포스 디렉티드 방법과 최적 인터리빙 기법을 이용한 타이밍 드리븐 배치

(Timing Driven Placement using Force Directed Method and Optimal Interleaving Technique)

성 영 태 † 허 성 우 ††
(Young-Tae Sung) (Sung-Woo Hur)

요 약 본 논문에서 제안하는 기법은 기존의 첨단 배치기인 Kraftwerk (& KraftwerkNC)와 Mongrel을 개선 확장한 것으로써, 광역배치에서 셀 중첩을 효과적으로 해결하는 Mongrel의 ripple move 기법과 force directed 광역배치기인 KraftwerkNC의 강력한 성능을 결합한 것이다. 제안한 기법에서는 Mongrel의 ripple move를 최적화하기 위해 Kraftwerk에서 사용한 힘 분산(force spreading)기법을 이용한다. 셀 밀집을 개선시키고, 배선길이를 최적화하는 과정에서 타이밍을 위해 넷 제약조건들이 고려된다. 제안된 기법을 통해 얻은 실험 결과는 배선길이 뿐만 아니라 타이밍에서 향상된 결과를 보여준다.

키워드 : 포스 디렉티드 기법, 최적 인터리빙 기법, 타이밍 드리븐 배치

Abstract The proposed method for a force directed global placement algorithm exploits and extends techniques from two leading placers, Kraftwerk (& KraftwerkNC) and Mongrel. It combines the strengths of KraftwerkNC, force directed global placer, and Mongrel's ripple move technique which resolves cell overlaps effectively. The proposed technique uses the force spreading technique used in Kraftwerk to optimize the ripple movement. While it is resolving the cell overlap and optimizing wire length physical net constraints are considered for timing. The experimental results obtained by the proposed approach shows significant improvement on wire length as well as on timing.

Key words : Force-Directed Method, Optimal Interleaving technique, Timing-Driven placement

1. 서 론

자동화된 셀 배치는 VLSI 회로의 빠르고 효율적인 설계에 있어서 매우 중요한 단계이다. 셀 배치는 설계에서의 배선길이, 성능(performance) 그리고 배선가능성(routability) 같은 주요한 디자인변수에 큰 영향을 미친다. 반도체기술의 계속되는 발전과 함께 회로의 성능은 배선지연(wire delay)에 점점 더 의존적이게 되는데 이것은 배선지연이 게이트지연(gate delay)만큼 빠르게 감소되지 않기 때문이다[1]. 이런 이유에서 배선에 의한 지연을 최소화 할 수 있도록 하는 효율적인 타이밍드리븐 배치 알고리즘이 더욱 절실하게 필요하다. 왜냐하면 배치단계가 타이밍 조건을 만족시키는 설계를 위해 매우 중요한 역할을 하기 때문이다[2]. 또한, 집적율의 증가는 더 큰 회로의 설계를 가능하게 하고, 따라서 배치단계의 중요성은 더욱 커지게 된다[3].

셀 배치의 공통적인 목표는 셀들이 중첩되지 않도록 하면서 배선길이를 최소화하는 것이다. 배치를 위한 최근의 기술은 크게 세 가지로 분류될 수 있다. 첫 번째 부류는 처음부터 중첩을 허용하지 않으면서 배치 과정을 처리하는 것이다. TimberWolf[4]는 어닐링(annealing)을 기반으로 하는 잘 알려진 배치 툴이다. 이는 주어진 배치에서 셀들의 위치를 변형시킴으로써 새로운 배치를 얻어 가는데 이 기법을 사용하여 배치를 구한 결과들이 다수 발표되었다[5-8]. 두 번째 부류는 계층구조에서 분할기법을 적용함으로써 광역배치(global placement)로부터 상세배치(detailed placement)를 점진적으로 구해가는 방법을 사용하는 것이다. 분할기법의 핵심은 최소 컷(cut)을 재귀적으로 구하는 것인데, 컷이란 어떤 기준이 되는 직선(수직 또는 수평)에 대해 그 기준

· 본 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2003-002-D00303)
† 학생회원 : 동아대학교 컴퓨터공학과
saint@donga.ac.kr
†† 종신회원 : 동아대학교 컴퓨터공학과 교수
swhur@dau.ac.kr
논문접수 : 2005년 4월 28일
심사완료 : 2005년 11월 1일

선을 가로지르는 넷의 수를 말하고, 최소 컷을 구한다는 말은 컷이 최소가 되도록 셀들을 분할하는 것을 의미한다. 셀 분할 시 분할된 두 부분에 속한 셀의 면적의 합이 허용하는 범위 내에서 균형을 이루어야 한다. 이런 분할기법을 사용하는 툴로는 Capo[9], Quad[10], Feng-Shui[11], Snap-on[8] 등이 있다[12]. Caldwell[13]이 지적한대로 이 기법을 사용하는 툴들은 대부분 변형된 KL알고리즘[14]이나 FM알고리즘[15]을 이용하고 있다. 세 번째 부류는 해석적 기법 또는 FD(Force-Directed) 기법을 사용하는 것이다. FD 방법은 그 변화가 매우 다양하다. 셀의 위치를 구하기 위한 식을 어떻게 푸느냐에 따라 해의 질과 해를 구하는 시간이 달라진다. FD 방법은 근본적으로 중복을 허용하기 때문에 중복을 어떻게 해결하는가에 따라 해의 질이 크게 좌우된다. Goto는 iterative FD 방법을 이용한 독특한 알고리즘을 제안하였다[16]. Parakh[17]와 Yang[18]는 라우팅을 고려한 FD 배치 알고리즘을 제시하였으며, 셀 중복을 가능한 줄이기 위해 인장력 (attraction force)뿐만 아니라 척력 (repelling force)을 이용한 기법도 소개되었다[19,20]. Snap-on[8], Dragon[21,22]은 광역배치를 최적화시키기 위해 분할기법을, 상세배치 최적화를 위해 어닐링 기법을 사용하였다.

FD 기법을 사용하면 셀 중첩 문제를 해결하면서 동시에 배선길이를 최적화할 수 있기 때문에 최근에 많은 관심을 끌고 있다. 이 기법은 마치 여러 개의 용수철에 서로 연결된 다수의 물건들이 힘의 평형상태를 이루는 지점에서 그들의 위치가 정해지는 원리와 유사하다. 이런 기법을 사용하는 툴 가운데 하나인 Kraftwerk[19]는 광역배치를 최적화하기 위한 것이다. KraftwerkNC[23]는 Kraftwerk를 개선하여 타이밍 모델링을 좀 더 정확하게 할 수 있도록 하였다. KraftwerkNC는 넷의 제약조건을 이용하여 타이밍을 모델링 한다. 넷 제약조건이란 넷에 속한 모든 핀들을 둘러싼 가장 작은 사각형의 HP(half-perimeter)의 상한값을 나타낸다. 넷 제약조건들은 주어진 배치에서 타이밍-크리티컬 넷들의 타이밍을 분석함으로써 정해진다. KraftwerkNC는 매우 좋은 배선길이와 타이밍을 가지는 광역배치를 생성한다. 그러나 KraftwerkNC는 분산력을 위한 가중치의 증가량에 배치 결과가 크게 좌우되는 문제점이 있다. 또한, 코어 영역의 가장자리에 있는 셀들에게 작용하는 분산력의 모델링은 정확하지 않게 되고 이는 결국 가장자리에 있는 영역에선 셀들의 밀집현상이 발생하는 단점이 있다.

한편, Mongrel[24]은 하이브리드 기법을 이용하는 배치기로서, RBLS(Relaxation Based Local Search) 기법, 리플 이동(ripple movement) 기법, FM 분할기법[15], 그리고 최적인터리빙 기법 등을 통해 배선길이를 최적화 시킨다. 그러나 Mongrel은 타이밍을 고려하지 않고 셀을 이동하기 때문에 배선길이는 개선되나 타이밍이 나빠지는 문제가 있다.

본 논문에서는 KraftwerkNC가 가지는 세 가지 단점 -첫째, 배치 개선의 후반단계에서 셀의 중복을 해결하기 위한 제어가 까다로운 문제점, 둘째, 타이밍 조건을 위해 사용하는 넷 모델링의 부정확한 문제점, 셋째, 중첩을 해결하기 위한 정보가 코어 영역의 경계부분에서는 정확하지 않은 문제점을 보완하고, 또한 기존의 Mongrel이 가지는 단점 즉, 타이밍을 고려하지 않고 셀을 이동하는 문제점을 고려한 개선된 배치기법을 제안한다.

2장에서는 KraftwerkNC와 Mongrel의 동작원리에 대해 간략히 설명한다. 3장에서는 KraftwerkNC의 단점과 Mongrel의 단점을 개선하고, Mongrel의 장점을 융합한 매우 효과적인 배치기법인 개선된 Mongrel을 제안하며, 이의 동작원리에 대해 자세히 설명한다. 4장에서는 제안된 기법의 성능을 보여주는 실험결과를 보였으며, 5장에서는 결론을 맺는다.

2. 관련연구

2.1 Kraftwerk 및 KraftwerkNC의 개요

Kraftwerk는 각 셀에 연결된 넷 정보를 이용하여 셀의 위치를 결정하고, 셀 밀집도로부터 유도된 정보를 이용하여 광역 배치 중에 발생된 셀 중첩을 제거한다. Kraftwerk를 비롯하여 FD(Force-Directed) 기법을 사용하는 배치기의 가장 큰 장점은 배선길이, 셀 밀집도, 그리고 타이밍을 수식으로 모델링한 후 반복적으로 개선해 간다는 점이다. 이 기법은 이 세 가지 항목을 동시에 고려하면서 배치를 최적화 시킬 수 있도록 한다. 배선길이와 셀 밀집도는 힘(force)으로 모델링 되며 타이밍은 타이밍 크리티컬한 넷들에게 더 큰 가중치를 할당함으로써 모델링된다. 배선길이와 셀 밀집도를 모델링하는 힘은 최적 또는 준-최적의 밀집도를 가진 결과를 얻기 위해 아주 정밀하게 할당되어야한다. 셀 밀집도를 모델링한 분산력(spreading force)의 가중치는 배선길이를 위한 가중치에 비해 초기에 상대적으로 적은 값이 할당되어야 한다. 하지만 셀 분산을 위해 반복의 매 주기마다 그 가중치는 증가된다.

KraftwerkNC는 Kraftwerk를 개선하여 넷의 제약조건을 이용하여 타이밍을 좀 더 정확하게 모델링 할 수 있도록 하였다. 넷 제약조건이란 넷에 속한 모든 핀들을 둘러싼 가장 작은 사각형의 HP(half-perimeter)의 상한값을 나타낸다. 넷 제약조건들은 주어진 배치에서 타이밍-크리티컬 넷들의 타이밍을 분석함으로써 정해진다.

2.1.1 목적함수

Kraftwerk 및 KraftwerkNC는 회로를 그래프로 모

델링하며, 셀은 정점(vertex)으로, 넷은 간선(edge)들의 집합으로 나타낸다. k개의 셀들에 연결된 넷은 사이즈 k인 클릭(clique)으로 표현된다. 배치문제에서 넷 비용(cost)은 그 넷에 관계된 모든 간선들의 비용 합으로 표현되고, 한 간선의 비용은 그 간선에 있는 두 정점 사이 거리제곱으로 모델링되며, 목적함수(objective function)는 모든 넷의 비용 합으로 표시된다. 설계자에 의해 미리 위치가 고정된 패드(pad)를 제외한 셀 즉, 움직일 수 있는 셀이 n개라면 셀 $i(1 \le i \le n)$의 좌표는 셀의 중심의 좌표를 의미하며, (x_i, y_i)로 나타낼 수 있다. 두 셀 i와 j사이에 있는 간선의 x-성분에 대한 비용은 $(x_i - x_j)^2 = x_i^2 + 2x_i x_j + x_j^2$ 이고, y-성분에 대한 비용도 유사하게 표현된다. 모든 넷을 고려한 비용의 합 즉, 목적함수는 [19]에서처럼 다음과 같이 식 (1)로 나타낼 수 있다.

$$\Phi = \sum_{i,j} a_{ij}(x_i - x_j)^2 + \sum_{i,j} a_{ij}(y_i - y_j)^2 \quad (1)$$

여기서 $a_{ij} \in \{0,1\}$로서 셀 i와 셀 j사이에 간선이 존재하면 1이고 그렇지 않으면 0이 된다. 식 (1)은 행렬과 벡터의 곱으로 표현된 선형방정식으로 나타낼 수 있고, 이를 식 (2)에서 보였다[19,23].

$$\Phi = \frac{1}{2}\vec{p}^T C\vec{p} + \vec{d}^T + const \quad (2)$$

여기서 C는 2n X 2n인 Laplacian 행렬이고, $\vec{p} = (x_1,...,x_n,y_1,...,y_n)^T$, $\vec{d} = (d_1,...,d_{2n})^T$이다. 벡터 $\vec{d}$의 각 원소 d_i는 $1 \le i \le n$인 경우 셀 i에 직접 또는 간접적으로 연결된 패드의 x-좌표에 의해 결정된 상수이고, $n+1 \le i \le 2n$인 경우 패드의 y-좌표에 의해 결정된 상수이다. *const*는 패드의 위치로부터 결정된 상수이다.

목적함수 Φ를 최소로 만들기 위해 $\nabla\Phi = 0$를 풀면 되는데 이는 곧 식 (3)으로 표현된다[19,23].

$$C\vec{p} + \vec{d} = 0 \quad (3)$$

식 (3)을 풀어 벡터 $\vec{p}$의 각 원소의 값을 결정한다는 것은 Φ값을 최소로 하는 셀의 위치를 구하는 것을 의미한다. 식 (3)을 이용하여 셀의 위치를 구할 때 문제점은 많은 셀들이 중첩된다는 점이다. 이를 해결하기 위해 Kraftwerk에서는 중첩의 정도 즉, 배치 영역에서 셀 밀집도 분포로부터 유도된 벡터 $\vec{e}$를 추가한 식 (4)를 이용한다[19,23].

$$C\vec{p} + \vec{d}^T + \vec{e} = 0 \quad (4)$$

벡터 $\vec{e}$를 분산력 벡터라고 부르는데, 이는 높은 셀 밀집도를 가진 영역에서 낮은 셀 밀집도를 가진 영역으로 셀을 옮기는데 작용하고, 이를 이용함으로써 셀 중첩 문제를 완화시킬 수 있다.

2.1.2 알고리즘 KraftwerkNC

KraftwerkNC에선 2.1.1절에서 설명한 목적함수를 기반으로 셀의 위치를 정하되 타이밍 조건을 맞추기 위해 추가적인 넷 모델을 이용한다. 또한 셀 분산을 위한 반복이 진행되는 동안 크리티컬 넷들을 찾아 그 넷들에 적당한 가중치를 부여함으로써 타이밍을 최적화한다. KraftwerkNC의 동작과정은 다음 그림 1과 같다.

초기에 셀 분산을 위한 과정이 반복되는 동안, 분산력에 할당된 가중치는 배선길이와 비교해서 상대적으로 작다. 초기 반복이 진행되는 동안 계산된 분산력 값은 매우 유용한데 이는 중첩 제거를 위해 어떻게 셀이 이

```
algorithm KraftwerkNC
input: circuit information
output: placement
Transform input circuit to a graph
while (placement is not spread out) {
   for (fixed number of iterations) {
      for (all net constraints) {
         if (net not meeting constraints) {
            Distribute additional weight to bounding cells on this net
         }
      }
      Calculate spreading forces to form e vector
      Construct the C matrix and the d vector
      Solve the equation (4) for the new placement P
   }
   Run the timing analysis engine on P
   Use heuristics to constrain critical nets
}
return P
```

그림 1 알고리즘 KraftwerkNC

동되어야 하는지에 대한 거시적인 관점을 제공하기 때문이다. 그러나 반복이 진행될수록 분산력의 가중치는 증가되는데 이는 진동을 피하고 해에 수렴하기 위함이다.

크리티컬 넷에 있는 간선의 가중치만을 이용하여 타이밍 조건을 맞추기가 어렵기 때문에 이것을 해결하기 위해 마지막 단계에선 크리티컬 넷의 모든 간선을 중복하여 생성하는 휴리스틱을 이용한다.

2.2 Mongrel의 개요

Mongrel은 RBLS(Relaxation Based Local Search), 리플이동 기법, FM 분할기법[15], 그리고 최적인터리빙을 포함한 최적화 기법들의 모음이다. Mongrel은 광역배치와 상세배치를 각각의 단계에서 최적화시킨다. 광역배치 단계에서 코어영역(core region)은 $n \times m$의 격자(grid)로 분할된다. 격자 상에 분할된 $n \cdot m$개의 작은 직사각형 각각을 빈(bin)이라 부른다. 각 빈의 높이는 설계 시 정해진 행 높이와 같고, 넓이는 몇 개의 표준 셀을 담을 수 있도록 정해진다. Mongrel은 각 빈에 셀들을 할당하면서 시작한다. 그리고 회로에서 부분회로를 추출하여 그 부분회로에 있는 셀들의 최적위치를 네트워크 플로우(network-flow) 기법을 이용하여 결정하고, 결정된 빈에 셀을 넣는다. 이때 각 빈이 수용할 수 있는 셀의 한계를 고려하지 않는다. 그러므로 어떤 빈에는 셀들이 과다하게 집중되어 배치되는 다른 형태의 중첩이 발생한다. 이런 중첩문제는 리플이동에 기반한 방법을 이용하여 해결한다. 리플이동은 최소의 배선길이를 산출하는 단조경로(monotone path)에 따라 밀집도가 가장 높은 빈에 있는 셀들을 밀집도가 가장 낮은 빈으로 옮긴다. 이런 과정을 반복하여 광역배치를 최적화시킨 후 이를 상세배치로 변환한다. 상세배치 단계에선 최적인터리빙(optimal interleaving)을 사용하여 각 행에서 셀들의 선형배치를 향상시킨다. Mongrel의 강력한 성능은 셀 중첩 제거를 위한 효과적인 리플이동 기법과 각 행에서 진행되는 최적인터리빙, 그리고 넷 바운딩박스의 HP로 넷 길이를 정확하게 모델링하는데 있다. 알고리즘 RBLS는 배치의 거시적인 관점을 이용하여 광역배치를 개선시키는 최적화 기법이다. RBLS의 각 반복에서 적용되는 전통적인 FM 분할기법은 각 인접한 빈들 간에 지역적인 최적화를 위해 사용된다. Mongrel의 진행절차는 다음과 같이 요약될 수 있다.

첫째, 초기 광역배치를 얻는다.

둘째, RBLS 기법을 사용해서 현재의 광역배치를 최적화한다.

셋째, 현재의 광역배치를 상세배치로 변형한다.

넷째, 최적인터리빙 기법을 사용해서 상세배치를 최적화한다.

2.2.1 알고리즘 RBLS

알고리즘 RBLS에서는 배치의 거시적인 관점을 효율적으로 이용하기 위해 해석적(analytical) 방법을 사용한다. RBLS가 호출될 때 마다 새로운 광역 배치가 얻어진다. 알고리즘 RBLS를 그림 2에서 보였다.

입력으로 주어진 광역배치를 변형하여 목적함수가 개선되었다면 이를 현재의 배치로 간주하고, 개선되지 않으면 새로운 변형을 시도한다. 파라메터 k는 광역배치가 개선되지 않을 경우 얼마나 많이 배치의 변형을 시도할 것인지를 결정한다. 대부분의 최적화 문제가 그렇듯이 배치도 처음에는 많이 개선되지만 반복이 계속될수록 개선율이 떨어지고 궁극적으로는 더 이상 개선이 되지 않고 수렴하게 된다.

배치의 변형을 위해 단계 3에서 보듯이 넷에 기초하여 '유동(mobile) 셀'이라 불리는 여러 개의 셀들을 선택한다. 단계 4에서는 각 유동셀의 최적위치가 결정된다. 이 때 특정 빈에 많은 셀들이 과다하게 위치될 수

```
algorithm RBLS
input: m, k and current placement P
output: new placement
1. counter ← k
2. while (counter > 0) {
3.         Select a mobile cell set M (|M| = m)
4.         For each cell in M, determine optimal relaxed location
5.         P' ← New placement after ripple-movement
6.         P' ← Optimize P' with FM-partitioning technique
7.         if  (WL(P') < WL(P)) then
8.             counter ← k
9.             P ← P'
10.        else
11.            counter ← counter - 1
12. }
13. return P
```

그림 2 알고리즘 RBLS

있도록 허용한다. 즉, 광역 배치의 관점에서 셀 중첩을 허용한다. 단계 5에서는 변형된 광역배치에서 셀의 중첩 문제를 해결하기 위해 배선길이에 기반한 리플이동 기법을 이용하여 셀들을 이동시킨다. 즉, 셀들이 많이 밀집된 빈에서 여유가 있는 빈으로 물결치듯 이동하는 것이다. 리플이동이 진행되는 동안 셀들은 배선길이를 고려하여 최적의 셀이 선택되어 이동된다. 리플이동은 모든 빈들이 용량 제약조건을 만족할 때까지 반복된다.

단계 6에서는 변형된 광역배치에 FM 분할기법을 이용하여 인접한 빈들(수직과 수평방향으로 인접한 모든 빈들)간에 컷 사이즈를 최소화 시킨다. RBLS를 실행하는 동안에는 모든 셀들이 빈의 중앙에 있다고 가정하기 때문에 빈을 이용하여 배선길이를 개선하는 것과 빈 사이의 컷 사이즈를 줄이는 것은 궁극적으로 같은 목적함수이다. 단계 7에서는 지금까지 방법으로 구한 새로운 광역배치의 개선 여부를 판단한다. 만약 배치가 개선되었다면 새로운 배치가 '지금까지 가장 좋은 배치'로 채택된다. 개선되지 않았다면 새로운 배치는 버려지고 알고리즘은 단계 3으로 가서 반복을 계속한다.

어닐링 방법에서 사용되는 냉각 스케쥴링과 같이 RBLS를 호출할 때 전달되는 인수인 유동셀의 개수는 시간이 지남에 따라 감소되고 유동셀의 개수가 미리 결정된 한계보다 적게 될 때 RBLS는 더 이상 호출되지 않는다.

알고리즘 RBLS의 각 실행 단계 가운데 본 논문과 관련된 주요 단계는 리플이동 기법을 이용하여 셀을 이동하는 단계 5이고, 이는 다음과 같이 요약될 수 있다.

첫째, 현재의 광역배치에서 밀집도가 가장 높은 빈 S(Source bin)를 찾는다.

둘째, 현재의 광역배치에서 밀집도가 가장 낮은 빈 T(Target bin)를 찾는다.

셋째, 빈 S로부터 T까지 가능한 모든 단조경로(monotone path)를 고려하여 이득 그래프(gain-graph)를 만든다. 이득 그래프는 배선길이에 근거한 것으로써 순환이 없는 방향성 그래프이다.

넷째, 이득 그래프로부터 최대 이득 경로를 찾는다. 이때 각 빈에서 어느 셀을 이동하는 것이 최대 이득을 생성하는 지에 관한 정보도 저장한다.

다섯째, 최대 이득 경로를 따라 셀을 이웃 빈으로 이동한다.

여섯째, 모든 빈의 용량 제한조건을 만족할 때까지 첫째 과정부터 다섯 번째 과정을 반복한다.

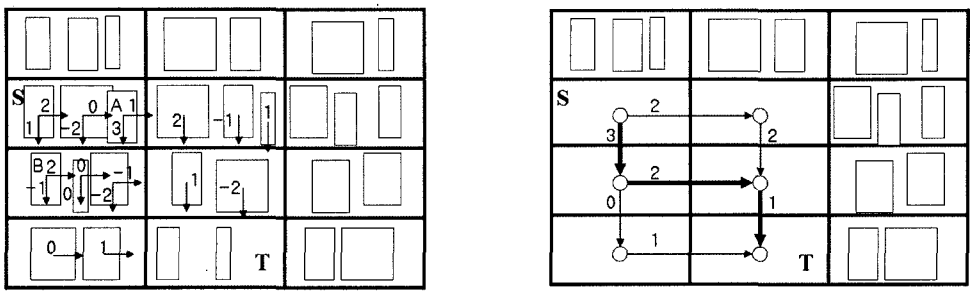
이득 그래프를 생성하기 위해 우리는 소스 빈 S로부터 타겟 빈 T까지의 모든 가능한 단조경로를 고려한다. S의 빈 인덱스가 (r_S, c_S)이고 T의 빈 인덱스가 (r_T, c_T)라면 S에서 T까지 가능한 모든 단조경로의 수는 $\binom{|r_T - r_S| + |c_T - c_S|}{|c_T - c_S|}$가 된다. 모든 단조경로 상의 빈에 있는 각 셀에 대해, 단조경로의 방향으로 셀을 이동할 경우 배선길이의 이득이 얼마나 되는지를 계산하고, 각 방향에 대해 이득이 가장 큰 셀과 그 때의 이득 값을 저장한다. 저장된 이득 값을 이용하여 전체적인 이득이 가장 큰 단조경로 한 개를 선택하고 선택된 단조경로를 따라 가며 셀을 이동한다. 방금 이동된 셀이 단조경로를 따라 계속 이동될지 아니면 다른 셀이 이동될지는 이득의 비교에 의해 결정된다. 이득 그래프 생성을 위한 알고리즘은 그림 3과 같다.

그림 4에서는 이득 그래프가 어떻게 생성되며 또 최대 이득 경로가 어떻게 결정되는지를 보여준다. 이득 값은 셀을 이웃한 셀의 중앙으로 옮길 경우 배선길이가 얼마나 줄어드는지에 대한 값이다. 참고로, 그림 4에서 보인 값들은 개념만 보이기 위한 것으로써 실제 값은 아니다.

그림 4에서 보였듯이, 최대 이득 경로를 따라 셀을 옮길 경우 적어도 6만큼의 이득은 보장받는다. 왜냐하면 우선 S로부터 아래방향으로 이득이 3인 셀 A를 옮긴다. 셀 A를 옮긴 다음 이를 계속 우측으로 옮길 경우 이득

```
algorithm construct-gain-graph(S, T)
input: source and destination bins S, T in the grid
output: gain graph G as a DAG (Directed Acyclic Graph)
1.  G ← Ø
2.  // bins in bounding box of S, T become nodes
3.  for (each bin B between S and T) {
4.      Add node B to the gain graph G
5.      for (each candidate cell C in bin B) {
6.          gain g(C, X/Y) ← wire-length reduction, when C moves towards T along X/Y
7.      }
8.      gain g(B, X/Y) ← MAX(g(C,X/Y) | all C in B)
9.      //bin B has at most 2 neighbor bins in X/Y direction towards T
10.     Add an edge to gain graph G as an out arc along X/Y with gain cost g(B, X/Y)
11. }
12. return G
```

그림 3 알고리즘 Construct-gain-graph



(a) 셀 밀집도에 따라 S와 T를 결정하고, S에서 T까지의 단조 경로 상에 있는 빈에 속한 각 셀에 대해, 경로 상의 각 방향에 대해 이득을 계산함

(b) 각 빈을 정점으로 한 이득 그래프를 생성한 후 최대 이득 경로를 결정함

그림 4 이득 그래프를 생성하는 과정과 최대 이득 그래프를 결정하는 과정을 보여주는 예

이 얼마나 되는지를 계산한다. 그 이득이 2보다 작다면 셀 B를 우측으로 옮긴다. 이런 식으로 최대 이득 경로를 따라 셀을 옮긴다.

2.2.2 최적인터리빙 기법

상세배치를 구하고 나면, Mongrel은 최적인터리빙 기법을 이용하여 행 내에서 셀들의 위치를 최적화시킨다. 이 기법에 적용되는 대략적인 과정은 다음과 같이 요약될 수 있으며 그림 5가 이를 개괄적으로 보여준다.

첫째, 주어진 윈도우 사이즈 W에서, W내의 한 행에 있는 셀들의 현재 선형배열로부터 부분배열 A를 찾는다. A에 있는 셀들의 상대적인 순서는 유지된다. W 내에 있는 셀 들 중 A에 속한 것들을 제외한 셀들을 B라 하자. 이때 부분 배열 B에 속한 셀들의 순서도 원래의 순서에서와 같이 셀들의 상대적인 순서는 유지한다.

둘째, 최적의 순서를 얻기 위해 A와 B를 서로 끼워 넣으면서 새로운 순서를 구한다.

셋째, 첫 행부터 마지막 행까지, 각 행에서는 왼쪽에서 오른쪽으로 윈도우를 옮기면서 첫째와 둘째 단계를 반복한다.

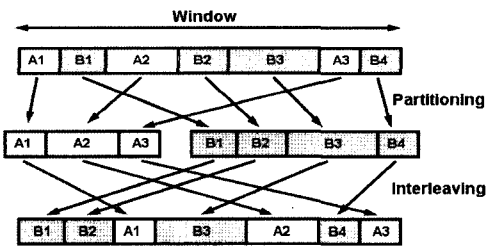


그림 5 최적인터리빙의 동작원리를 보이는 예

동적 프로그래밍 기법을 이용하면 두 부분 배열 A와 B를 효과적으로 인터리브시켜 최적의 순서를 구할 수 있음을 [24]가 보여준다.

3. 제안기법 - Force Directed Mongrel

3.1 제안기법의 동기

좋은 광역배치를 구하고자 하는 궁극적인 목표는 좋은 상세배치를 구할 수 있는 여건을 마련하는 것이라고 볼 수 있다. 좋은 상세배치는 좋은 광역배치로부터 얻기가 쉽기 때문이다. 좋은 상세배치란 크리티컬 넷에 기초한 넷 제약조건을 만족하면서 전체 배선길이를 최소화시킨 것이다.

KraftwerkNC는 배선길이와 타이밍 측면에서 좋은 광역배치를 생성한다. 그러나 몇 가지 문제점도 있는데, 첫 번째는 분산력을 위한 가중치의 증가량에 매우 민감하다는 점이다. 만약 분산력의 가중치가 빠르게 증가할 경우 배치는 매우 빨리 수렴하지만 배선길이와 타이밍의 관점에서 보면 좋은 배치가 생성되지 않는다. 이를 해결하기 위해 분산력의 가중치는 매번 반복할 때마다 천천히 증가된다. KraftwerkNC에서 초기의 셀 분산 이후, 분산력의 가중치가 천천히 증가되면 배치 전반에 걸쳐 국부적인 밀집이 존재함에도 불구하고 (분산력의 가중치와 배선길이를 위한 힘의 가중치 사이에 존재하는 평형에 의해) 셀 이동은 거의 일어나지 않게 된다. 이런 국부적인 밀집을 제거하기 위해 KraftwerkNC는 분산력의 가중치를 증가시키게 되는데 그 결과 배선길이, 타이밍 그리고 실행시간 면에서는 손실이 발생한다. 두 번째 문제점은 코어 영역의 가장자리에 있는 셀들에게 작용하는 분산력의 모델링이 정확하지 않다는 점이다. 이 문제로 인해 가장자리에 있는 영역에선 셀들의 밀집현상이 발생한다. 마지막으로, KraftwerkNC에서는 넷을 클릭으로 표현하는데 이는 넷에 연결된 모든 핀들의 쌍들에 대해 간선(edge)이 존재하게 된다. 따라서 넷 길이를 추정하기 위해 사용되는 HP로는 넷의 길이가 정확하게 모델링되지 않는다. 이런 약점은 KraftwerkNC가 실행을 반복할수록 더욱 더 나쁜 영향을 미치게 되는

원인이 된다.

Mongrel의 리플이동에 기반한 셀 이동 기법은 KraftwerkNC의 후반 단계에서 적용하는 미세한(fine-grained) 밀집 분산의 약점을 정확하게 보완할 수 있다. 즉, 리플이동 기법은 국부적인 밀집문제를 효과적으로 해결할 수 있다. 왜냐하면 Mongrel이 사용하는 넷 바운딩박스 HP 모델링은 KraftwerkNC에서 사용하는 클릭 모델에 비해 더욱 정확하기 때문이다.

제안하는 새로운 기법은 이런 KraftwerkNC의 약점을 Mongrel의 장점으로 보완한 것으로써, 이는 타이밍을 고려하면서 배선길이를 향상시킨다. 광역배치를 생성하기 위한 새로운 기법의 큰 흐름은 다음과 같다. 첫째, KraftwerkNC를 사용하여 단지 몇 개의 미세한 밀집을 가지는 광역배치를 생성한다. 둘째, Mongrel을 이용하여 배치의 변형(perturbation)을 최소로 하면서 배선길이와 타이밍을 최적화하기 위해 미세한 밀집 문제를 해결해 간다.

그러나 KraftwerkNC가 생성한 광역배치를 원래의 (수정되지 않은) Mongrel을 통해 밀집 문제를 해결할 경우 두 가지 문제가 발생된다. 첫째, KraftwerkNC가 생성한 배치를 원래의 Mongrel은 매우 크게 변형시킬 수 있다. 왜냐하면 KraftwerkNC와 Mongrel은 밀집문제를 해결하기 위해 서로 다른 방법을 사용하기 때문이다. 둘째, KraftwerkNC가 생성한 광역배치를 원래의 Mongrel을 사용하여 밀집 문제를 해결하면 타이밍이 더 나빠질 수 있다. 왜냐하면 Mongrel은 크리티컬 넷 제약조건을 고려하지 않기 때문이다.

이런 문제를 해결하기 위해 본 논문에서는 수정된 Mongrel을 제안한다. 즉, Kraftwerk에서 한 것처럼 셀 밀집도에 기반하여 분산력을 결정하고 이 분산력 정보를 이용해서 중첩을 제거할 수 있도록 한다. 이때 셀들이 너무 많이 이동하지 않도록 하기 위해, 즉 배치가 이전의 배치에 비해 크게 변형되지 않도록 이동거리를 제한한다. 또한 타이밍을 고려하여 이동될 셀을 결정함으로써 KraftwerkNC가 향상된 타이밍 결과를 보이듯이 제안한 기법에서도 넷 제약조건을 만족시키도록 한다. 물론 배선길이를 최적화시키는 본래의 목적은 살려둔다. 이렇게 수정된 Mongrel을 FD-Mongrel (Force-Directed Mongrel)이라 부른다.

3.2 FD-Mongrel의 동작 설명

FD-Mongrel에선 정밀도가 다른 두 종류의 격자 즉, 성근격자(coarse grid)와 미세격자(fine grid)를 사용한다. 성근격자 각 빈에는 여러 개의 작은 빈이 존재하는데, 이 작은 빈들은 미세격자 상에 있는 것이다. 그림 6에서 성근격자와 미세격자의 관계를 예로 보였다. 굵은 선은 성근격자를, 점선은 미세격자를 나타낸다.

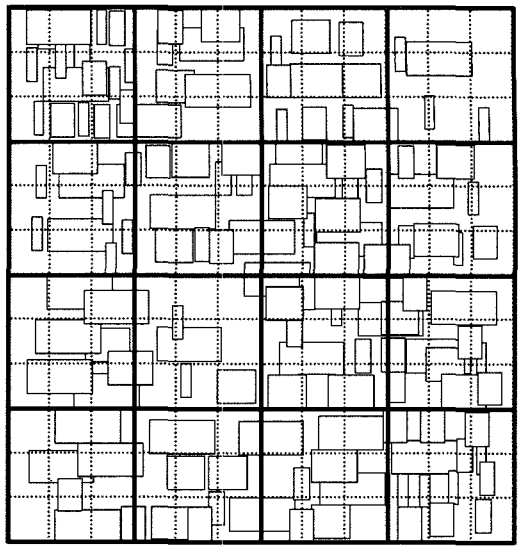


그림 6 성근격자와 미세격자의 관계를 보인 예

성근격자는 분산력을 구하기 위해 사용된다. 성근격자상의 각 빈에서 셀 밀집도를 바탕으로 분산력의 방향을 결정하고, 이 방향에 근거하여 셀들을 이동하게 된다. 성근격자의 각 빈 중심에서 셀 밀집도에 근거한 분산력 벡터를 구한다. 이런 이유로 우리는 성근격자를 '힘(force)-격자'로 부르기로 한다.

미세격자의 목적은 원래의 Mongrel에서와 유사하게 셀들을 리플이동하기 위한 경로를 결정하는데 사용된다. 미세 빈을 따라 셀을 이동하는 이유는 FD-Mongrel 수행이 끝난 후 광역배치를 상세배치로 변환하는 적법화(legalization) 과정을 효과적으로 할 수 있기 때문이다.

3.2.1 프로시저 move-cells

FD-Mongrel의 동작원리를 설명하기 이전에 셀들이 구체적으로 어떻게 이동되는지를 보이는 프로시저 move-cells에 대해 먼저 설명한다. 이는 FD-Mongrel에서 호출되는 핵심적인 프로시저이며, 그림 7에서 동작 개요를 보였다.

프로시저 move-cells에 전달되는 파라메터로는 소스(source) 빈과 타겟(target) 빈을 나타내는 힘-격자 상의 두 빈 S_{force}와 T_{force}, 그리고 셀 이동을 제어하기 위해 사용되는 임계값 D_{th}이다. 소스 빈에서는 셀이 이동하여 나가고, 타겟 빈으로는 셀이 들어온다. S_{force}와 T_{force}는 힘-격자 상에서 같은 빈일 수도 있고, 서로 다른 빈이 될 수도 있다. 서로 다른 빈일 경우 이 두 빈은 힘-격자 상에서 이웃한 빈이 된다. S_{force}와 T_{force}를 결정하는 방법에 대해선 다음 절에서 설명한다.

임계값 D_{th}는 코어 영역의 이용률(utilization)에 따라 결정되는 값으로써, 각 빈의 셀 밀집도가 최소한 이 값

```
procedure move-cells
input: Sforce, Tforce  // source and target bin in the force grid
       Dth             // density threshold value
1.   begin
2.       Sfine ← fine bin with max density in Sforce
3.       Tfine ← fine bin with min density in Tforce
4.       G ← construct-gain-graph(Sfine, Tfine)
5.       Determine max-gain monotone path P in G
6.       for (each edge E(Borigin, Bend) of P in order from Sfine to Tfine) {
7.           C ← cell in Borigin with gain g(C) = g(Borigin)
8.           Move cell C from Borigin to Bend
9.           if (d(Bend) < Dth) break
10.      }
11.  end
```

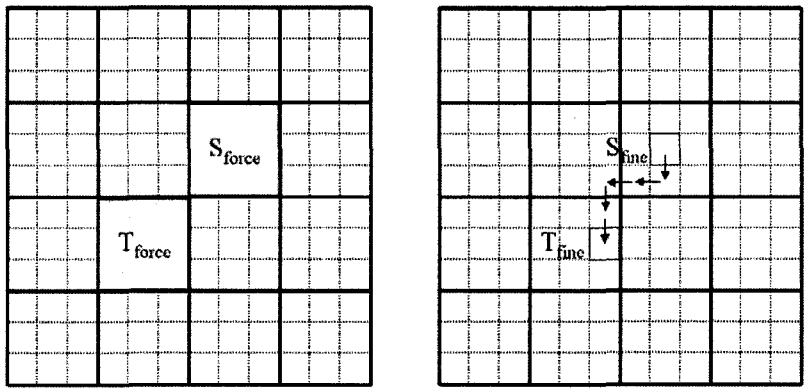
그림 7 프로시저 move-cells

이상이 되도록 밀집도를 조절하는데 사용된다.

S_{force}와 T_{force}가 입력으로 주어지면 단계 2에서는 미세격자를 이용하여 S_{fine}을 구한다. S_{force} 내에 있는 미세 빈 중에서 밀집도가 가장 높은 빈 S_{fine}이 된다. 유사하게 단계 3에서는 T_{force} 내에 있는 미세 빈 중에서 밀집도가 가장 낮은 빈을 T_{fine}으로 선정한다. S_{fine}과 T_{fine}이 결정되면, 2.2.1절에서 설명한 알고리즘 construct-gain-graph(그림 3, 그림 4 참조)를 이용하여 최대이득 경로 P를 미세격자 상에서 구한다(단계 4-5). 최대이득 경로 P를 구한 후, 이 경로를 따라 먼저 S_{fine}에서 이득이 가장 큰 셀을 옮긴다. 이 과정을 그림 8에서 예를 들어 보였다.

그림 8(b)에서 보면 S_{fine}에서 한 셀을 아래로 옮기고, 이어서 우측 빈에서 좌측 빈으로 또 하나의 셀을 옮긴다. 이처럼 최대이득 경로 P를 따라 미세격자 상에서 셀을 리플이동할 때, 셀이 나가는 미세 빈과 셀이 들어오는 미세 빈이 있다. 단계 6-9에선 이런 미세 빈을 각각 B_{origin}, B_{end}로 나타내었다. 경로 P를 따라 셀을 하나씩 옮기는 과정에서, 셀이 들어온 미세 빈의 밀집도(셀이 들어온 후의 밀집도)가 임계값 D_{th}보다 작으면 이동을 중지한다. 임계값 D_{th}를 이용하여 리플이동 과정이 중간에 중지될 수 있도록 한 점은 원래의 Mongrel과는 다른 점이다.

단계 8에서, 미세 빈 B_{origin}에서 최대 이득을 가진 셀을 목적지 빈 B_{end}으로 이동한다. 만약 최대 이득을 가진 셀에 연결된 넷이 넷 제약조건을 가지고 있고 (즉, 타이밍 제약에 관계된 넷이다) 이 셀을 움직였을 때 타이밍 측면에서 배치가 나빠진다면, 알고리즘은 이 셀을 그냥 지나치고 다음으로 최대 이득을 가진 셀을 고른다. 넷 제약조건을 가능한 지킴으로써 FD-Mongrel은 광역



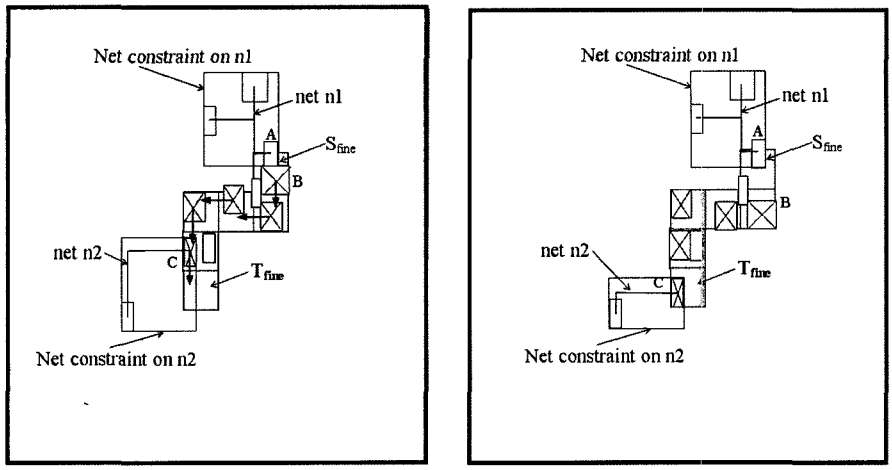
(a) 힘-격자에서 결정된 S_{force}와 T_{force} (b) 미세격자에서 결정된 S_{fine}과 T_{fine} 그리고 최대 이득 단조경로

그림 8 힘-격자에서 S_{force}와 T_{force}가 결정되면, S_{force} 내에 있는 미세 빈 중 가장 밀집도가 높은 빈이 S_{fine}으로, T_{force} 내에 있는 미세 빈 중 가장 밀집도가 낮은 빈이 T_{fine}으로 결정된다. S_{fine}과 T_{fine}이 결정된 후 미세격자 상에서 최대 이득 경로를 구한다.

배치 정제 중에 타이밍이 저하되지 않는 것을 보장해 준다. 그림 9는 이동 중인 셀에 연결된 넷 n1, n2가 제약조건을 가지는 넷일 때 이동 중인 셀이 넷 n1의 제약조건은 떨어뜨리지 않고, 넷 n2의 넷 제약조건은 개선시키는 것을 보여준다. 결과적으로 셀을 이동한 후 타이밍이 개선되는 것을 보여준다. 즉, 그림 9(a)에서 셀 A가 최대 이득을 보장하는 셀이라 하더라도 이는 타이밍 제약을 가지는 넷 n1에 연결되어 있기 때문에 이를 옮길 경우 타이밍이 나빠질 수 있다. 그래서 A 대신 B를 옮긴다. 한편, 셀 C는 역시 타이밍 제약조건을 가지는 넷 n2에 연결되어 있으나 이를 옮기는 것은 오히려 타이밍을 개선시키는 효과가 있다. 그림 9(b)에서 보면 넷 n1의 바운딩박스는 변하지 않은데 비해 넷 n2의 바운딩박스는 감소하였음을 알 수 있다.

3.2.2 프로시저 FD-Mongrel

프로시저 FD-Mongrel의 동작개요를 그림 10에서 보



(a) S_{fine}에서 한 셀이 타이밍 제약조건을 가지는 넷 n1에 연결되어 있고 T_{fine}에서 또 다른 한 셀이 타이밍 제약조건 가지는 넷 n2에 연결되어 있다.

(b) 넷 제약조건을 고려하던서 셀을 이동한 결과

그림 9 제약조건을 가지는 두 넷 n1과 n2에서 셀 이동 후 n1의 제약조건은 그대로 만족시키면서 n2의 넷 제약조건은 개선되는 것을 보여준 예

```
procedure FD-Mongrel
input: global placement P, density threshold value Dth
output: new global placement P'
1.  begin
        // d(B) : cell density of a bin B
2.      while (there is a bin B such that d(B) >Dth w.r.t. P) {
3.          Unlock every bin B in the force grid
4.          Determine force for each bin in the coarse grid
5.          while (there is a bin B such that d(B) > Dth and B is unlocked) {
6.              Sforce ← the most congested bin among unlocked bins // source bin
7.              Tforce ← target (neighbor) bin based on the force vector of Sforce
8.              while (d(Sforce) > d(Tforce) && d(Sforce)>Dth ) {
9.                  move-cells(Sforce, Tforce, Dth) // move some cells from Sforce to Tforce
10.             }
11.             Lock the source bin Sforce
12.         }
13.     }
14.     for (each bin S in the coarse grid that has an over-congested fine bin) {
15.         move-cells(S, S, Dth)   // move cells within the bin S
16.     }
17.     return new placement
18. end
```

그림 10 프로시저 FD-Mongrel

였다. FD-Mongrel에 주어지는 입력 광역배치는 KraftwerkNC를 이용하여 구한 것으로써 어느 정도의 중첩을 가진 것이다. 또 다른 입력 파라메터는 셀 이동을 제어하기 위해 사용되는 임계값 D_{th}인데, 이는 회로의 특성 즉, 코어 영역의 이용률에 따라 자동으로 결정되는 값이다.

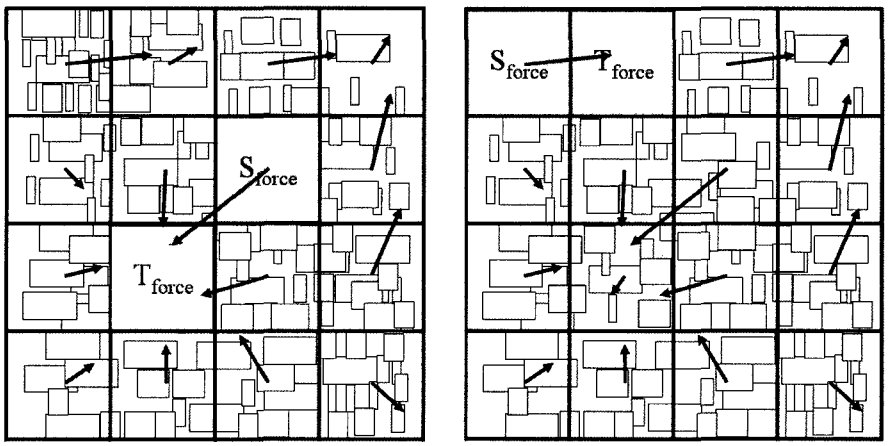
힘-격자에서 셀 밀집도가 D_{th}를 초과하는 빈이 존재하는 동안 단계 2-13이 반복된다. 이 반복문 내에서 셀을 이동하여 셀 중첩의 문제를 해결한다. 셀은 소스 빈에서 타겟 빈으로 이동하는데, 반복이 진행되는 동안 셀들이 어떤 빈에서 다른 빈으로 이동했다가 나중에 반대로 그 빈에서 원래의 빈으로 이동되는 진동현상이 발생할 수 있다. 이를 방지하기 위해 한번 소스 빈이 된 것은 반복문 내에서 다시 소스 빈으로 선정되지 않도록 그 빈을 잠근다(lock).

단계 3에선 우선 모든 빈이 소스 빈이 될 수 있도록 하기 위해 잠금을 푼다. 단계 4에선 힘-격자의 각 빈 중심에서 셀 밀집도에 근거한 분산력 벡터를 구한다. 셀 밀집도가 높을수록 분산력의 값이 크며, 셀을 어느 방향으로 이동하는 것이 효과적으로 중첩을 해결할 수 있는지를 분산력 벡터가 가리킨다. 단계 4를 실행한 후, 밀집도가 D_{th}를 초과하는 힘-격자의 빈들은 그 밀집도 값에 따라 내림차순으로 정렬된다. 내림차순으로 정렬된 빈의 정보를 이용하여 단계 5-12를 반복한다. 단계 6에선 밀집도가 가장 높은 빈을 소스 빈 S_{force}로 선택하고, 단계 7에서는 S_{force}에서 계산된 분산력 벡터를 이용하여 타겟 빈 T_{force}를 결정한다.

그림 11에선 힘-격자와 각 빈의 중심에서 출발하는 분산력 벡터의 예를 보여준다. 그림 11(a)에서 보듯이 셀 밀집도가 가장 높은 빈이 소스 빈, S_{force}가 된다. S_{force}에 이웃한 8개의 빈 가운데서 분산력 벡터가 가리키는 방향에 의해 타겟 빈, T_{force}가 결정된다. 소스 빈에서 이웃한 빈 중에서 타겟 빈을 선택함으로써 배치가 너무 많이 변형되는 것을 방지할 수 있다.

S_{force}와 T_{force}를 정한 후, 3.2.1절에서 설명한 프로시저 move-cells를 이용하여 한 개의 셀을 리플 이동한다. 셀을 옮긴 후 S_{force}와 T_{force} 셀 밀집도는 갱신된다. S_{force}의 밀집도가 T_{force}의 밀집도보다 크고, S_{force}의 밀집도가 D_{th}를 초과하는 동안 프로시저 move-cells가 반복 호출되고, 매 호출 때 마다 미세격자에서 S_{fine}과 T_{fine}이 새로 결정되며, S_{fine}에서 T_{fine}으로 셀이 리플 이동한다. 단계 8-10에서 이런 이동이 반복되기 때문에며, S_{force}에서 T_{force}로 이동되는 셀은 다수가 될 수 있다.

결정된 S_{force}와 T_{force} 사이에서 셀 이동이 완료되면 단계 8-10의 반복문이 한 번 실행된 것이다. 이 과정은 단계 5-12의 반복문 내에 있기 때문에, 단계 4에서 이미 계산된 분산력에 의해 새로운 S_{force}가 결정된다. 이 때 이전에 S_{force}로 선정된 힘-격자 빈은 다시 선정되지 않는다. 왜냐하면 단계 11에서 S_{force}를 잠그고, 잠겨진 S_{force}는 단계 5의 조건에 의해 다시 소스 빈으로 선정되지 않기 때문이다. 새로운 S_{force}가 분산력에 의해 결정되면 T_{force}는 그때의 분산력 벡터에 의해 자동으로 정해진다. 그리고는 새로운 S_{force}와 T_{force} 사이에 셀들이 리플 이동한다. 그림 11(b)는 새롭게 정해진 S_{force}와 T_{force}를 보여준다. 그림 11(a)는 그림 6과 같다. 단계 5-12의



(a) 단계 5-12의 반복문 내에서 처음 결정된 S_{force}와 T_{force}

(b) 단계 5-12의 반복문 내에서 두 번째로 결정된 S_{force}와 T_{force}

그림 11 힘-격자와 각 빈의 중심에서 계산된 분산력, 그리고 분산력에 의해 결정된 소스 빈과 타겟 빈의 예

반복문이 그림 11(a)에서 한 번 실행된 후 몇 개의 셀이 소스 빈에서 타겟 빈으로 이동된 것을 그림 11(b)에서 보였다.

단계 8-10이 처음 시작될 때, S_{force}와 T_{force} 사이의 밀집도 차이가 아주 클 수도 있고 따라서 많은 셀들이 이동될 수 있다. S_{force}에서 T_{force}로 많은 셀들이 이동되면 밀집도의 변화가 크게 되고, 따라서 단계 5-12의 반복이 진행되는 동안 단계 4에서 계산된 분산력이 변화된 상황과 맞지 않는 문제점이 발생한다. 이런 문제점을 방지하기 위해 S_{force}에서 T_{force}로 이동하는 셀의 개수는 S_{force}에 있는 셀들의 10%로 제한한다. 이렇게 함으로써, 프로시저 FD-Mongrel의 단계 5-12의 반복이 진행되는 동안 분산력 정보의 부정확성을 감소시키고 점진적으로 부드럽게 셀을 분산시키는 효과를 얻을 수 있다.

단계 2-13의 반복이 종료되면 힘-격자에서 밀집도가 임계값 D_{th}를 초과하는 빈은 존재하지 않는다. 즉, 힘-격자 상에서 중첩은 해결된 것이다. 하지만 미세격자 상에서 보면 특정 미세 빈에 많은 셀이 중복되어 있을 수 있다. 이를 해결하기 위해 단계 14-16에서는 특정 미세 빈에 많은 셀 중복이 있는 경우, 힘-격자의 동일 빈 내에서 셀들을 이동시켜 중첩을 해결한다.

이 모든 과정을 마치면 밀집도가 임계값 D_{th}를 초과하는 빈은 힘-격자에서 뿐만 아니라 미세격자에서도 존재하지 않게 되어 중첩의 문제가 완전히 해결된다. 또한 프로시저 move-cells를 이용하여 셀을 리플 이동함으로써 중첩을 해결할 뿐만 아니라, 배선길이와 타이밍도 동시에 개선되는 효과를 얻을 수 있다.

4. 실험 및 결과

FD-Mongrel 알고리즘은 Linux 상에서 C++로 구현되었다. MCNC 벤치마크회로[25]를 사용하는 대신, 반도체 제작회사로 잘 알려진 I 사에서 최근에 개발된 마이크로프로세서로부터 얻은 회로를 사용했는데, 이것은 타이밍 분석을 위한 최근의 엔진을 사용하고, 최근에 제조 중인 프로세서로부터 얻은 데이타를 이용하는 것이 타이밍을 위해 넷 제약조건을 만족시키는지의 여부가 더욱 정확하게 연구될 수 있기 때문이다. 실험을 위해 0.18 마이크론 공정에서 설계된 1.5 GHz 마이크로 프로세서로부터 얻은 6개의 회로를 사용하였다. 회로의 크기는 3,374부터 6,223개의 셀을 가진다. 셀과 넷에 따라 지연과 전송시간을 정확하게 측정하는 정적타이밍 분석을 사용하였다. 사용된 셀 라이브러리(cell libarary) 역시 I 사에서 자체 설계되어 이용 중인 것이다.

표 1에서는 실험에 사용된 회로의 특성을 보여준다. 모든 회로에 대해서 두 가지 실험을 했다 (a) 광역배치와 정제(refinement)를 위해 모두 KraftwerkNC를 사용

표 1 실험에 사용된 회로의 특성

Test circuits	Number of cells	Number of nets
Ckt 1	6223	7296
Ckt 2	6039	7081
Ckt 3	5010	5855
Ckt 4	4905	5735
Ckt 5	3399	4150
Ckt 6	3374	4122

표 2 배선 길이의 비교(단위: micron)

Test circuits	KraftwerkNC	KraftwerkNC, FD-Mongrel	% Improvement
Ckt 1	850144	752203	11.52
Ckt 2	841097	791976	5.84
Ckt 3	666798	653781	1.95
Ckt 4	593115	508404	14.28
Ckt 5	423055	394214	6.82
Ckt 6	527916	491762	6.85
Average			7.88

(b) 광역배치를 위해 KraftwerkNC를 사용하고, 정제를 위해 FD-Mongrel을 사용. 두 가지 방법으로 얻는 최종 광역배치는 동일한 방법으로 적법화 되었다. KraftwerkNC의 10번째 반복 때마다 타이밍이 다시 분석되고, 분석된 결과에 따라 수정된 넷 제약조건이 적용되었다.

표 2에서는 두 방법의 최종 배선길이를 비교한 것이다. 광역배치 정제를 위해FD-Mongrel을 사용했을 때 평균 7.88%가 향상되었다. 더 좋은 배선길이 결과가 나온 이유는 KraftwerkNC에 비해 개선된 넷 모델링을 사용한 것과, 넷 제약조건을 고려하지 않아도 되는 경우에 FD-Mongrel에서는 배선길이가 개선되도록 셀을 선택하여 이동했기 때문이다. 즉 최대 이득 단조경로를 구하고 그 경로를 따라 셀을 이동함으로써 배선길이가 개선될 수 있기 때문이다.

표 3에서는 두 방법으로부터 구한 최종배치에서 타이밍에 가장 심각한 문제를 야기하는 경로, 즉 WNS (Worst Negative Slack)를 비교한 것이다. 결과에서 보듯이 FD-Mongrel은 셀을 선택할 때 넷 제약조건을 고려하고 또한 적합한 배선길이 모델링을 사용하기 때문에 WNS 또한 개선시키고 있음을 보인다. WNS의 평균 개선율은 14.2%이다.

표 4는 TNS(Total Negative Slack)이 얼마나 향상되었는지를 보여준다. TNS는 negative margin을 가진 모든 경로들의 slack 합이다. FD-Mongrel은 TNS를 21.25% 향상 시켰다. 이것은 FD-Mongrel이 넷 제약조건 만족을 향상시키고 전체 배선길이를 감소시켰기 때문이다.

표 3 배선 후 최종 WNS(Worst Negative Slack of all timing end points)의 비교 (단위: nanoseconds)

Test circuits	KraftwerkNC	KraftwerkNC, FD-Mongrel	% Improvement
Ckt 1	-0.255	-0.193	24.31%
Ckt 2	-0.165	-0.144	12.73%
Ckt 3	-0.267	-0.26	2.62%
Ckt 4	-1.434	-1.405	2.02%
Ckt 5	-0.314	-0.246	21.66%
Ckt 6	-0.119	-0.093	21.85%
Average			14.20%

표 4 배선 후 최종 TNS(Total Negative Slack of all timing end points)의 비교 (단위: nanoseconds)

Test circuits	KraftwerkNC	KraftwerkNC, FD-Mongrel	% Improvement
Ckt 1	-36.546	-27.652	24.34
Ckt 2	-16.453	-12.574	23.58
Ckt 3	-71.939	-62.859	12.62
Ckt 4	-101.981	-80.643	20.92
Ckt 5	-28.224	-24.755	12.29
Ckt 6	-6.982	-4.626	33.74
Average			21.25

표 5 실행시간 비교 (단위: minutes)

Test circuits	KraftwerkNC	KraftwerkNC, FD-Mongrel	% Improvement
Ckt 1	35	17	51.42
Ckt 2	27	24	11.11
Ckt 3	26	25	3.85
Ckt 4	24	18	25.00
Ckt 5	13	8	38.46
Ckt 6	13	11	15.38
Average			24.21

표 5에서는 정제를 위해 FD-Mongrel을 사용한 방법이 실행시간에서 평균 24.21% 개선되었음을 보여준다. 이것은 KraftwerkNC가 최종단계에서 셀 밀집 분해에 더 많은 시간을 소비하기 때문이다. FD-Mongrel은 더 효율적인 방법으로 같은 효과를 낼 수 있음을 보여준다.

5. 결 론

본 논문에서는 Mongrel과 KraftwerkNC의 기법을 개선시켜 접목한 FD-Mongrel이라는 새로운 배치 접근법을 제시한다. 제안한 기법은 KraftwerkNC의 광역배치와 FD-Mongrel의 정제기법에 기초한 포스 디렉티드 기반 배치 기법이다. 실험 결과들은 배치의 질을 측정하는 몇 가지 중요 항목 모두에서 제안한 기법이 우수함을 보여준다. 해석적 기법을 사용하는 KraftwerkNC의 제약점 즉, 부적합한 넷 모델링과 중첩해결의 약점을 보완하고, 원래의 Mongrel이 가지고 있는 문제점 즉, 배치를 너무 많이 변형함으로써 타이밍을 심각하게 나쁘게 하는 문제점 등을 보완한 개선된 셀 이동 기법을 제안함으로써 이런 좋은 결과를 얻을 수 있었다.

앞으로의 연구에서는 FD-Mongrel을 개선하여 타이밍 개선을 위해 더 나은 넷 모델링 기법을 찾는 것과 셀 선택 및 이동기법, 버퍼링, gate-sizing 등을 동시에 고려한 배치 기법을 찾아야 할 것이다. 또한 배치 후 배선이 불가능하면 배치 과정을 다시 해야 하는 점을 고려할 때 배선을 고려하면서 동시에 타이밍을 개선할 수 있는 기법 또한 주요한 연구 과제이다.

6. 감사의 글

본 논문의 실험을 도와주고 논문의 내용에 대해 같이 의논한 인텔의 Mr. Tung Cao, Dr. Amit Chowdhary, 그리고 Synplicity의 Dr. Bill Halpin에게 감사를 드립니다. 또한 논문의 심사를 맡아서 수고해 주신 익명의 여러 심사위원님들 덕분에 논문이 더욱 명료하게 정리될 수 있었기에 이에 감사를 드립니다.

참 고 문 헌

[1] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron," ICCAD, pp. 203-211, 1998.

[2] P. Villarrubia, "Important Placement Considerations for Modern VLSI Chips," ISPD, pp. 6, 2003.

[3] Yih-Chih Chou and Youn-Long Lin, "A performance-driven standard-cell placer based on a modified force-directed algorithm," ISPD, pp. 24-29, 2001.

[4] 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.

[5] M. Sarrafzadeh and M. Wang, "NRG: Global and Detailed Placement," Proc. of ICCAD, pp. 532-537, 1997.

[6] F. Romeo, A. Sangiovanni-Vincentelli, and C. Sechen, "Research on Simulated Annealing at Berkeley," ICCAD, pp. 652-657, 1984.

[7] C. Sechen and K. W. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement," ICCAD, pp. 478-481, 1987.

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

[9] A. E. Caldwell, A. B. Kahng, and Igor L. Markov, "Can Recursive Bisection Alone Produce Routable

Placements?," DAC, pp. 477-482, 2000.
[10] D. J. -H. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective," ISPD, pp. 18-25, 1997.
[11] M. C. Yildiz and P. H. Madden, "Improved Cut Sequences for Partitioning Based Placement," DAC, pp. 776-729, 2001.
[12] Ke Zhong and S. Dutt, "Effective Partition-Driven Placement with Simultaneous Level Processing and a Global Net Views," ICCAD, pp. 254-259, 2000.
[13] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Optimal End-Case Partitioners and Placers for Standard-Cell Layout," ISPD, pp. 90-96, 1999.
[14] 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.
[15] C. M. Fiduccia and R. M. Matteyses, "A Linear Time Heuristic for Improving Network Partitions," DAC, pp. 175-181, 1982.
[16] 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.
[17] P. N. Parakh, R. B. Brown and Karem A. Sakallah, "Congestion Driven Quadratic Placement," DAC, pp. 275-278, 1998.
[18] X. Yang, B.-K. Choi, and M. Sarrafzadeh, "Routability Driven White Space Allocation for Fixed-Die Standard-Cell Placement," ISPD, pp. 42-47, 2002.
[19] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," DAC, pp. 269-274, 1998.
[20] H. Etawil, S. Arebi, and A. Vannelli, "Attractor-Repeller Approach for Global Placement," ICCAD, pp. 20-24, 1999.
[21] Maogang Wang, X. Yang, Ken Eguro, and M. Sarrafzadeh, "Dragon2000: Placement of Industrial Circuits," ICCAD, pp. 260-263, 2000.
[22] 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.
[23] Karthik Rajagopal, Tal Shaked, Yegna Parasuram, Tung Cao, Amit Chowdhary, Bill Halpin, "Timing Driven Force Directed Placement with Physical Net Constraints," ISPD, pp. 60-66, 2003.
[24] S. Hur and J. Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement," ICCAD, pp. 165-170, 2000.
[25] www.cbl.ncsu.edu/benchmarks/layoutsynth92/

성 영 태
2002년 동아대학교 컴퓨터공학과 학사 졸업. 2004년 동아대학교 컴퓨터공학과 석사 졸업. 2004년~현재 동아대학교 컴퓨터 공학과 대학원 박사과정. 관심분야는 알고리즘, VLSI CAD

허 성 우
1981년 경북대학교 전자공학과 학사 졸업. 1983년 한국과학기술원(KAIST) 전산학과 석사 졸업. 2000년 UIC Dept. of EECS 박사 졸업. 1986년~현재 동아대학교 전자컴퓨터공학부 교수. 2001년~현재 미국 Intel사 Physical Design분야 기술자문위원. 관심분야는 CAD, 알고리즘, 계산 기하학, Combinatorial optimization