j)$,..., $b(r,s)$를 지나가야 한다. 따라서 $d_{h_e}(i,j)$ $(p\le i\le r,\ q\le j<s)$는 다음과 같다. (참고로 $b(i,s)$ $(p\le i\le r)$에서는 넷 e에 의해 필요한 수평배선요구가 없기 때문에 $d_{h_e}(i,s)$는 0이 된다.)

$$d_{h_e}(i,j)=\begin{cases}\dfrac{\binom{i-p+j-q}{i-p}\binom{r-i+s-j-1}{r-i}}{\binom{r-p+s-q}{r-p}} & if\ p\le i\le r,\ q\le j<s\\ 0 & otherwise\end{cases}$$

유사하게 $d_{v_e}(i,j)$ $(p\le i<r,\ q\le j\le s)$는 다음과 같다. (유사하게, $b(r,j)$ $(q\le j<s)$에서는 넷 e에 의해 필요한 수직배선요구가 없기 때문에 $d_{v_e}(r,j)$는 0이 된다.)

$$d_{v_e}(i,j)=\begin{cases}\dfrac{\binom{i-p+j-q}{i-p}\binom{r-i-1+s-j}{r-i-1}}{\binom{r-p+s-q}{r-p}} & if\ p\le i<r,\ q\le j\le s\\ 0 & otherwise\end{cases}$$

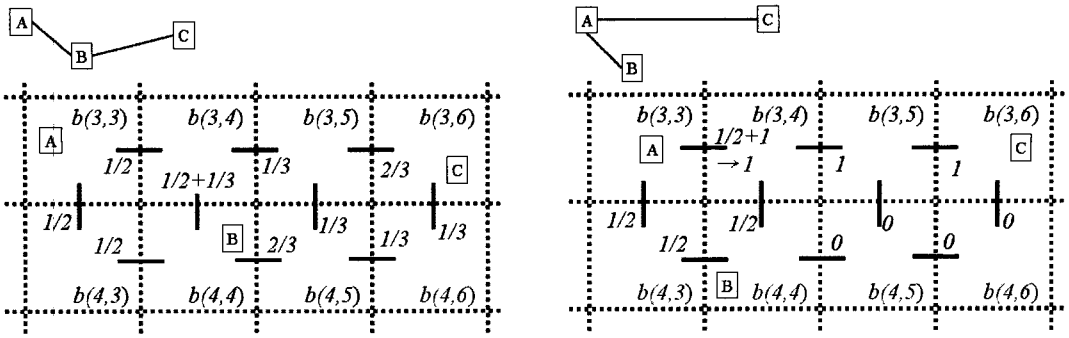
3.2 배선요구량 추정을 위한 넷 모델링

배선요구량 계산을 위해 넷에 있는 셀들이 어떻게 서로 연결되는지를 결정해야 한다. 본 논문에서는 [27]에서처럼 배선을 위해 RMST(Rectilinear Minimum Spanning Tree) 모델을 사용하여 넷에 있는 셀들을 연결한다. 그 이유는, RSMT(Rectilinear Steiner Minimum Tree)모델과 배선결과에선 큰 차이가 없으면서 계산 속도는 빠른 장점이 있기 때문이다. 실제로 넷에 연결된 셀이 2개인 경우 두 모델의 차이는 없는데, 실제 회로에서는 넷의 66% 정도가 셀이 2개인 넷으로 되어있다. 또한 넷에 있는 셀이 3개인 경우도 두 모델의 차이는 거의 없는데 이 경우까지 포함하면 전체 넷의 평균 80%가 3개의 이하의 셀들로 구성되어 있다.

넷에 연결된 셀이 k개인 경우 RSMT 모델을 이용하여 셀을 연결하기 위해선 가능한 $\binom{k}{2}$가지의 셀 쌍에 대해 맨해턴 (Manhattan) 거리를 구한다. (그림 3)에서 두 셀 A와 B의 좌표가 각각 (A_x,A_y), (B_x,B_y)라면 맨해턴거리는 $|A_x-B_x|+|A_y-B_y|$이다. 맨해턴 거리의 합이 최소가 되도록 k개의 셀들을 연결한 스패닝트리가 RMST가 된다. RMST가 결정되면 RMST에 속한 각 에지에 의해 연결된 두 셀들에 의해 파생되는 배선요구량을 3.1절에서 설명한대로 구할 수 있다.

(그림 4)에선 셀 수가 3인 유사한 두 넷의 RMST와 그에 따라 계산된 배선요구량 예를 보였다. 격자 인덱스는 설명의 편의를 위하여 임의로 부여하였다. $e=\{A,B,C\}$라고 가정하자. 세 개의 셀이 서로 다른 빈에 있지만, 두 경우 빈 내에서의 위치가 달라 RMST의 구성이 다르게 결정된다. 그림 상단 부분에 RMST의 구성도를 간략하게 보였다. (그림 4)(a)의 경우, RMST에 포함되는 에지가 (A,B)와 (B,C)이고, 이 각각의 에지에 대해 3.1절에서 설명한대로 수직/수평 성분의 배선요구량을 구하였다. 그 값을 각각 그림에서 보였으며, 특히 $b(3,4)$에서 수직성분의 배선요구량은 두 에지에 의해 파생되는 요구량의 합으로 표현함으로써 이해를 돕고자 하였다. 각 에지에 대해 파생되는 수직/수평 성분의 배선요구량의 합은 1이 됨을 쉽게 알 수 있다. (그림 4)(a)의 경우 넷 e에 의해 각 빈에서 발생되는 배선요구량은 $d_{h_e}(3,3)=0.5$, $d_{v_e}(3,3)=0.5$, $d_{h_e}(3,4)=0.333$, $d_{v_e}(3,4)=0.533$, ..., $d_{h_e}(4,5)=0.33$, $d_{v_e}(4,5)=0$이 된다.

배선요구량 계산 시 유의하여야 할 점 중 하나는 (그림 4)(b)와 같은 경우이다. 이 경우엔 에지 (A,C)에 의해 $b(3,3)$에서 필요한 수평성분 배선요구량은 1이 되며, 또한 에지 (A,B)에 의해 $b(3,3)$에서 필요한 수평성분 배선요구량은 0.5가 된다. 이 둘을 합하면 $d_{h_e}(3,3)$=1.5가 되는데 실제 배선에서 한 넷에 의해 파생되는 배선요구량은 1을 넘지 않는다. 따라서 RMST 모델에 의해 배선요구량을 각 에지별로 계산했다 하더라도, 한 넷에 의해 파생되는 수직/수평 성분의 배선요구량이 어떤 빈에서 1을 초과할 경우 우리는 그 값을 1로 보정해 준다.



(그림 4) 세 개의 셀로 구성된 넷에서 RMST를 구하고, 배선요구량을 계산한 예

4. 배선밀집도 개선을 위한 기법

배선밀집도를 배치 과정에서 개선하기 위해선 배선밀집도 계산 후 어느 영역(빈)에서 배선자원용량보다 배선요구량이 많은지 아는 것도 중요하지만 그 영역에서 어떤 넷에 의해 배선요구량이 많이 발생했는지를 알아내는 것이 매우 중요하다. 배선요구량을 많이 발생시키는 넷을 알면 그 넷에 있는 셀들의 위치를 옮겨 줌으로써 배선요구량을 낮출 수 있기 때문이다.

동일한 빈 내에 있는 셀들이 서로 연결되는 것도 실제로는 배선자원을 사용하기 때문에 이에 의해 발생될 수 있는 배선요구량에 대한 고려도 필요하다. 이런 배선 요구량은 3절에서 설명한 방법으로는 파악할 수 없지만, 빈의 크기가 크지 않다면 빈 내부에서 발생되는 이런 요구량은 국부적이어서 거시적인 관점에서 볼 때 큰 문제가 되지 않는다. 또한 이런 국부적인 배선밀집도는 그 빈 내에 얼마나 많은 셀들이 있는 가에 거의 비례하기 때문에 각 빈 내에 있는 셀의 밀집도를 조절함으로써 가능하다. 셀 밀집도란 각 빈 내부에 있는 셀들의 면적의 합에 빈 면적을 나눈 값으로써 그 값이 1이란 말은 그 빈에 셀로 100% 꽉 찬 것을 의미하며, 1을 초과하는 경우는 셀들이 중첩되어 셀 면적의 합이 빈 면적보다 큰 경우이다. 어떤 셀이 두 개 이상의 빈에 걸쳐 있는 경우, 어느 빈에 어느 정도의 면적을 차지하는지를 정확하게 알 수 있기 때문에 걸친 정도에 비례하여 각 빈의 셀 밀집도를 계산한다.

셀 밀집도가 1을 초과하는 경우, 광역배치를 상세배치로 변환할 때 셀들의 위치가 다시 조정되어야 하며, 셀의 중첩 정도가 클수록 상세배치로 전환 후 셀들의 위치가 크게 달라지기 때문에 광역배치 단계에서 조정한 셀의 위치가 무의미해 진다.

제안하는 기법은 주어진 배치와 배선자원용량 정보를 바탕으로 배선요구량을 분석한 후, 각 빈의 셀 밀집도와 배선요구량의 변화를 고려하면서 셀을 효과적으로 옮겨 궁극적으로 배선밀집도가 높은 지역의 배선밀집도를 낮추고, 실행시 옵션의 선택에 따라 배선길이도 개선시킬 수 있다.

배선밀집도를 개선하는 기법을 설명하기 전에 중요한 함수의 기능을 먼저 설명한다.

4.1 배치의 평가

본 절에서는 어떤 배치 P가 얼마나 좋은지, 또 셀들이 이동한 후 배치가 수정되었을 때 이것이 개선되었는지 또는 더 나빠졌는지를 평가하기 위한 평가함수를 설명한다.

가장 쉽게 평가할 수 있는 방법은 HP를 이용한 모든 넷의 길이를 합한 것, 즉 $len(P)$이다. 일반적으로 배선길이를 줄이는 것은 전반적으로 배선을 용이하게 할 뿐 아니라 타이밍에도 긍정적인 영향을 미치기 때문에 배선길이를 줄이고자 하는 목표는 대부분의 배치기가 추구하는 것이다.

본 연구에서는 배선밀집도가 높은 곳을 찾아 배선밀집도를 낮추는 것이 주요 목표이기 때문에 배선밀집도와 관련되어 배치를 평가하는 방법이 필요하다. 가장 간단한 방법은 배선밀집도가 1을 초과하는 빈의 개수를 이용하는 방법도 있으나, 이는 배선밀집도가 1보다 아주 큰 경우와 겨우 1을 넘긴 경우를 구별하지 않기 때문에 문제가 있다. 이런 문제를 해결하기 위해 본 연구에서는 배선밀집도와 관련되어 배치를 평가할 때 다음과 같이 하였다. $cong()$는 배치 P에 대한

Function *cong*

```
Input: P, th // th: threshold value of congestion
Output: congestion_value
        S  // Set of indices of congested bins
S = Φ // empty set
congestion_value = 0;
for(i=1; i<=n; i++) {
   for(j=1; j<=m; j++) {
      δh = rch(i,j) - th;
      δv = rcv(i,j) - th;
      // α a constant greater than 1
      if (δh > 0) {
            congestion_value += (1+δh)^α;
            S = S∪(i,j)
      }
      if (δv > 0) {
            congestion_value += (1+δv)^α;
            S = S∪(i,j)
      }
   }
}
return S, congestion_value
```

배선밀집도를 평가한 값을 나타낸 것으로써 $len(P)$와 마찬가지로 그 값이 적을수록 좋은 배치라고 볼 수 있다.

위 함수에서 보인 바와 같이 배치 P의 배선밀집도를 평가할 때 각 빈에서 각 방향에 대해 배선밀집도가 임계값(threshold value)을 초과하는 경우에 대해서만 고려하였다. 임계값은 외부에서 입력되는 값으로써 주로 0.7부터 1사이에 있다. 이렇게 한 이유는 배선밀집도가 특정 값 이하인 빈에 대해선 배선에 문제가 없다고 보아서 이를 무시하기 위함이다. congestion_value에 $(1+\delta_h)^{\alpha}(\alpha>1)$을 더한 이유는 배선밀집도가 특정 값보다 클수록 더 많은 가중치를 두어 심각성을 더하게 하기 위함이다.

두 배치를 평가할 때 두 가지 측정값 $len(P)$와 $cong()$의 반환값인 congestion_value를 이용하여 다음과 같이 비교한다. w는 외부에서 제공되는 제어변수로서 이 값이 1이면 배치의 평가는 배선밀집도에만 가중치를 두고, 배선길이는 무시하게 된다. 반대로 w가 0이면 배치의 평가는 배선길이에만 초점을 맞추게 된다. 함수 $IsImproved(P_1, P_2, th, w)$는 배치 P_1에 대해 배치 P_2가 얼마나 개선되었는지를 숫자로 나타내는 함수로서 배치 P_1에 대해 배치 P_2가 개선되었다면 양의 값을, 그렇지 않으면 음수를 반환한다. 반환되는 양의 값이 클수록 P_2가 P_1에 비해 더 많이 개선되었다고 볼 수 있다.

Function *IsImproved*

Input: P_1, P_2, th, w // $0.7 \le th \le 1$, $0 \le w \le 1$
Output: gain

$$gainLen = \frac{len(P_1) - len(P_2)}{len(P_1)}$$

// Improved rate on wire length
// Among $cong()$'s return values, only 'congestion_value' is used here

$$gainCong = \frac{cong(P_1, th) - cong(P_2, th)}{cong(P_1, th)}$$

// Improved rate on congestion

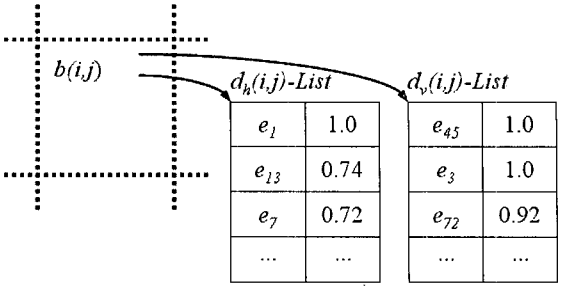
$$gainCombined = w \times gainCong + (1-w) \times gainLen$$

return *gainCombined*

4.2 제안하는 기법을 위한 자료구조

각 넷을 중심으로 배선요구량을 계산하는 과정에서 중요한 정보가 기억되는데, 본 절에서는 그 정보를 저장하는 자료구조와 관련된 사항을 설명한다.

3.2절에서 설명하였듯이, 각 빈에서 수직/수평 방향의 배선요구량을 계산하는데, 이 과정에서 어느 넷에 의해 어느 정도의 배선요구량이 필요한지 그 정보를 각 빈에 리스트를 이용해 저장해 둔다. 예를 들어 (그림 5)의 경우 $b(i,j)$에서 넷 e_1에 의한 수평 배선요구량이 1임을, 넷 e_{45}에 의한 수직 배선요구량이 1임을 각각 나타낸다. 이 배선요구량 리스트는 모든 빈에 대해, 각 방향에 대해 저장되며, 배선요구량 값이 감소하는 순으로 정렬되어 저장된다. 또한 리스트의 내용은 셀이 이동되어 배치가 변형되는 동안에도 동적으로



(그림 5) 수직/수평 성분의 배선요구량 리스트

rc	row	col	rc	row	col	rc	row	col	rc	row	col	rc	row	col
1.56	12	4	1.54	12	3	1.41	12	7	1.33	5	16	1.32	13	5

(그림 6) 빈의 배선밀집도 $rc(i,j)$에 따라 정렬된 리스트

바뀌면서 유지된다. 이 리스트가 너무 커지는 것을 방지하기 위해 배선요구량 값이 미리 정한 한계 값(예를 들어 0.9)을 넘어 가는 경우에만 저장된다.

이렇게 한 이유는 어떤 빈의 배선밀집도가 높아 이를 낮추길 원할 때 어느 넷이 그 빈의 배선밀집도에 가장 큰 영향을 미치는지 빨리 파악할 수 있고, 이 리스트를 동적으로 관리함으로써 셀이 이동되어 배치가 바뀔 때, 리스트의 변화가 필요한 빈에서 효과적으로 그 값을 수정해 줄 수 있기 때문이다.

또 다른 중요한 자료구조 중 하나는 2장 마지막 부분에 설명한 빈의 배선밀집도 $rc(i,j)$의 값에 따라 내림차순으로 구성된 정렬된 빈 리스트이다. (그림 6)에서 보인 것처럼 이 리스트의 각 항목에는 빈의 배선밀집도 $rc(i,j)$의 값과 빈 인덱스가 저장되어 있다. 이렇게 정렬된 리스트를 만드는 이유는 배선밀집도가 가장 큰 빈부터 시작하여 배선밀집도를 낮추도록 처리하기 위함이다. 우리는 이 리스트를 배선밀집도 리스트라고 부르기로 한다.

배선밀집도 리스트가 일단 구성되면, 리스트에 가장 먼저 있는 빈에 대해 배선밀집도를 낮추도록 셀들을 옮기는데 어떤 셀을 어떻게 옮길 것인가는 4.3절에서 설명한다. 한번 고려된 빈은 리스트에서 삭제된다. 또한 리스트에서 한번 삭제된 빈에 대해선 리스트에 남아 있는 모든 빈을 고려할 때까지 다시 고려하지 않는다. 이렇게 하는 주된 이유는 배선밀집도의 진동을 막기 위함이다. 배선밀집도의 진동이란 어떤 특정지역 (A 지역이라고 하자)의 배선밀집도가 높아 이를 해결하기 위해 셀들을 이동하면 다른 지역 (B 지역이라고 하자)의 배선밀집도가 증가할 수 있고, 다음엔 B지역의 배선밀집도를 해결하기 위해 셀들을 이동하다 보면 다시 A 지역의 배선밀집도가 증가하는 현상이 반복되는 것을 말한다.

리스트에 있는 모든 빈에 대한 고려가 끝나면 4.4절에서 설명할 배선밀집도 개선 기법의 제어 변수에 따라 배선밀집도 리스트가 새로 구성된다.

4.3 셀 이동에 따른 이득 계산

배선밀집도 개선을 위해 우리는 셀을 이동하는데, 셀이 현재 $b(i,j)$에 있다면 이웃한 8개의 빈 $b(i-1,j)$, $b(i-1,j+1)$,

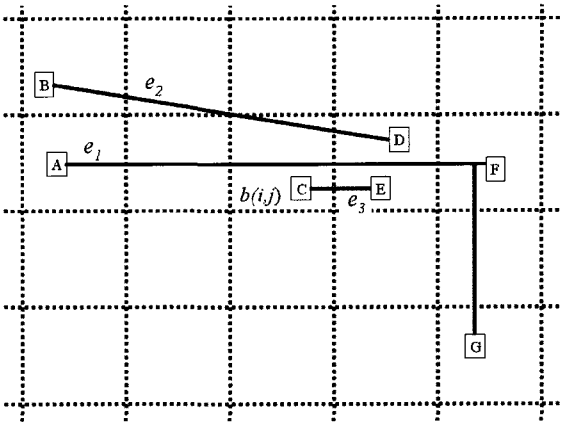
$b(i,j+1)$, $b(i+1,j+1)$, $b(i+1,j)$, $b(i+1,j-1)$, $b(i,j-1)$, $b(i-1,j-1)$ 가운데 하나로 이동한다. 이렇게 이웃한 빈으로만 이동을 제한하는 이유는 다음과 같다. 우리는 입력 배치가 다른 최적화 툴에 의해 최적조건(예를 들어 타이밍, 배선길이 등)이 어느 정도 수렴되었다고 가정하기 때문이다. 배치가 수렴되었다는 것은 더 이상의 개선의 여지가 없거나 개선되어도 그 정도가 매우 미미하다는 것을 의미한다. 본 연구의 주된 목적은 이미 수렴한 배치를 배선밀집도 관점에서 개선하되 이미 수렴된 조건을 희생하지 않거나, 희생하더라도 그 정도를 매우 적게 하면서 개선하는 것이다. 셀을 한 번에 너무 멀리 옮기는 것은 기존의 수렴된 조건을 심각하게 훼손할 우려가 있기 때문에 이웃한 빈으로만 이동을 제한한다.

셀을 이동하면 그 셀에 관계된 넷들의 길이와 토포로지가 변하고, 따라서 변한 넷들이 통과할 가능성이 있는 빈들에 대해 배선요구량이 변하게 된다. 이런 점들을 고려하여 셀 이동 시 발생하는 이득을 계산하는 함수 *ComputeGain*을 다음에 보였다. 이 함수의 기본 동작원리를 쉽게 보이기 위해 *IsImproved*함수를 내부에서 호출하였으나 실제로는 그렇지 않다. 그 이유는, 한 셀이 이웃한 빈으로 이동할 때, 그 셀에 연결된 넷에 따라 다르기는 하지만 일반적으로 한 셀에 연결된 넷이 많지 않기 때문에 배치의 변화는 매우 국부적이다. 이런 점을 고려하여 이득을 계산할 때 변화되는 정보만 추적하여 매우 효과적으로 이득을 계산한다.

```
Function ComputeGain
Input: C // cell to move
       r,c // bin index where C is
       t_r, t_c // target bin's index where C
                   moves to th, w
Output: gain
P  ← Current placement
P' ← New placement after moving cell C from
     b(r,c) to b(t_r, t_c)
gain = IsImproved(P, P', th, w)
Move C back to original bin
return gain
```

4.4 이동할 셀 선택

빈 $b(i,j)$의 배선 밀집도 $rc(i,j)$가 주어진 임계값보다 같거나 크다고 하자. 본 절에서는 $b(i,j)$의 배선밀집도를 낮추기 위해 어떻게 셀을 선택하는지를 설명한다. 설명의 편의를 위해 $rc_h(i,j) > rc_v(i,j)$라고 가정하자. 즉, 빈 $b(i,j)$의 수평 배선밀집도가 수직 배선밀집도보다 크며, 또한 $rc(i,j)$가 주어진 임계값 th보다 같거나 크기 때문에 $rc_h(i,j) \geq th$이 된다. $b(i,j)$의 수평 배선밀집도를 낮추기 위해 우선 어떤 넷이 수평 배선밀집도에 크게 영향을 미치는지 알아야 하는데 이는 4.2절((그림 5) 참조)에서 설명한 $d_h(i,j)-List$(수평 배선요구량 리스트)를 이용하면 쉽게 알 수 있다.



(그림 7) $b(i,j)$의 수평 배선밀집도에 영향을 미치는 넷을 보인 예

(그림 7)에서 예를 보인 것과 같이 $d_h(i,j)-List$에 있는 넷이 각각 $e_1=\{A,F,G\}$, $e_2=\{B,D\}$, $e_3=\{C,E\}$이고, 각 셀의 위치가 그림에서 보인 것과 같다고 가정하자. 셀 A, B, C, D, E, 또는 F를 위 혹은 아래 방향으로 옮기면 수평방향의 배선밀집도가 개선되는 것을 알 수 있다. 하지만 셀 G를 옮기는 것은 $b(i,j)$에서 수평 방향의 배선밀집도 개선에는 크게 도움이 되어 보이지 않는다.

또한 셀 G를 제외한 나머지 셀들에 대해 좌 혹은 우로 옮기는 것도 생각해 볼 수 있다. 예를 들어 셀 B를 우측 빈 $b(i-1,j-1)$으로 옮기면 넷 e_2에 의해 $b(i,j)$에서 야기되는 수평방향 배선요구량은 0.75에서 0.66으로 줄어들 뿐만 아니라 넷 e_2의 길이가 줄어드는 효과가 있다. 넷 길이를 줄이는 것은 배선밀집도 개선에 잠정적으로 도움이 되기 때문에 우리는 이런 움직임도 고려해 본다. 셀 D를 왼쪽 빈으로 옮기는 것도 같은 논리에 의해 고려해 볼 수 있다. 나아가 셀 B 또는 D를 앞에서 설명한대로 움직이면 $b(i-1,j)$에서도 수평방향의 배선밀집도 개선을 가져온다.

실제 배치는 (그림 7)에서 보인 것처럼 단순하지 않다. 셀 B가 다른 넷에 연결되어 있을 수 있고 B를 우측으로 옮기면 그 넷의 길이가 증가할 수 있으며, 나아가 그 넷으로 인해 다른 빈의 배선밀집도에 영향을 미칠 수 있기 때문이다. 이런 관찰을 바탕으로 $b(i,j)$의 수평 배선밀집도 개선을 위해 다음에 보일 함수 CellSelection을 이용하여 셀을 선택한다. 수직방향에 대해선 동일한 방법으로 처리될 수 있어 설명을 생략한다.

또한 셀을 움직일 때 목적지 빈의 셀 밀집도를 고려한다. 만약 빈 $b(i-1,j+1)$에 이미 여러 셀이 있어 셀 B가 이동되어 올 경우 셀 밀집도가 1을 초과한다면 우리는 셀 B를 $b(i-1,j+1)$로 옮기지 않는다. 어떤 빈의 셀 밀집도가 1을 초과한다는 말은 이는 그 빈 내에서 셀들이 중첩된 상황임을 의미하고 이를 해결하기 위해선 결국 어떤 셀을 다른 빈으로 옮겨야 된다. 그러면 배선길이나 배선밀집도가 변경될 수밖에 없고, 이 변경은 대부분 나쁜 방향으로 되어지기 때문에 이런 상황을 미리 차단하기 위함이다.

지금까지 설명한 것을 아래 *CellSelection*함수에서 의사 코드로 보였다.

```
Function CellSelection
Input: i,j // bin index of a congested bin
Output: cell name and direction
// The bin is assumed being congested horizontally
max_gain = 0
  for every e ∈ d_k(i,j) − List {
   for every cell C ∈ e {
      r,c ← index of a bin which cell C belongs
            to
       // if r is far from row i, ignore this case
       if (r > i+1 || r < i-1) continue
       for every 8 neighbor bins of b(r,c) {
           // Each neighbor bin can be a candidate
              for target bin
           if(cell density of target bin after
           moving > 1) {
               continue
           }
           // Assume the target bin's index is
              (t_r, t_c)
           gain = ComputeGain(C,r,c,t_r,t_c)
           if (gain > max_gain) {
               max_gain = gain
               save all necessary information
               regarding max_gain
           }
       } // neighbor bins
   } // every cell C ∈ e
} // every e ∈ d_k(i,j) − List
return information of cell name, target bin
       regarding max_gain
```

4.4 배선밀집도를 개선을 위한 효과적인 기법

일반적인 최적화 문제를 풀기 위해서는 상태 공간 탐색을 효과적으로 하여야 하는데 이 중 널리 사용되는 기법이 국부 탐색이다. 국부탐색 기법은 다음과 같이 간단히 요약할 수 있다. 상태 공간에 있는 현재의 해 A에서 이웃한 해 A′를 방문하여 A′가 A보다 개선된 것이라면 A를 A′로 대치하고, 만약 A′가 A보다 나쁜 해이면 A′의 이웃해 A″를 방문해 본다. 이처럼 A에서 출발하여 이웃한 해를 k번 방문했는데도 개선된 해를 찾지 못한다면 탐색을 중단하고 A를 반환한다. 탐색 중 개선된 해가 발견되면 개선된 해가 A가 되어 위의 과정을 반복한다.

국부탐색기법과 지금까지 설명한 주요 함수들을 이용하여 본 절에서는 배치에서 배선밀집도를 개선하기 위한 기법을 제안한다. 제안하는 기법은 CDP(Congestion Driven Placer)라 부르며, 이의 동작은 다음과 같이 기술될 수 있다.

단계 4-25에 걸친 while 문은 국부 탐색 기법을 적용한 것이다. P는 현재까지 발견한 최적의 배치이다. 단계 6-17에서는 P의 배선밀집도를 평가하고 그 평가를 바탕으로 셀들을 이동하여 새로운 배치 P′를 얻는다. 단계 19에서 P와 P′를 비교하여 P를 P′로 대치할 것인지 새로운 배치를 계속 얻어갈 것인지 결정한다.

```
procedure CDP (Congestion Driven Placer)
Input: P: a detailed placement
       k: // max num of iteration with no
             improvement
       w: // weight for congestion gain
       th: // threshold value of congestion
Output: new placement P
1. counter = k
2. S = Φ // S: a set of congested bins
3. // Consider input placement the best one

4. while (counter > 0) {
5.    // Note: S will contain a set of
         congested bins
      //       'congestion_value' is not used
               here
6.    S, congestion_value ← cong(P, th)
7.    Sort S in a non-increasing order
      according to congestion value
8.    for each bin b ∈ S {
9.       // Assume b's index is (i,j) and it is
            congested horizontally
10.      mv_cell, mv_direction ← CellSelection
         (i,j)
11.      Move 'mv_cell' along 'mv_direction' to
         a target bin
12.      P' ← New placement after moving a cell
13.   } // for each bin b ∈ S

14.   for each bin b(i,j) {
15.       Move cells in b(i,j) within the bin
          to resolve overlap
16.   }

17.   P' ← New placement after finishing step
      14~16

18.   // Now compare the new placement with the
         so-far best placement
19.   if( IsImproved(P,P',th,w) > 0) {
20.       P = P'
21.       counter = k
22.   } else {
23.       counter = counter - 1;
24.   }
25.} // while (counter > 0)

26.return P
```

단계 6은 3절에서 설명한 방법을 이용하여 실행할 수 있다. 즉, 모든 넷 각각에 대해 우선 RMST를 구하고, RMST에 있는 에지에 각각에 대해 두 셀을 연결하는 경로가 지나갈 확률에 근거하여 배선요구량을 계산한다. 3.2절 마지막 부분에서 설명했듯이 한 넷에 의해 필요한 배선요구량을 계산한 후 특정 빈에서 배선요구량이 1을 초과하면 이를 1로 조정해 준다. 수직 혹은 수평 성분 중 어느 것이라도 배선밀집도가 1을 초과하면 우리는 그 빈의 배선밀집도가 1을 초과한다고 한다.

단계 14-16은 셀들의 중첩을 없애기 위한 단계이다. 단계 10에서 선택된 셀을 옮길 때 목적지 빈에 그 셀이 들어갈

공간은 있지만 셀 이동 후 기존에 그 빈에 있던 다른 셀들과 위치가 중복될 수 있다. 이런 문제를 단계 14-16에서 해결한다. 이 단계에서는 셀들이 원래 있는 그 빈 내에서 조금씩 움직이기 때문에 셀이 다른 빈으로 넘어가는 일은 발생하지 않는다. 이 단계에서 셀을 움직일 때에는 배선길이를 고려하여 가장 적절한 위치로 셀들을 옮긴다.

5. 실험 결과 및 분석

CDP는 C로 구현되었고, 실험은 IntelXeon 3.6G 8G/Linux 상에서 하였다. 실험에 사용된 회로는 반도체 제작회사로 잘 알려진 I사에서 최근에 개발된 마이크로프로세서로부터 얻은 회로 중 8개를 사용하였다. 각 회로의 특징은 <표 1>에서 보였다. 회로이름은 회로에 있는 셀의 수가 증가함에 따라 번호를 붙여 표시하였다. CDP 입력으로 주어진 상세배치는 I사 내부에서 사용되는 툴을 이용하여 타이밍과 배선길이가 이미 수렴된 것이다.

〈표 1〉 실험에 사용된 회로의 특성

Test circuits	Number of cells	Number of nets
Ckt1	8,214	9,338
Ckt2	15,334	16,295
Ckt3	16,579	20,859
Ckt4	17,200	19,865
Ckt5	18,732	21,431
Ckt6	19,179	24,057
Ckt7	19,575	19,867
Ckt8	21,800	24,538

제안한 기법의 성능을 보이기 위해, 입력 파라미터의 값을 다양하게 변화시키면서 실험을 수행하였다.

격자의 차수는 코어 영역의 크기와 형태에 따라 행과 열의 개수가 2의 지수가 되도록 자동적으로 결정된다. 각 빈의 높이는 대략 표준 셀 높이의 2~3배가 되며, 빈의 넓이는 격자의 차수와 행의 수에 따라 좌우되지만 대략 각 빈의 모양이 정사각형에 가깝도록 결정된다.

CDP의 입력 파라미터 중 k는 모든 회로에서 20으로 고정을 시켰다. 즉, 현재까지 찾은 가장 좋은 배치로부터 while 문을 20번 반복하면서 변형을 시도해서 개선되지 않으면 반복을 중단한다. *cong*()를 계산하기 위해 지수 값으로 사용되는 a는 3으로 고정시켰다. 이는 실험을 통해 적절한 값으로 평가되었기 때문이다.

<표 2>에서는 실험을 통해 가장 적절하다고 판단된 파라미터의 값을 이용하여 얻은 결과를 보여준다. 이 표에선 배선밀집도 이득 가중치 w는 0.5로, 배선밀집도 임계값 th는 0.85로 두었다. $w=0.5$로 둔다는 말은 배선길이 개선과 배선밀집도의 개선에 대해 동일한 가중치를 둔다는 것을 의미하며, $th=0.85$로 둔다는 말은 빈의 배선밀집도가 임계값 0.85이상이면 그 빈의 배선밀집도를 낮추기 위해 셀을 옮긴다는 것을 의미한다. 배선밀집도의 정도를 보이기 위해 *cong*()의 값을 직접 보이지 않았는데 그 이유는 이 값은 표준화된 값이 아니고 본 연구에서 배선밀집도 평가하기 정의한 값이기 때문이다. 대신 배선밀집도가 특정 값 사이에 있는 빈의 개수가 어떻게 변화되는지를 보였다. 예를 들어, 표에서 빈의 개수가 0.7인 열에 있는 것은 빈의 배선밀집도가 0.7이상 0.8미만인 것의 개수를 보여준다.

<표 2>에서 보면 CDP를 적용하기 전 Ckt1의 경우 배선밀집도가 0.9이상인 빈의 총수가 213개이었으나 CDP를 적

〈표 2〉 k=20, w=0.5, th=0.85로 두었을 때의 결과(#bins: 빈의 배선밀집도가 주어진 값보다 큰 빈의 수)

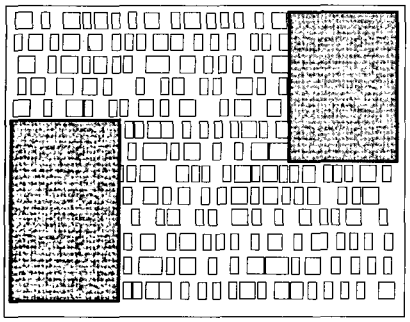
Test Ckts		wire length (meter)	number of bins							#iterations of while loop	CPU time
			0.7	0.8	0.9	1.0	1.1	1.2	1.3		
Ckt1	**Before CDP**	**0.36538**	**212**	**152**	**91**	**64**	**28**	**22**	**8**		
	After CDP	0.36300	448	190	15	3	0	0	0	77	2m15s
Ckt2	**Before CDP**	**0.78416**	**261**	**133**	**56**	**29**	**19**	**8**	**24**		
	After CDP	0.77064	409	151	19	7	2	0	1	120	4m13s
Ckt3	**Before CDP**	**1.1567**	**548**	**345**	**184**	**88**	**28**	**8**	**7**		
	After CDP	1.1467	938	350	19	4	0	0	0	54	9m9s
Ckt4	**Before CDP**	**1.0699**	**42**	**19**	**13**	**0**	**0**	**0**	**0**		
	After CDP	1.0699	57	18	0	0	0	0	0	60	32s
Ckt5	**Before CDP**	**1.0417**	**47**	**23**	**15**	**9**	**8**	**2**	**1**		
	After CDP	1.0342	73	39	3	4	0	0	0	75	82s
Ckt6	**Before CDP**	**1.2648**	**191**	**85**	**31**	**9**	**2**	**0**	**0**		
	After CDP	1.2595	233	77	0	0	0	0	0	49	2m34s
Ckt7	**Before CDP**	**1.0966**	**312**	**260**	**182**	**86**	**32**	**30**	**101**		
	After CDP	1.0792	609	358	88	45	10	4	15	274	33m15s
Ckt8	**Before CDP**	**1.1195**	**479**	**284**	**161**	**90**	**48**	**31**	**20**		
	After CDP	1.0973	842	376	7	0	0	0	0	182	11m38s

〈표 3〉 k=20, w=0.5, th=0.7~1.0으로 변화시켰을 때의 결과(#bins: 빈의 배선밀집도가 주어진 값보다 큰 빈의 수)

Test Ckts			wire length (meter)	number of bins							CPU time
				0.7	0.8	0.9	1.0	1.1	1.2	1.3	
Ckt1	Before CDP		**0.36538**	**212**	**152**	**91**	**64**	**28**	**22**	**8**	
	After CDP	th=0.7	0.35018	380	37	6	0	0	0	0	35m11s
		th=0.8	0.35994	541	62	9	0	0	0	0	4m3s
		th=0.9	0.36387	359	274	25	6	0	0	0	62s
		th=1.0	0.36487	268	229	129	8	5	2	0	28s
Ckt2	Before CDP		**0.78416**	**261**	**133**	**56**	**29**	**19**	**8**	**24**	
	After CDP	th=0.7	0.73087	267	52	9	7	0	0	0	25m56s
		th=0.8	0.76446	472	58	18	6	1	0	0	4m50s
		th=0.9	0.77358	367	198	23	6	1	1	0	132s
		th=1.0	0.77688	294	178	80	10	2	4	1	76s
Ckt3	Before CDP		**1.1567**	**548**	**345**	**184**	**88**	**28**	**8**	**7**	
	After CDP	th=0.7	1.0650	520	30	1	0	0	0	0	3h19m27s
		th=0.8	1.1374	1154	98	9	2	0	0	0	13m46s
		th=0.9	1.1510	750	524	28	7	2	0	0	5m46s
		th=1.0	1.1551	596	438	217	11	6	0	0	2m4s
Ckt4	Before CDP		**1.0699**	**42**	**19**	**13**	**0**	**0**	**0**	**0**	
	After CDP	th=0.7	1.0651	42	0	0	0	0	0	0	2m55s
		th=0.8	1.0693	72	1	0	0	0	0	0	36s
		th=0.9	1.0705	44	27	1	0	0	0	0	14s
		th=1.0	1.0703	40	19	13	0	0	0	0	10s
Ckt5	Before CDP		**1.0417**	**47**	**23**	**15**	**9**	**8**	**2**	**1**	
	After CDP	th=0.7	1.0150	49	13	1	1	0	0	0	6m58s
		th=0.8	1.0306	95	18	6	1	0	0	0	1m48s
		th=0.9	1.0370	58	41	8	5	0	0	0	70s
		th=1.0	1.0373	54	33	19	5	0	0	0	60s
Ckt6	Before CDP		**1.2648**	**191**	**85**	**31**	**9**	**2**	**0**	**0**	
	After CDP	th=0.7	1.2100	92	5	0	0	0	0	0	1h10m59s
		th=0.8	1.2540	93	13	0	0	0	0	0	6m7s
		th=0.9	1.2619	218	91	4	0	0	0	0	1m44s
		th=1.0	1.2636	204	87	30	2	0	0	0	25s
Ckt7	Before CDP		**1.0966**	**312**	**260**	**182**	**86**	**32**	**30**	**101**	
	After CDP	th=0.7	1.0173	450	183	51	11	2	3	10	1h55m17s
		th=0.8	1.0657	770	200	70	20	7	2	14	42m39s
		th=0.9	1.0830	509	449	99	40	13	4	16	29m47s
		th=1.0	1.0895	379	325	270	67	26	9	21	11m48s
Ckt8	Before CDP		**1.1195**	**479**	**284**	**161**	**90**	**48**	**31**	**20**	
	After CDP	th=0.7	1.0491	639	160	3	0	0	0	0	1h54m13s
		th=0.8	1.0866	926	179	10	0	0	0	0	18m29s
		th=0.9	1.1042	667	499	50	3	0	0	0	7m23s
		th=1.0	1.1098	476	372	287	21	1	0	0	2m45s

〈표 4〉 k=20, th=0.8, w=0~1.0으로 변화시켰을 때의 결과(#bins: 빈의 배선밀집도가 주어진 값보다 큰 빈의 수)

Test Ckts			wire length (meter)	number of bins							CPU time
				0.7	0.8	0.9	1.0	1.1	1.2	1.3	
Ckt1	Before CDP		**0.36538**	**212**	**152**	**91**	**64**	**28**	**22**	**8**	
	After CDP	w=1.00	0.37344	633	71	5	1	0	0	0	4m49
		w=0.75	0.36394	587	63	6	1	0	0	0	2m27
		w=0.50	0.35994	541	62	9	0	0	0	0	4m3s
		w=0.25	0.35803	480	92	14	4	0	0	0	5m16s
		w=0.00	0.35654	286	103	54	48	17	5	3	16m29s
Ckt2	Before CDP		**0.78416**	**261**	**133**	**56**	**29**	**19**	**8**	**24**	
	After CDP	w=1.00	0.78315	588	49	12	2	0	0	0	4m49s
		w=0.75	0.77046	515	61	10	7	1	0	0	3m17s
		w=0.50	0.76446	472	58	18	6	1	0	0	4m50s
		w=0.25	0.75887	453	57	18	10	5	1	0	7m31s
		w=0.00	0.75156	317	77	31	18	10	3	11	34m25s
Ckt3	Before CDP		**1.1567**	**548**	**345**	**184**	**88**	**28**	**8**	**7**	
	After CDP	w=1.00	1.1620	1303	78	5	2	0	0	0	11m33s
		w=0.75	1.1468	1206	95	4	2	0	0	0	12m15s
		w=0.50	1.1374	1154	98	9	2	0	0	0	13m46s
		w=0.25	1.1286	1042	101	11	2	0	0	0	21m33s
		w=0.00	1.1095	620	194	72	26	11	2	0	1h32m39s
Ckt4	Before CDP		**1.0699**	**42**	**19**	**13**	**0**	**0**	**0**	**0**	
	After CDP	w=1.00	1.0716	76	1	0	0	0	0	0	17s
		w=0.75	1.0701	70	3	0	0	0	0	0	29s
		w=0.50	1.0693	72	1	0	0	0	0	0	36s
		w=0.25	1.0687	71	4	0	0	0	0	0	40s
		w=0.00	1.0685	55	12	2	0	0	0	0	60s
Ckt5	Before CDP		**1.0417**	**47**	**23**	**15**	**9**	**8**	**2**	**1**	
	After CDP	w=1.00	1.0443	114	17	4	0	0	0	0	63s
		w=0.75	1.0347	103	16	3	1	0	0	0	72s
		w=0.50	1.0306	95	18	6	1	0	0	0	1m48s
		w=0.25	1.0270	74	18	6	3	0	0	0	4m9s
		w=0.00	1.0266	52	16	12	8	1	1	0	7m4s
Ckt6	Before CDP		**1.2648**	**191**	**85**	**31**	**9**	**2**	**0**	**0**	
	After CDP	w=1.00	1.2662	324	19	0	0	0	0	0	2m43s
		w=0.75	1.2590	312	16	0	0	0	0	0	4m14s
		w=0.50	1.2540	293	13	0	0	0	0	0	6m7s
		w=0.25	1.2460	241	11	0	0	0	0	0	12m38s
		w=0.00	1.2422	204	26	4	4	0	0	0	34m33s
Ckt7	Before CDP		**1.0966**	**312**	**260**	**182**	**86**	**32**	**30**	**101**	
	After CDP	w=1.00	1.1314	1081	261	35	12	1	3	12	27m41s
		w=0.75	1.0864	880	227	62	14	6	2	13	27m9s
		w=0.50	1.0657	770	200	70	20	7	2	14	42m39s
		w=0.25	1.0561	667	184	101	39	11	2	15	37m31s
		w=0.00	1.0496	400	179	129	80	38	14	53	2h31m15s
Ckt8	Before CDP		**1.1195**	**479**	**284**	**161**	**90**	**48**	**31**	**20**	
	After CDP	w=1.00	1.1470	1225	172	0	0	0	0	0	14m59s
		w=0.75	1.0984	1069	150	0	0	0	0	0	14m13s
		w=0.50	1.0866	926	179	10	0	0	0	0	18m29s
		w=0.25	1.0819	822	210	56	2	0	0	0	18m0s
		w=0.00	1.0760	487	167	106	99	37	13	0	1h54m2s



(그림 7) 배선불가능 영역을 보인 예

용한 후 그 수는 18로 줄어들었음을 보여준다. 반면에 배선밀집도가 0.9미만인 빈의 수는 CDP 적용하기 전 364개에서 638개로 늘어났다. 이는 배선밀집도를 분산시키는 과정에서 일어나는 자연스러운 현상이며, 모든 회로에서 이런 현상을 관찰할 수 있다.

CDP의 while 루프를 반복하는 횟수를 보면 Ckt4처럼 처음부터 배선밀집도가 그리 높지 않은 것은 일찍 수렴하나 배선밀집도가 높은 회로는 100여회를 반복한 후 수렴하는 것을 볼 수 있다. k=20인 점을 감안하면 Ckt6 같은 경우 29번 반복한 후 이미 배치는 수렴된 것을 알 수 있다. 이런 결과를 얻는데 필요한 CPU 시간도 빠른 경우는 1분 정도, Ckt7을 제외하면 늦은 경우도 대략 10분 정도인 점을 감안하면 매우 효과적으로 배선밀집도를 해결하고 있음을 알 수 있다.

Ckt7의 경우 다른 회로와 달리 배선밀집도가 1.3이상인 빈이 CDP 적용 후에도 15개나 남아 있는데 이는 (그림 7)에서 보인 것처럼 배선자원용량이 0인 배선불가능한 영역이 코어 영역 모서리 부분에 두 군데나 있어 이를 사이에 둔 셀들을 연결하기 위해선 배선불가능 영역 주변을 지나갈 수 밖에 없는 상황이 되어 그 주변의 배선밀집도는 제안한 방법으로 해결이 불가능하여 나타난 현상임을 알았다. 이런 배선장애를 해결하기 위해 더 많은 CPU 시간을 소모한 것으로 판단된다.

<표 2>에서 배선길이의 변화를 보면 CDP를 적용한 후에 배선길이가 오히려 줄어든 것을 볼 수 있는데 이는 배선밀집도를 해결하면서 또한 배선길이를 같이 고려하도록 $w=0.5$로 두었기 때문이다. 이처럼 배선밀집도 개선과 더불어 배선길이 또한 개선할 수 있다는 점은 제안한 기법이 매우 우수함을 보여주는 예라고 할 수 있겠다.

<표 3>에서는 CDP의 입력 파라미터 중 th(배선밀집도 임계값)의 변화에 따라 결과가 어떻게 달라지는지를 보인다. th의 값이 적을수록 더 많은 빈에 대해 처리하고, 결과적으로 배선밀집도와 배선길이가 더 많이 개선된다. 동시에 수행시간이 더 많이 필요한데 이는 자연스러운 결과로 보인다. w를 일정하게 둔다면 th값을 이용하여 배치의 질과 수행시간 사이에서 선택을 해야 한다. 더 좋은 배치를 원한다면 더 많은 CPU시간을 소비해야 할 것이며, 이는 대부분의 최적화 문제에서 볼 수 있는 현상이다. $th=0.7$일 경우 배선밀집도 1 이상인 빈이 Ckt7을 제외한 대부분의 회로에서 거의 없는 것을 볼 수 있다. 이는 제안한 알고리즘이 배선밀집도를 얼마나 효과적으로 낮추는지를 보여주는 증거라 하겠다. Ckt7의 경우는 앞에서 설명하였듯이 배선불가영역으로 인해 배치의 개선에 한계가 있음을 보여주며, 이는 앞으로 더 연구해야 할 과제로 보인다.

<표 4>에서는 CDP의 입력 파라미터 중 th는 특정 값에 고정시키고 w(배선밀집도 밀집도 이득 가중치)의 변화에 따라 결과가 어떻게 달라지는지를 보인다. w의 값이 클수록 배선 길이의 이득보다는 배선밀집도 이득에 대해 더 큰 가중치를 두는 것을 의미한다.

<표 4>에서 보면 예상대로 w의 값이 클수록 배선밀집도가 더 많이 개선되나 상대적으로 배선길이는 더 나빠지는 것을 볼 수 있다. 특히 $w=0.0$일 때, 배선밀집도 이득을 무시하고 배선길이 이득만 고려하게 되는데 이 경우 상당한 CPU 시간을 소비했음에도 불구하고 배선길이가 크게 개선되지 않은 것을 볼 수 있다. 이는 근본적으로 CDP가 배선밀집도를 해결하는데 초점을 맞추어 설계된 알고리즘이기 때문이다. 유사한 논리로 $w=0.0$로 두었다 하더라도 움직일 셀들의 후보를 고려하는 과정에서 근본적으로 배선밀집도를 개선하는 것을 전제로 하기 때문에 아무리 배선밀집도 이득에 대한 가중치를 0으로 두어도 입력 배치와 비교하면 배선밀집도가 개선되는 것을 볼 수 있다.

<표 4>에서 보여주는 결과에 의하면 w는 배선밀집도와 배선길이를 고려하여 결정해야 한다. w를 크게 두면 배선길이는 입력보다 더 나빠지는데 이는 입력배치 자체가 이미 다른 조건(예를 들어 타이밍)을 만족시키고 있는 것이라면 용납하기 어려운 경우라고 볼 수 있겠다.

th값을 다른 값(예를 들어 0.8 등)으로 고정시켜도 결과의 형태는 거의 동일하게 나오는 것을 관찰하였다.

6. 결 론

본 논문에서는 상세배치의 배선밀집도를 예측하고 배선밀집도가 높은 곳을 효과적으로 완화시키는 새로운 기법을 제안한다. 배선밀집도 예측을 위해서는 MST 넷 모델을 사용하였는데 이는 계산 속도가 빠르면서 실제적인 라우터의 배선과 큰 차이가 없기 때문이다. 제안된 기법의 특징은 다음과 같이 요약될 수 있다.

첫째, 기존의 배선밀집도 기반 배치기들과는 달리 특정지역을 관통하는 넷이 많아 그 지역의 배선밀집도가 높아질 경우 관통하는 넷에 연결된 셀들을 효과적으로 찾아내어 그들의 위치를 조정함으로써 배선밀집도를 완화시킬 수 있다.

둘째, 배선밀집도에 기반하여 셀들을 이동한 후, Mongrel [16]에서 제안한 것처럼 배선길이와 타이밍에 기반한 리플이동 (ripple movement) 기법을 이용하여 셀들을 국부적으로 다시 이동시킨다. 이렇게 함으로써 변화된 배치에서 셀의 중첩을 최소화시키고 따라서 결과적으로 얻은 광역배치를 상세배치로 변환하였을 때 배선길이가 최소한으로 희생되는

효과가 있으며, 주어진 입력배치보다 타이밍도 더 나빠지지 않는 효과가 있다.

셋째, 셀의 이동에 따른 배선밀집도의 변화 및 배선길이의 변화를 효과적으로 추적할 수 있도록 증분 자료구조(incremental data structure)를 사용하였다. 제안된 증분 자료구조는 배선밀집도가 높은 영역에 대해 이를 해결하는데 초점을 맞추어 설계되었기 때문에 매우 효과적이며, 또한 실행시간을 감안하여 모든 넷에 대한 정보를 저장하는 대신 배선밀집도를 개선하는데 효과가 크다고 판단되는 넷들에 대해서만 정보를 저장함으로써 매우 빠르게 실행되는 장점이 있다.

넷째, 본 논문에서는 MST 넷 모델을 사용하여 배선밀집도를 예측하지만 제안한 기법은 특정 넷 모델에 상관없이 적용할 수 있다. 특히 실제 라우터로부터 배선정보를 얻을 수 있다면, 이를 근거로 배선밀집도를 해결할 수 있다.

제안된 기법을 이용한 실험결과는 배선길이를 희생하지 않으면서 배선밀집도를 개선할 수 있음을 보여준다. 뿐만 아니라 파라메터의 설정에 따라 배선밀집도 개선과 배선길이 개선의 정도를 조절할 수 있다.

이번 연구를 바탕으로 앞으로 ① 타이밍과 배선밀집도를 동시에 개선할 수 있는 기법 ② 배선불가능 영역이 있을 경우 이를 고려하여 셀을 효과적으로 이동할 수 있는 기법 등을 찾아야 할 것으로 생각된다.

참 고 문 헌

[1] A. E. Caldwell, A. B. Kahng, and Igor L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?," Proc. of DAC, pp.477-482, 2000.

[2] D. J. -H. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective," Proc. of ISPD, pp.18-25, 1997.

[3] M. C. Yildiz and P. H. Madden "Improved Cut Sequences for Partitioning Based Placement," Proc. of DAC, pp.776-729, 2001.

[4] X. Yang, M. Wang,? K. Egur and M. Sarrafzadeh, "A Snap-on Placement Tool," Proc. of ISPD, pp.153-158, 2000.

[5] Ke Zhong and S. Dutt, "Effective Partition-Driven Placement with Simultaneous Level Processing and a Global Net Views," Proc. of ICCAD, pp.254-259, 2000.

[6] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Optimal End-Case Partitioners and Placers for Standard-Cell Layout," Proc. of ISPD, pp.90-96, 1999.

[7] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," Bell Syst. Tech. J., Vol.49 No.2, pp.291-307, 1970.

[8] C. M. Fiduccia and R. M. Matteyses, "A Linear Time Heuristic for Improving Network Partitions," Proc. of DAC, pp.175-181, 1982.

[9] Y. G. Saab, "A Fast Clustering-Based Min-Cut Placement Algorithm with Simulated Annealing Performance," VLSI Design, Vol.5, No.1, pp.37-48, 1996.

[10] Wern-Jieh and Carl Sechen, "Efficient and Effective Placement for Very Large Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp.349-359, 1995.

[11] S. Goto, "An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout," IEEE Trans. Circuits and Systems, CAS-28, pp.12-18, 1981.

[12] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," Proc. of DAC, pp.269-274, 1998.

[13] H. Etawil, S. Arebi, and A. Vannelli, "Attractor-Repeller Approach for Global Placement," Proc. of ICCAD, pp.20-24, 1999.

[14] Maogang Wang, X. Yang, Ken Eguro, and M. Sarrafzadeh, "Dragon2000: Placement of Industrial Circuits," Proc. of ICCAD, pp 260-263, 2000.

[15] X. Yang, B-K. Choi, and M. Sarrafzadeh, "A Standard-Cell Placement Tool for Designs with High Row Utilization," International Conference on Computer Design, pp.45-49 2002.

[16] S. Hur and J. Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement," Proc. of ICCAD, pp.165-170, 2000.

[17] G. Meixner and U. Lauther, "Congestion-Driven Placement Using a New Multi-Partitioning Heuristic," Porc. of ICCAD, pp.332-335, 1990.

[18] Maogang Wang, Xiaojian Yang, Majid Sarrafzadeh, "Congestion Minimization During Placement," IEEE Trans. CAD of Integrated Circuits and Systems, Vol.19, No.10, pp.1140-1148, 2000.

[19] M. Wang and M. Sarrafzadeh, "Modeling and Minimization of Routing Congestion," Proc. of ASP-DAC, pp.185-190, 2000.

[20] M. Wang and M. Sarrafzadeh, "On the Behavior of Congestion Minimization During Placement," Proc. of International Symposium on Physical Design, pp.145-150, 1999.

[21] C.C.Chang, Jason Cong, Zhigang Pan, Zin Yuan, "Multilevel Global Placement With Congestion Control," IEEE Trans. CAD of Integrated Circuits and Systems, Vol.22, No.4, pp.395-409, 2003.

[22] Jason Cong and Majid Sarrafzadeh, "Incremental Physical Design," Proc. of ISPD, pp.84-92, 2000.

[23] Olivier Coudert, Jason Cong, Sharad Malik, and Majid Sarrafzadeh, "Incremental CAD," Proc. of ICCAD, pp.236-243, 2000.

[24] Ulrich Brenner & Andre Rohe, "An Effective Congestion-Driven Placement Framework," IEEE Trans. CAD of Integrated Circuits and Systems, Vol.22, No.4, pp.387-394, 2003.

[25] P. N. Parakh, R. B. Brown and Karem A. Sakallah,

"Congestion Driven Quadratic Placement," Proc. of DAC, pp.275-278, 1998.
[26] Wenting How, Hong Yu, Xianlong Hong, Yici Cai, Weimin Wu, Jun Gu, bunch, and William H.Kao, "A New Congestion Driven Placement Algorithm Based on Cell Inflation," Proc. of ASP-DAC, pp.605-608, 2001.
[27] Andrew B. Kahng and Xu Xu, "Accurate Pseudo-Constructive Wirelength and Congestion Estimation," ACM International Workshop on System-Level Interconnect Prediction, pp.61-68, 2003.
[28] Chris C. N. Chu, "FLUTE: Fast Lookup Table Based Wirelength Estimation Technique," Proc. ICCAD, pp.696-701, 2004.
[29] Chris Chu and Yiu-Chung Wong, "Fast and Accurate Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," Proc. ISPD, pp.28-35, 2005.
[30] M. Wang, X. Yang, K Eguro and M. Sarrafzadeh, "Multi-center Congestion Estimation and Minimization During Placement," Proc. ISPD, pp.147-152, 2000.
[31] Xiaojian Yang, Ryan Kastner, M.Sarrafzadeh, "Congestion Reduction During Placement Based on Integer Programming," Proc. ICCAD, pp.573-576, 2001.

오 은 경

e-mail : ekoh@donga.ac.kr
1997년 동아대학교 컴퓨터공학과(학사)
2002년 동아대학교 컴퓨터공학과 (공학석사)
2005년 동아대학교 컴퓨터공학과 박사과정 수료
관심분야 : CAD, 컴퓨터 알고리즘

허 성 우

e-mail : swhur@daunet.donga.ac.kr
1981년 경북대학교 전자공학과(학사)
1983년 한국과학기술원 전산학과 (공학석사)
2000년 UIC Dept. of EECS(공학박사)
1986년~현재 동아대학교 전기전자컴퓨터 공학부 교수
2001년~현재 미국 Intel사 Physical Design분야 기술자문위원
관심분야 : CAD, 알고리즘, 계산 기하학, Combinatorial optimization