# 상세배치를 위한 확장된 인터리빙 기법
## (An Extended Interleaving Technique for Detailed Placement)

오 은 경 † 허 성 우 ††
(Eun Kyung Oh) (Sung Woo Hur)

**요 약** 본 논문에서는 상세배치를 개선할 수 있는 확장된 인터리빙 기법을 제안한다. 기존의 행-기반 인터리빙 기법은 한 행 내에서만 셀의 위치를 이동할 수 있는 제약이 있으며, 모든 셀이 여백 없이 인접해 있다는 가정 하에 적용 가능하였다. 본 논문에서 제안한 확장된 인터리빙 기법은 그런 제약을 극복하여 셀이 다른 행간에도 이동할 수 있도록 하였고, 또한 셀들 사이에 여백이 있는 경우에도 인터리빙 기법을 적용할 수 있도록 하였다. 반도체 설계 회사에서 사용 중인 CAD 툴에 의해 수렴된 상세배치를 제안된 인터리빙 기법을 이용하여 추가로 개선시킨 결과 HP(half perimeter)가 평균 9.5%가 개선되었다.

**키워드** : VLSI 레이아웃, 표준 셀, 배치, 동적 프로그래밍, 인터리빙 기법

***Abstract*** In this paper we propose an extended interleaving technique to improve a detailed placement. The existing row-based interleaving technique allows cells to move only within a row and it can be applied when there is no space between cells. The proposed extended-interleaving technique releases such constraints so that cells can move along with a vertical line parallel to a y-axis and space between cells is properly handled. Converged detailed-placements by a mature CAD tool have been improved by the proposed interleaving technique by 9.5% on average in half-perimeter wire length.

**Key words** : VLSI Layout, Standard Cell, Placement, Dynamic Programming, Interleaving Technique

## 1. 서 론

### 1.1 관련 연구

VLSI 회로의 기술 발달로 인해 오늘날에는 한 칩 내에 수십 또는 수백만 개의 트랜지스터가 있기 때문에 이를 설계하는 과정은 여러 단계를 거치고, 각 과정은 적합한 CAD 툴들을 이용하여 설계하게 된다. 여러 과정 가운데 물리설계는 매우 복잡하고 시간이 많이 걸릴 뿐 아니라 이 설계결과는 칩의 면적, 성능 등에 절대적인 영향을 미치기 때문에 물리설계 과정 자체도 여러 세부 단계를 거치게 된다. 물리설계 과정의 여러 세부 단계 중 하나가 배치인데, 이는 최근에 물리설계 과정에서 매우 중요한 역할을 하는 과정으로 많은 관심을 끌고 있다[1,2]. 배치에 의해 회로의 배선길이와 배선 가능성이 크게 영향을 받고 따라서 배선 지연에도 큰 영향을 미치기 때문이다. 배선지연은 회로 성능을 떨어뜨리는 주된 요인이 되므로 배치는 회로의 성능에 매우 중요한 영향을 미치는 단계라고 볼 수 있다.

전통적인 표준 셀 배치의 목표는 셀들이 중첩되지 않도록 하면서 배선길이를 최소화하는 것이다. 배선길이를 기반으로 한 최근의 배치기법은 크게 세 가지로 분류될 수 있다. 첫 번째 부류는 처음부터 중첩을 허용하지 않으면서 배치 과정을 처리하는 것이다. TimberWolf[3,4]는 simulated annealing을 기반으로 하는 잘 알려진 배치 툴이다. 이 기법에선 주어진 배치에서 두 셀의 위치를 맞바꾸거나 또는 한 셀의 위치를 옮기는 작업 등을 반복함으로써 새로운 배치를 얻어 가는데 이 기법을 사용하여 배치를 구한 결과들이 많이 발표되었다[5-7]. 이는 주어진 초기 배치로부터 다양한 셀 이동을 통하여 변형된 배치를 얻는 방법으로, 좋은 배치를 얻을 수 있는 방법으로 알려져 있으나 시간이 많이 소요된다는 단점이 있다. 왜냐하면 검색해야 할 해 공간의 크기에 비해 셀 교환을 통한 해 공간에서의 이동은 너무나 단순하기 때문에 방대한 해 공간을 검색하는데 많은 시간이 요구되기 때문이다. 두 번째 부류는 계층구조에서 분할

· 본 연구는 한국과학재단 목적기초연구(과제번호: 2003-004-27-040) 지원으로 수행되었음
† 학생회원 : 동아대학교 컴퓨터공학과
ekoh@donga.ac.kr
†† 종신회원 : 동아대학교 전자컴퓨터공학부 교수
swhur@daunet.donga.ac.kr
논문접수 : 2004년 7월 10일
심사완료 : 2006년 5월 10일

기법을 적용함으로써 광역배치(global placement)로부터 상세배치(detailed placement)를 점진적으로 구해가는 방법을 사용한다. 최소 컷을 재귀적으로 얻음으로써 가능한데, 이 기법을 사용하는 툴로는 Capo[8], Quad[9], FengShui[10], Snap-on[7] 등[11]이 있다. Caldwell[12]이 지적한대로 이 기법을 사용하는 툴들은 대부분 변형된 KL알고리즘[13]이나 FM알고리즘[14]을 이용하고 있다. 세 번째 부류는 해석적 기법 또는 FD(Force-Directed) 기법을 사용하는 것이다[16-22]. FD 방법은 그 변화가 매우 다양하다. 셀의 위치를 구하기 위한 식을 어떻게 푸느냐에 따라 중복의 정도가 달라지고 결국 해의 질이 달라진다. 또한 해를 구하는 시간도 다르다. FD 방법은 근본적으로 중복을 허용하기 때문에 중복을 어떻게 해결하는가에 따라 해의 질이 크게 좌우된다. Goto는 iterative FD 방법을 이용한 독특한 알고리즘을 제안하였다[23]. Parakh[24]와 Yang *et al*[25]은 라우팅을 고려한 FD 배치 알고리즘을 제시하였으며, Etawil 등은 셀 중복을 가능한 줄이기 위해 인장력뿐만 아니라 척력 (repelling force)을 이용한 기법도 소개하였다[20]. Snap-on[7], Dragon[26,27]은 광역배치를 최적화시키기 위해 분할기법을, 상세배치 최적화를 위해 어닐링 기법을 사용한다.

표준 셀 배치에서 상세배치를 개선하는 방법 중 하나는 각 행에 있는 셀들의 순서를 다시 조절함으로써 가능한데 이는 결국 좋은 선형배치를 구하는 문제와 동일하고 이를 위한 연구도 많이 발표되었다[30-34]. 특히 Kahng과 Tucker[34]는 동적 프로그램 기법을 이용하여 효과적으로 셀의 위치를 결정함으로써 상세배치를 개선하였다.

셀 배치 과정은 다양한 제약 조건을 만족하도록 수행되어야 하며 동시에 목적 함수를 최적화해야 하는 복잡한 과정이다. 이러한 문제를 효과적으로 해결하는 방법 중 하나가 [28,29,7,26,27]에서처럼 다단계(multi-level) 혹은 광역배치(global placement)와 상세배치(detailed placement)의 2단계 과정을 수행하는 것이다. [29]에서는 middle-down 접근기법과 network-flow 기법을 이용하여 광역배치를 구하고, 최적 인터리빙이라 불리는 기법을 이용하여 상세배치를 개선한다.

최근에 발표된 대부분의 배치 휴리스틱은 첫 단계에서 광역 배치를 얻고, 다음 단계에서 상세배치를 개선해 가는 기법을 주로 사용하고 있다. 좋은 광역 배치를 구하는 것이 궁극적으로 좋은 배치를 얻을 수 있는 필수 과정이기 때문에 광역배치를 위해 여러 가지 기법을 사용한다. 또한 광역배치가 좋다 하더라도 광역배치는 많은 중첩을 허용하고 있기 때문에 중첩을 해결해나가는 방법에 따라 상세배치의 질이 크게 영향을 받는다.

본 논문에서는 [29]에서 제안한 행-기반 인터리빙 기법의 단점을 개선하여 더욱 효과적으로 상세배치를 최적화할 수 있는 기법을 제시하였다. 본 논문에서 제안한 기법의 장점은 다음과 같이 요약될 수 있다.

① [29]에서 제안한 행-기반 인터리빙 기법에선 셀들이 한 행 내에서만 위치를 옮기는 제약, 즉 셀 이동이 수평방향으로만 가능하다는 제약이 있으나, 제안한 기법에선 수직방향으로도 셀을 옮길 수 있다. 셀들을 코어 영역 내에서 수평방향 뿐만 아니라 수직방향으로도 옮길 수 있기 때문에 최적의 위치에 배치할 수 있다.

② [29]에서 제안한 인터리빙 기법에선 셀들이 여백없이 인접한 경우에만 처리 가능하기 때문에 셀들 사이에 여백을 가지고 있는 실제 회로에서는 정확하게 최적화 시키지 못하는 한계점이 있다. 그러나 본 논문에 제안한 기법에선 셀 사이에 여백이 있는 경우에도 효과적으로 최적의 위치를 찾을 수 있도록 하였다.

③ I사에서 사용 중인 CAD 툴에 의해 이미 수렴된 상세배치에 제안한 기법을 행과 열에 대해 반복 적용한 결과 배선길이가 평균 9.5% 개선되었다. 또한 평균 실행시간도 1~2분 정도이어서 제안한 기법이 효과적임을 실험적으로 보여 주었다.

논문 구성은 다음과 같다. 1.2절에서는 배선길이 추정을 위한 넷 모델링에 대해 간략히 설명한다. 2장에서는 행-기반 최적 인터리빙 기법에 대해서 설명하고, 셀과 셀 사이에 여백이 있는 경우 셀의 위치를 결정하는 방법에 대해 설명한다. 3장에서는 열-기반 인터리빙 기법과 셀 이동 후 중첩 문제를 어떻게 처리하는지에 대해서 설명한다. 4장에서는 실험결과를 보이며, 5장에서는 결론을 맺는다.

### 1.2 넷 모델링

배선길이를 추정하기 위해 효율적이면서 널리 사용되고 있는 방법으로 HP(Half-Perimeter)모델을 사용한다. HP는 그 넷에 관계된 모든 핀을 둘러싸는 직사각형을 구한 후, 둘레 길이의 반을 계산한 것이다. 이것은 핀 수가 2또는 3인 넷에서 우회배선(detour)이 없을 경우 정확한 길이가 되고, 핀 수가 4 이상인 경우 배선길이의 하한값(lower bound)이 된다. 실제 회로에선 대부분의 넷이 2개 혹은 3개의 핀으로만 구성되어 있기 때문에 배선길이를 추정하기 위해 HP를 사용해도 큰 오차가 없다. 계산의 편의상 각 핀은 셀의 중앙에 위치한다고 가정한다.

## 2. 행-기반 최적 인터리빙(Row-Based Optimal Interleaving)

### 2.1 알고리즘의 개요

본 절에서는 행-기반 인터리빙의 동작원리를 간략하

게 보인다. 행-기반 인터리빙 기법은 셀들이 속한 현재의 행 내에서 일부 셀들의 위치를 재조정함으로써 배선 길이를 줄일 수 있는 효과적인 기법이다. 이때 다른 행에 있는 셀들은 모두 고정되어 있다고 가정한다. 어떤 행에 있는 셀들을 그 행 내에서 상대적인 위치를 바꾸어 가장 좋은 순서를 구하는 방법 중 한 가지는 모든 가능한 순서를 다 검사해 보는 것이다. 그러나 이럴 경우, 한 행 내에 있는 셀들의 개수가 k라고 할 때 모든 가능한 셀들의 재배열 방법은 k! 개가 되고, 실제 회로에서 대부분 $k > 100$ 이므로 모든 가능한 재배열을 검사하기엔 현실적으로 불가능하다. 한 행 내에 있는 모든 셀에 k! 개의 재배열을 검사하는 대신, 크기 W인 어떤 윈도우를 설정하여 그 윈도우 내에 있는 셀들에 대해 W! 개의 가능한 재배열을 검사하고, 윈도우를 이동하여 이 과정을 반복하는 방법도 고려해 볼 수 있다. 이럴 경우에도 W의 크기가 7이상이면 실행시간이 많이 걸려 현실적으로 사용하기 어렵다.

행-기반 인터리빙 기법은 이런 문제점을 극복하면서 다양한 재배열을 효과적으로 검사할 수 있는 방법으로써 전체적인 알고리즘은 그림 1과 같다.

행-기반 최적 인터리빙 알고리즘의 단계 1에서 윈도우 크기 W 내에서 부분열 A와 B를 얻는 과정을 그림 2에서 예로 보였다. 여기서 윈도의 크기 W는 7로 두었으며, 각 셀이 어느 부분열에 속하게 될지는 무작위로 결정된다. 그림 2의 두번째 줄에서 보였듯이 두 부분열 A와 B 내에서 셀들의 순서는 원래의 상대적인 순서를 그대로 유지한다. 두 부분열이 결정되면 이를 최적 인터리빙이라 불리는 동적 프로그래밍 기법을 이용하여 개선된 새로운 셀 순서를 얻게 된다. 두 부분열 내에서 상대적인 순서를 유지하는 이유는 해 공간이 방대하게 커지는 것을 방지하기 위함이다.

주어진 한 부분열에 속한 셀들에 대해 상대적인 위치를 유지하면서 두 부분열을 인터리브시켜 새로운 순서를 구하여도 그 셀들 간의 순서가 계속 유지되지는 않는다. 왜냐하면 알고리즘이 반복되어 적용되기 때문에 셀들의 상호 순서는 얼마든지 바뀔 수 있기 때문이다. 예를 들어, 그림 2에서는 한번 인터리빙을 통해 $a_1, a_2, a_3$의 상대적인 순서를 유지하면서 구한 새로운 순서가 $b_1, a_1, b_2, b_3, a_2, b_4, a_3$로 되나 다음 반복 시에 이 열이 다시 A와 B로 분할될 경우 $A = b_1, a_1, b_2, a_3$, $B = b_3, a_2, b_4$가 될 수 있고 이를 인터리빙하면 우리는 $b_3, a_2, b_1, a_1, b_4, b_2, a_3$처럼 새로운 순서를 구할 수 있다. 이렇게 되면 두 번의 인터리빙 적용으로 인해 원래의 순서 $a_1, b_1, a_2, a_3, b_2, b_3, b_4$

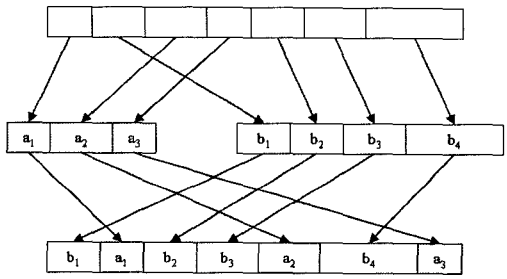


그림 2 행-기반 최적 인터리빙의 동작을 보여주는 예

입력 : 상세배치, 윈도우 크기 W
출력 : 개선된 상세 배치

1. 처리중인 행 내에서 윈도우 사이즈 W 내에 포함되는 셀들을 먼저 결정한다. 즉, 연속된 W개의 셀을 선택한 후, 그것들을 두 개의 부분열 A, B로 분할한다. 이 때 어느 셀을 어느 부분열에 포함시킬지는 무작위로 결정하며, 각각의 부분열 A, B에 속한 셀의 상대적인 순서는 원래의 상대적인 순서를 그대로 유지한다 (그림 2참조).
2. 2.2절에서 설명하는 동적 프로그래밍 기법을 사용하여 두 부분열 A와 B에 있는 셀들로부터 유도된 새로운 순서를 얻는다. 이때, 새롭게 얻은 순서에는 W개의 셀들이 있게 되며, 부분열 A 또는 B에 속한 셀들의 상대적인 순서는 새롭게 얻은 순서에서도 지켜진다.
3. 단계1~2를 첫째 행부터 마지막 행까지, 그리고 각 행 내에서는 윈도우를 좌측에서 우측으로 이동하면서 반복 적용한다. 윈도우를 우측으로 이동시킬 때 이전에 윈도우에 포함된 셀 들 중 반이 새로운 윈도우에 포함되도록 윈도우 거리를 W/2 만큼만 이동시킨다.

그림 1 행-기반 최적 인터리빙 알고리즘

와는 전혀 다른 순서를 얻을 수 있게 된다. 이처럼 인터리빙이 반복 적용되면서 얼마든지 좋은 순서를 구할 수 있기 때문에 A와 B에서 상대적인 순서를 유지시키는 것이 좋은 결과로의 접근을 방해하지는 않음을 알 수 있다.

### 2.2 행-기반 최적 인터리빙을 위한 동적 프로그래밍 기법

본 절에서는 주어진 두 부분열 A와 B로부터 유도된 새로운 최적의 순서를 얻기 위한 동적 프로그래밍 기법에 대해 설명한다. 여기서 말하는 최적의 순서란 부분열 A와 B에 속한 셀들의 상대적인 순서를 지키면서 두 부분열을 인터리브시킬 때 얻을 수 있는 모든 가능한 순서 가운데 최적인 것을 의미한다. 다시 말해, 부분열 A, B의 크기가 각각 $n,m$ 일 때 ($W=n+m$), 각 부분열 내에 속한 셀들의 상대적인 순서를 유지하면서 새로운 W개의 열을 얻을 수 있는 모든 가능한 순서의 수는 $\binom{n+m}{n}$이 되며, 제시한 알고리즘에서는 동적 프로그래밍 기법을 이용해 이 가능한 순서들 가운데 최적인 순서를 찾는다. W의 크기가 20일 경우, 두 부분열이 각각 어떤 크기로 분할되는가에 따라 결과가 달라질 수도 있지만, 만약 n과 m이 각각 10이라고 할 때 가능한 순서가 모두 184,756가지나 되어 이 중 가장 최적인 순서를 빠른 시간에 찾을 수 있다.

두 부분열이 $A=a_1,a_2,...,a_n$, $B=b_1,b_2,...,b_m$라고 하자. $S_{i,j}$를 $a_1,a_2,...,a_i$ $(i\leq n)$와 $b_1,b_2,...,b_j$ $(j\leq m)$ 로부터 얻은 최적 순서라고 하고, $C(S_{i,j})$는 $S_{i,j}$의 비용이라고 하자. 여기서 $S_{i,j}$의 비용이란 $S_{i,j}$에 속한 셀들로 인해 결정된 배선길이의 합을 의미한다. 1.2절에서 설명했듯이 배선길이를 위해 본 논문에서는 HP(Half-Perimeter)모델을 사용한다. 이 때 배선길이는 x 성분만 고려하면 되기 때문에 y 좌표에 의한 길이는 무시된다. 또한, 두 부분열 A와 B에 속한 셀들을 제외한 나머지 모든 셀들은 고정되어 있다고 가정한다. 그림 3에서 $C(S_{i,j})$의 예를 보였다. 계산의 편의를 위해 모든 핀은 셀의 중앙에 있다고 가정하였기 때문에 넷의 경계는 항상 셀의 중앙이 된다. 그림에서 보인 넷은 핀이 모두 2개라고 가정하였고, 인터리빙 과정에서 $S_{1,2}$의 상황과 그에 따른 넷의 상태를 보였다. 인터리빙을 위한 윈도우의 좌측 경계의 x-좌표가 100이고, 넷 $n_1$은 셀 $a_1$과 윈도우 좌측 경계 밖에 있는 어떤 셀과 연결되어 있기 때문에 $n_1$이 $C(S_{1,2})$에 기여하는 값은 170(=270-100)이 된다. 넷 $n_2$는 셀 $a_1$과 우측에 있는 어떤 셀과 연결되어 있어 $n_2$가 $C(S_{1,2})$에 기여하는 값은 170(=440-270)이 된다. 이런 방법으로 $n_1$부터 $n_6$까지 각 넷이 $C(S_{1,2})$에 기여하는 값은 각각 170, 170, 60, 60, 110, 60이 되어 $C(S_{1,2})$ =630이 된다.

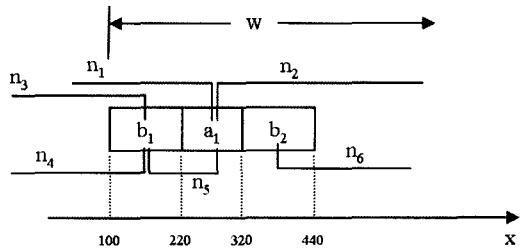


그림 3 인터리빙 시 $C(S_{i,j})$를 계산하는 예

$S_{i,j}$는 $a_1,a_2,...,a_i$ $(i\leq n)$와 $b_1,b_2,...,b_j$ $(j\leq m)$ 로부터 얻은 최적 순서이고, 각 부분열에 속한 셀들의 상대적인 순서는 유지되기 때문에 $S_{i,j}$의 마지막 셀은 $a_i$ 이거나 $b_j$ 이다. $S_{i,j}$의 마지막 셀이 $a_i$ 이라는 것은 $S_{i-1,j}a_i$ 가 최적이라는 것을 의미하고, 유사하게 $S_{i,j}$의 마지막 셀이 $b_j$ 이라는 것은 $S_{i,j-1}b_j$ 가 최적이라는 것을 의미한다. 즉, $S_{i,j}$의 최적 순서는 $S_{i-1,j}$과 $S_{i,j-1}$을 이용하여 얻을 수 있기 때문에 $S_{i,j}$를 구할 때 동적 프로그래밍 기법을 사용할 수 있다. 부분열 A와 B를 인터리빙하는 궁극적인 목표는 $S_{n,m}$을 찾는 것이다.

동적프로그래밍을 위한 재귀 관계(recurrence relation)는 다음과 같이 나타내어진다. 정의대로 $S_{0,0}$는 셀이 하나도 포함되지 않은 부분열이므로 $S_{0,0}$는 공집합이 되고, 따라서 이 부분열에 대응하는 비용(길이)도 0이 된다. $S_{i,j}$는 이미 설명한 대로 둘 중 하나가 되어 다음처럼 나타내어진다.

$$S_{0,0}=\varnothing$$

$$C(S_{0,0})=0$$

$$S_{i,j}=\begin{cases}S_{i-1,j}a_i, & \text{if } C(S_{i-1,j}a_i)<C(S_{i,j-1}b_j)\\ S_{i,j-1}b_j, & otherwise\end{cases}$$

$S_{n,m}$을 구하는 데 필요한 시간은 $O(nm+p(n+m))$이 됨을 쉽게 알 수 있다. 여기서 $p$는 윈도우 내에 있는 각 셀에 관계된 핀(pin) 수의 최대값이며, 이는 회로의 설계가 끝나면 상수가 된다. 따라서 이 알고리즘의 실행시간은 매우 효과적임을 알 수 있다.

### 2.3 여백을 고려한 행-기반 최적 인터리빙

회로에 따라 코어 영역의 사용율이 크게 다른데 이 사용율이 떨어질수록 셀과 셀 사이에 여백(white space)이 많이 생긴다. 행-기반 최적 인터리빙은 모든 셀들 사이에 여백이 없이 인접해 있다는 가정 하에 주어진 두 부분열을 가지고 새로운 최적의 순서를 찾을

수 있으나 셀 사이에 여백이 있는 경우엔 이 기법을 직접 적용할 수 없다.

이런 여백을 처리하는 방법 중 하나는 그림 4에서 보인 것처럼 주어진 W에 의해 결정된 셀들 사이에 있는 여백을 모두 없애 셀들을 중앙으로 옮긴 후 최적 인터리빙 기법을 적용하는 것이다. 그런 다음 얻은 새로운 순서에 따라 셀들 사이에 여백을 적당히 삽입하는 기법이다. 그림 4(a)는 W=6인 경우, 인터리빙을 적용하기 전의 순서를 보여주며, 그림 4(b)는 6개의 셀들을 중앙으로 이동 한 후의 순서를 보여준다. 그림 4(c)에서는 두 부분열 A=bde와 B=acf를 보여주며, 그림 4(d)에서는 이 두 부분 열을 인터리브하여 새롭게 구한 순서 bdaecf에서 각 셀의 상대적인 순서를 유지하면서 셀들 사이에 여백을 삽입하고, 그 후 셀들의 위치를 조정한 것이다.

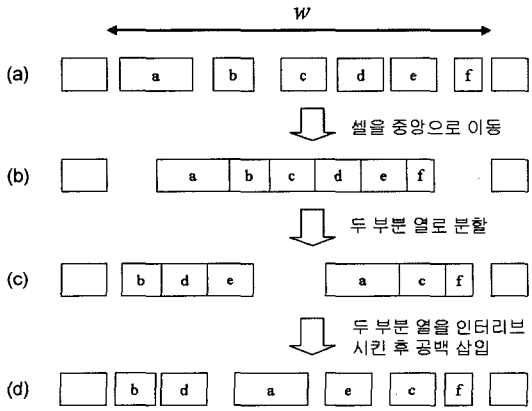


그림 4 여백을 고려한 행-기반 인터리빙: W개의 셀을 중앙으로 이동 후 인터리빙

그림 4에서처럼 여백을 처리할 경우 셀들을 (b)에서처럼 중앙으로 이동하면 셀들의 x-좌표가 원래의 값과 모두 다르게 되어 그런 상황에서 최적의 순서를 찾는다 해도 실제적으로 (d)에서 얻은 순서는 배선길이를 크게 개선시키지 못하는 경우가 발생한다. 즉, (a)에서 (b)으로 변환하는 과정에서 원래의 정보를 잃어버리는 문제로 인해 인터리빙 후 구한 새로운 순서가 원래의 것에 비해 개선되었다는 것을 보장하지 못한다.

이런 문제를 개선하기 위해 우리는 그림 4(a)에서 그림 4(b)으로 변환하는 대신 여백을 적당한 크기를 가지는 더미(dummy) 셀로 변환하여 처리한다. 이 때 새로 추가된 더미 셀 수만큼 W의 크기를 증가시켜 더미 셀을 마치 실제 셀처럼 간주하여 처리함으로써 셀 위치 변환에 따른 정보의 손실을 막는다. 이런 과정의 개요를 그림 5에 보였다.

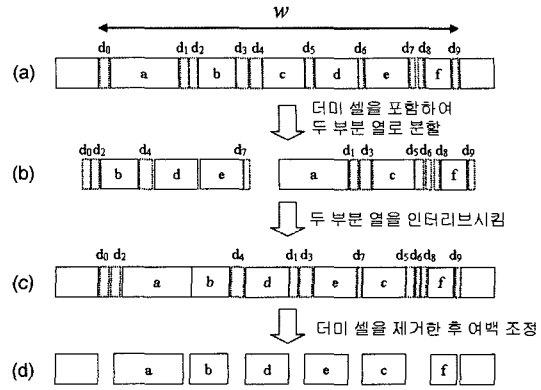


그림 5 여백을 더미 셀로 대치한 후 행-기반 인터리빙 처리 과정

각 더미 셀의 너비는 여백의 크기에 따라 다르게 결정되는데, 회로 전체에서 제일 작은 셀의 너비의 반 또는 그 이하가 되도록 조정된다. 회로에서 제일 작은 셀의 너비가 $w_{\min}$ 라고 하고, 현재 여백의 크기가 $S$ 라고 하자. 그리고 그 여백에 의해 $k$ 개의 더미 셀이 생성되었다고 할 때, 각 더미 셀 $d_i (0 \le i < k)$ 의 너비 $w(d_i)$ 는 $\frac{2S}{w_{\min}}$ 또는 $\frac{2S}{w_{\min}}+1$ 된다. 물론 $S = \sum_{i=0}^{k-1} w(d_i)$ 이 된다.

그림 5에서 보인 예에서는 W=6으로 두었고 따라서 셀들은 abcdef 총 6개가 된다. 각 셀 사이에 있는 여백을 더미 셀로 대치하는데 앞에서 설명하였듯이 여백의 크기에 따라 더미 셀의 크기가 달라지며, 그림 5(a)에서 보듯이 10개의 더미가 추가된다. 따라서 우리는 W=16으로 간주하게 된다. 그림 5(b)는 분할된 두 부분열 $A = d_0 d_2 b d_4 d e d_7$ , $B = a d_1 d_3 c d_5 d_6 d_8 f d_9$ 를 보여주며, 그림 5(c)에선 이 두 부분열을 인터리브시켜 구한 새로운 순서 $d_0 d_2 a b d_4 d d_1 d_3 e d_7 c d_5 d_6 d_8 f d_9$ 를 보였다. 그 후 더미 셀을 제거하고, 각 셀이 이동할 수 있는 범위 내에서 위치를 재조정하여 가장 최적의 x-좌표를 결정한 것을 그림 5(d)에서 보였다.

## 3. 열-기반 인터리빙(Column-Based Interleaving)

### 3.1 열-기반 인터리빙 기법의 개요

열-기반 인터리빙이란 행-기반 인터리빙과 기본 동작 원리는 동일하나, 가상의 수직선을 고려하여 그 수직선을 중심으로 셀들이 y축을 따라 이동할 수 있도록 하는 기법이다. 부분열 $S_{i,j}$ 의 최소 비용 $C(S_{i,j})$ 는 여기서 한 넷의 y 성분만을 고려하여 결정된다는 점이 행-기반 인터리빙과 또 다른 점이다. 열-기반 인터리빙 기법을 행-기반 인터리빙과 더불어 적용하면 2차원 평면 위에서 셀들의 최적의 위치를 찾아 움직일 수 있기 때문에

이 두 기법을 차례로 적용하면 코어 영역 내에서 셀들의 국부 최적의 위치를 찾을 수 있다.

열-기반 인터리빙에선 셀들의 너비가 다르기 때문에 가상의 수직선을 따라 셀들이 이동할 때 셀들이 중첩되는 문제가 발생한다. 이 문제로 인해 열-기반 인터리빙은 행-기반 인터리빙에서처럼 주어진 재귀 관계식을 이용하여 두 부분열을 이용한 최적의 인터리빙을 찾을 수 없다. 대신 셀들이 중첩되는 경우 이를 고려하여 비용을 계산함으로써 셀들의 이동을 제어할 수 있다. 그림 6에서는 열-기반 인터리빙 알고리즘의 개요를 보였다.

### 3.2 수직선에 관계된 셀 결정

$x=l$ 인 수직선 $vl$ 이 설정되면 $vl$ 을 중심으로 x 값이 δ범위 내에 있는 셀들을 찾고, 그 범위에 포함된 셀들이 다수인 경우 각 행에서 가장 가까운 셀들을 선정하며, 만약 어떤 행에서 그 범위 내에 있는 셀이 없으면 그 행에는 더미 셀을 선정한다. 더미 셀이란 $l-\delta \le x \le l+\delta$ 범위 내에 어떤 셀도 존재하지 않을 때 $vl$ 을 중심으로 그 행에서 최대로 크게 잡을 수 있는 여백을 셀로 간주한 것이다. 그림 7에서는 W=6(즉, 6행에 걸쳐 셀들을 선택함)인 경우에 각 행에서 어떤 셀들이 선택되는지를 보였다. 넷째 행의 셀 d는 왼쪽에 있는 셀보다 $vl$ 에 조금 더 가까워서 선택되며, 점선으로 표시된 셀 b는 더미 셀을 나타낸다. δ 값은 자동으로 결정되는 값으로서 더미 셀 수가 W/2 이하가 되도록 조정된다.

### 3.3 열-기반 인터리빙 비용 계산

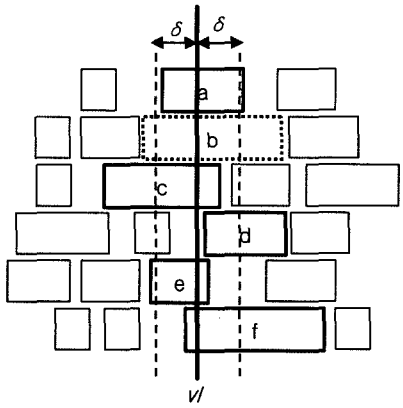


그림 7 수직선 $vl$ 에 관계된 셀들

수직선 $vl$ 에 관계된 셀들이 결정되면 행-기반 인터리빙에서처럼 원래의 상대적인 순서를 유지하면서 이들을 두 부분열로 나눈다. 그리고 나누어진 두 부분열을 아래의 재귀식에 따라 인터리브시킨다. 행-기반 인터리빙에서 사용된 재귀식과 유사하지만 셀들이 이동 후 너비의 차이로 인해 중첩될 수 있기 때문에 simulated annealing 기법에서 채택한 것처럼, 이를 고려하여 비용 계산 시 새로운 항목 $C_{ovrlp}$ 을 추가하여 셀 교체 시 중첩이 발생하면 중첩에 의한 페널티를 비용 계산에 부여함으로써 중첩이 발생하는 부분 순서를 기피할 수 있도록 하였다. 즉, 부분열의 비용 $C(S_{i-1,j}a_i)+C_{ovrlp}$는 셀 $a_i$

| 입력 : 상세배치, 윈도우 크기 W |
|---|
| 출력 : 개선된 상세배치 |
| 1. $x=l$인 수직선 $vl$(vertical line)을 고려하고, $vl$에 관계된 셀들을 결정한다. $vl$에 관계된 셀이란 더미 셀과 실제 셀들로 구성 된다 (그림 7 참조). |
| 2. 단계 1에 의해 결정된 셀들 (더미 셀 포함) 중에서 위로부터 윈도우 W 내에 있는 셀들을 두 개의 부분열 A, B 로 분할한다. 이때 분할은 무작위로 결정한다. 각각 분할된 셀들은 상대적 순서를 유지하면서 A와 B의 부분열로 분할된다. |
| 3. 단계 2에서 결정된 두 부분열 A, B에 있는 셀들의 상대적 위치를 유지하면서 A와 B를 인터리브(interleave)시켜 비용을 최소화 하는 새로운 순서를 얻는다. 이 때 동적 프로그래밍 기법을 사용한다. |
| 4. 윈도우를 아래로 이동하면서 단계 2와 3을 반복 적용한다. |
| 5. $l=l+\alpha$ 로 수정한 후, 즉 $vl$을 우측으로 이동시킨 후 단계 1~4과정을 반복한다. |

그림 6 열-기반 인터리빙 알고리즘

가 부분열 $S_{i-1,j}$ 뒤에 놓일 때 행-기반 인터리빙에서처럼 배선길이(이 때는 배선의 y-성분에 의한 길이임)로 인한 비용 $C(S_{i-1,j}a_i)$와 $a_i$가 그 부분열 뒤에 위치함으로써 중첩이 발생할 수 있으므로 이에 의한 페널티 $C_{ovrlp}$를 총 비용에 포함시킨 것이다.

$S_{0,0} = \varnothing$

$C(S_{0,0}) = 0$

$$S_{i,j} = \begin{cases} S_{i-1,j}a_i, & \text{if } C(S_{i-1,j}a_i) + C_{ovrlp} < C(Si,j-1b_j) + C_{ovrlp} \\ S_{i,j-1}b_j, & otherwise \end{cases}$$

셀들이 이동된 후 그 행 내에서 주변에 있는 셀들 위치를 고려하여 중첩되지 않는 경우엔 $C_{ovrlp}$ 의 값은 0이 되고, 최적의 x 위치를 결정하여 셀을 배치한다. 만약 셀이 이동된 후 주위의 셀들과 중첩이 된다면 $C_{ovrlp}$ 의 값은 중첩된 너비에 비례하여 그 값이 결정된다. $C_{ovrlp}$ 항목을 비용에 추가하는 이유는 셀을 수직으로 옮긴 후 다른 셀들과 중첩이 되는 경우, 셀들이 중첩되지 않도록 주변의 셀들을 수평적으로 이동하여야 하기 때문이다. 이처럼 중첩 해결을 위해 주변 셀들을 수평적으로 이동할 경우 넷의 x-성분이 증가할 수 있다. 이런 점을 고려하여 열-기반 인터리빙 적용 후 중첩이 발생한다면 이에 따른 페널티를 비용에 부과하는 것이다. 이 $C_{ovrlp}$ 의 값은 정확하게 계산된 값이라기보다 추정치를 사용하게 되는데 이는 열-기반 인터리빙의 수행 속도를 떨어뜨리지 않기 위함이다. 이 값을 정확하게 계산하려면 중첩이 생길 때 마다 주변의 셀들이 어떻게 이동되어야 하며, 그에 따른 넷의 x-성분이 얼마나 증가하는지를 정확히 계산한 후 그 증가된 값을 $C_{ovrlp}$ 으로 두어야 하는데 이는 수행속도를 상당히 떨어뜨리는 요인이 된다.

그림 7에서 보인 것처럼 선택된 셀 abcdef를 두 부분열로 나눈 것이 각각 A=abd, B=cef라고 할 때, A와 B를 인터리브 한 과정을 그림 8에서 개략적으로 보였다. 최종적으로 구한 새로운 순서가 ceabfd가 되며, 셀 b는 그림에서 이동 후 다른 셀과 중첩되는 것처럼 보이나 더미 셀이기 때문에 이는 중첩과 상관이 없게 되어 이와 관련된 $C_{ovrlp}$ 의 값은 0이 된다. 반면 셀 f는 이동 후 좌우의 셀과 중첩이 되기 때문에 $C_{ovrlp}$ 의 값이 0보다 크게 되어 중첩에 기인한 대가가 새로운 순서를 결정하는 데 반영된다. 또한 셀 f의 위치는 좌우 셀과 중첩되는 부분의 크기가 같도록 중앙에 놓이게 된다. 그리고 이 중첩은 좌우 셀들을 이동시켜 해결하게 된다.

## 4. 실험 및 고찰

본 논문에서 제안한 기법은 C언어로 구현하여 펜티엄 IV 2Ghz/리눅스 머신상에서 실행되었으며 실험을 위한 입력 회로는 반도체 제조회사인 I사에서 제작 중인 회로를 사용하였다. 인터리빙 기법의 효율성을 검증하기 위해 I사 내부적으로 사용하는 CAD 툴을 이용하여 수렴된 배치를 구한 후, 그 배치에 제안된 기법을 적용하여 개선시켰다. 참고로 I사에서 현재 사용하는 툴에 의해 생성된 상세 배치는 더 이상 개선될 여지가 없다고 간주되는 최종 배치이다.

실험에 사용된 회로는 약 20-30%의 여백을 코어 영역에 갖고 있으며, 여백을 고려한 행-기반 인터리빙과 열-기반 인터리빙을 각각 별도로 적용하여 배치가 어떻게 개선되는지를 보임으로써 각 기법의 효과가 어떤지를 보였으며, 또한 열-기반 인터리빙과 행-기반 인터리빙을 번갈아 가면서 적용한 후 개선된 배치의 결과도 보였다. 표 1은 실험에 사용된 각 회로의 사양을 보인다.

표 2에서는 여백을 고려한 행-기반 인터리빙만을 적용했을 때의 결과를 보여준다. 윈도우 크기 즉, W의 값은 10~14 사이의 값이 사용되었다. W의 값이 너무 작으면 인터리빙의 효과가 미미하고, 너무 커지면 속도가 느려질 뿐만 아니라 효과 역시 미미함을 실험을 통해 알았고, 이 사실에 근거하여 윈도우의 크기를 10~14로 두었다.

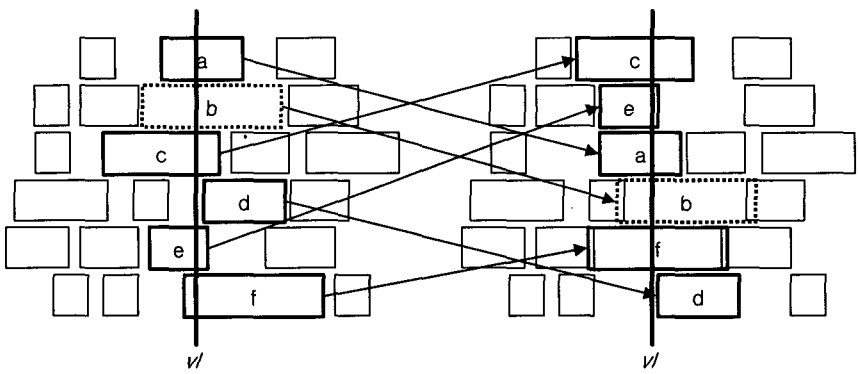


그림 8 열-기반 인터리빙을 수행한 예

표 1 회로 사양

| | #cells | #nets | #rows |
|---|---|---|---|
| ckt1 | 7399 | 7642 | 113 |
| ckt2 | 8065 | 8462 | 56 |
| ckt3 | 11214 | 11375 | 111 |
| ckt4 | 21724 | 21307 | 186 |
| ckt5 | 30071 | 28930 | 191 |
| ckt6 | 50913 | 51048 | 135 |

그림 1에서 보인 행-기반 알고리즘을 첫 행부터 마지막 행까지, 각 행에서는 좌측에서 우측으로 윈도우를 이동하며 적용하였다. 코어 영역에 있는 모든 셀에 대해 인터리빙을 적용하여 배치를 개선한 후 개선율을 평가하였다. 만약 개선율이 크면 더 이상 개선의 여지가 있다고 보고 다시 위 알고리즘을 적용하고 만약 개선율이 0.01%이하가 될 때 즉, 이전의 배치와 개선된 배치의 결과가 0.01%이하일 때 반복을 중단하였다. 표 2에서 보인 개선율은 입력에 사용되었던 배치의 배선길이와 비교하여 최종 배치의 배선길이의 감소율을 나타낸다.

표 3에서는 열-기반 인터리빙만 수행한 결과를 보여주며, 알고리즘 적용 전의 결과는 표 2에서와 동일하다. 열-기반 인터리빙에서 W의 값도 행-기반에서처럼 10~14사이의 값을 가진다. 기준 수직선의 이동 값 즉, 그림 6 단계 5에서 보인 α의 값은 평균 셀 너비를 ψ라 할 때 $2X\psi$로 두었다. 알고리즘을 반복 적용하게 될 때 마다 첫 기준 수직선은 1~ψ에 있는 랜덤 값 τ를 구한 후 x=τ인 지점에 첫 기준 수직선을 두었다. 이렇게 함으로써 반복이 진행될 때 마다 기준 수직선의 값이 동일한 지점에 설정되는 것을 방지할 수 있고, 이는 결국 윈도우의 위치가 매번 다르게 되어 인터리빙에서 제외되는 셀을 최소화할 수 있다. 그림 6에서 보인 알고리즘의 반복회수 제한방법은 행-기반 인터리빙과 달리 CPU사용 시간이 90초를 넘을 경우 더 이상 알고리즘을 적용시키지 않았다. 그렇게 한 이유는 열-기반 알고리즘의 실행 시간은 행-기반 보다 많이 걸렸고, 실험에 근거할 때 90초 이상 적용시킬 경우 개선 정도가 미미하였기 때문이다. 참고로 행-기반 인터리빙을 ckt1에서 1회 실행하는 시간은 1초 미만인데 반해 열-기반 인터리빙의 실행 시간은 약 10초였다.

표 4에서는 행-기반과 열-기반 인터리빙을 번갈아 가면서 적용한 결과를 보였다. 즉, 행-기반 알고리즘을 한 번 적용하여 개선한 배치에 열-기반 알고리즘을 적용하고 이 과정을 반복하였다. CPU 사용시간이 100초를 넘을 경우, 행-기반과 열-기반의 반복 과정을 더 이상 적용하지 않았다. 그러나 이 과정이 일단 시작된 경우엔 CPU 시간이 100초를 초과하더라도 이 반복 과정이 끝날 때 까지 실행하였다. 또한 개선율이 행-기반에서처럼 0.01%미만일 경우에도 실행을 중지시켰다. 이런 이유로 인해 표 4에서 보듯이 비록 제한 시간은 100초로 두었으나 실제 실행시간이 100초 이내가 될 수도 있고, 100초를 초과할 수도 있다.

이렇게 행-기반과 열-기반 인터리빙 두 기법을 동시에 적용하면 셀들이 코어 영역 내에서 임의의 위치로 이동이 가능하게 되고 그만큼 배치가 개선될 가능성이 높다고 볼 수 있고, 실험 결과가 이 사실을 입증한다.

## 5. 결 론

표 2 행-기반 인터리빙만 적용한 결과

| | 알고리즘 적용 전 | 알고리즘 적용 후 | CPU 시간(sec.) | 개선율(%) |
|---|---|---|---|---|
| ckt1 | 4.90502e+07 | 4.66672e+07 | 3 | 4.86 |
| ckt2 | 5.14974e+07 | 5.01667e+07 | 3 | 2.58 |
| ckt3 | 9.08622e+07 | 8.62639e+07 | 34 | 5.06 |
| ckt4 | 1.26529e+08 | 1.20987e+08 | 9 | 4.38 |
| ckt5 | 4.12333e+08 | 3.95490e+08 | 13 | 4.08 |
| ckt6 | 7.65106e+08 | 7.38051e+08 | 10 | 3.54 |
| 평 균 | | | | 4.08 |

표 3 열-기반 인터리빙 만을 적용한 결과

| | 알고리즘 적용 전 | 알고리즘 적용 후 | CPU 시간(sec.) | 개선율(%) |
|---|---|---|---|---|
| ckt1 | 4.90502e+07 | 4.49309e+07 | 90 | 8.34 |
| ckt2 | 5.14974e+07 | 4.82139e+07 | 91 | 6.38 |
| ckt3 | 9.08622e+07 | 8.46881e+07 | 90 | 6.80 |
| ckt4 | 1.26529e+08 | 1.17608e+08 | 94 | 7.05 |
| ckt5 | 4.12333e+08 | 3.97842e+08 | 96 | 3.51 |
| ckt6 | 7.65106e+08 | 7.46141e+08 | 116 | 2.48 |
| 평 균 | | | | 5.77 |

표 4 행-기반 인터리빙과 열-기반 인터리빙을 번갈아 가면서 같이 적용한 결과

| | 알고리즘 적용 전 | 알고리즘 적용 후 | CPU 시간(sec.) | 개선율(%) |
|---|---|---|---|---|
| ckt1 | 4.90502e+07 | 4.29133e+07 | 93 | 12.51 |
| ckt2 | 5.14974e+07 | 4.70688e+07 | 93 | 8.60 |
| ckt3 | 9.08622e+07 | 8.16071e+07 | 95 | 10.19 |
| ckt4 | 1.26529e+08 | 1.12319e+08 | 104 | 11.23 |
| ckt5 | 4.12333e+08 | 3.80018e+08 | 108 | 7.84 |
| ckt6 | 7.65106e+08 | 7.13330e+08 | 143 | 6.77 |
| 평 균 | | | | 9.52 |

본 논문에서는 상세배치를 개선하기 위해 셀의 x-좌표 개선만을 통해 배치를 개선하는 행-기반 최적 인터리빙(optimal interleaving) 알고리즘[29]을 확장하여 셀의 y-좌표 또한 변형할 수 있는 열-기반 인터리빙 기법을 제안하였으며, 또한 셀 사이에 여백이 있는 경우 여백을 더미 셀로 간주하여 처리함으로써 셀의 최적 위치를 효과적으로 찾도록 하였다. 행-기반과 열-기반 인터리빙을 번갈아 적용함으로써 셀이 코어 내에 어디서나 위치할 수 있으며, 셀의 상대적인 순서도 얼마든지 뒤섞일 수 있음을 보였다. 또한 행-기반과 열-기반 인터리빙 기법을 각각 적용하였을 때에도 배치가 개선되었음을 보여주는 결과는 제안한 기법이 상세배치를 개선하는데 매우 효과적이라는 것을 보여준다.

표 2와 표 3에서 보였듯이 행-기반 인터리빙을 통해 평균 4.08%의 배치 개선을 얻었으며 열-기반 인터리빙을 통해 평균 5.77% 배치 개선을 얻었다. 그리고 표 4에서는 두 알고리즘을 번갈아 적용한 결과를 보였고, 평균 9.52% 의 향상된 배치를 구할 수 있었다.

I사 내부에서는 더 이상 개선을 기대하지 않았던 배치를 입력으로 하여 본 논문에서 제안한 기법으로 평균 9.5% 개선했다는 사실은 제안한 기법이 얼마나 우수한지를 실험적으로 보여주는 결과라고 할 수 있다.

실험 결과에서 밝히지는 않았지만 제안된 인터리빙 기법은 회로의 배선 밀집도와 타이밍 즉, 성능을 개선하는 데도 어느 정도 효과가 있음을 알았다. 특히 배선 밀집도의 개선은 x-성분의 배선길이를 줄이는 효과가 배선 밀집도에 미치는 영향이 매우 크다는 것을 알 수 있었다.

제안한 인터리빙의 개념은 2-way 방식의 인터리빙이다. 이를 확장하여 3-way 인터리빙을 적용할 경우 상세 배치가 더 개선될 것으로 추측된다. 또한 지금의 행-기반 인터리빙은 한 행 내에서만 적용 가능하나 인접한 두 행 또는 세 행을 동시에 고려한 인터리빙 기법도 앞으로의 연구를 통해 생각해 볼 여지가 있다.

또한 본 논문에서 제안한 기법을 좀 더 수정하여 타이밍에 매우 영향을 미치는 넷들에 대해 추가적인 제약조건을 두고 이를 중점적으로 개선할 수 있도록 인터리빙 기법을 수정한다면 타이밍을 개선하는 효과가 있을 것으로 판단된다.

## 참 고 문 헌


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

[2] R. Varadarajan, "Convergence of Placement Technology in Physical Synthesis: Is Placement Really a Point Tool?," Proc. of ISPD, pp. 6, 2003.

[3] C. Sechen and A. Sangiovanni-Vincentelli, "TimberWolf3.2: A New Standard Cell Placement and Global Routing Package," Proc. of DAC, pp. 432-439, 1986.

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

[7] X. Yang, M. Wang, K. Egur and M. Sarrafzadeh, "A Snap-on Placement Tool," Proc. of International Symposium on Physical Design, pp. 153-158, 2000.

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

[9] D. J. -H. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective," Proc. of International Symposium on Physical Design, pp. 18-25, 1997.

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

[11] 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.

[12] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Optimal End-Case Partitioners and Placers for

Standard-Cell Layout," Proc. of International Symposium on Physical Design, pp. 90-96, 1999.
[13] 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.
[14] C. M. Fiduccia and R. M. Matteyses, "A Linear Time Heuristic for Improving Network Partitions," Proc. of DAC, pp. 175-181, 1982.
[15] Natarajan Viswanathan, Chris C. Chu, "FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model," IEEE Trans. CAD of Integrated Circuits and Systems, Vol. 24, No. 5, pp. 722-733, 2005.
[16] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," Proc. of DAC, pp. 269-274, 1998.
[17] Karthik Rajagopal, Tal Shaked, Yegna Parasuram, Tung Cao, Amit Chowdhary, Bill Halpin, "Timing Driven Force Directed Placement with Physical Net Constraints," Proc. of ISPD, pp. 60-66, 2003.
[18] Sung-Woo Hur, Tung Cao, Karthik Rajgopal, Yegna Parasuram, Amit Chowdhary, Vladimir Tiourin, and Bill Halpin, "Force Directed Mongrel with Physical Net Constraints," Proc. of DAC, pp. 214-219, 2003.
[19] Adrew B. Kahng and Qinke Wang, "Implementation and Extensibility of an Analytic Placer," Proc. of ISPD, pp. 18-25, 2004.
[20] H. Etawil, S. Arebi, and A. Vannelli, "Attractor-Repeller Approach for Global Placement," Proc. of ICCAD, pp. 20-24, 1999.
[21] Bo Hu, Marek-Sadowska, "FAR: Fixed-Points Addition and Relaxation Based Placement," Proc. ISPD, pp. 161-166, 2002.
[22] Jurgen M. Kleinhans, Georg Sigl, Frank M. Johannes, and Kurt Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," IEEE Transactions on CAD, Volume 10, No. 3, pp. 356-365, 1991.
[23] 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.
[24] P. N. Parakh, R. B. Brown and Karem A. Sakallah, "Congestion Driven Quadratic Placement," Proc. of DAC, pp. 275-278, 1998.
[25] X. Yang, B.-K. Choi, and M. Sarrafzadeh, "Routability Driven White Space Allocation for Fixed-Die Standard-Cell Placement," Proc. of ISPD, pp. 42-47, 2002.
[26] Maogang Wang, X. Yang, Ken Eguro, and M. Sarrafzadeh, "Dragon2000: Placement of Industrial Circuits," Proc. of ICCAD, pp. 260-263, 2000.
[27] 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.
[28] Tony Chan, Jason Cong, Tianming Kong, Joseph R. Shinnerl, "Multilevel Optimization for Large-Scale Circuit Placement," Proc. of ICCAD, pp. 171-176, 2000.
[29] S. Hur and J. Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement," Proc. of ICCAD, pp. 165-170, 2000.
[30] S. Chowdhury, "Analytical Approaches to the Combinatorial Optimization in Linear Placement," IEEE Trans. CAD, Vol. 8, pp. 630-639, 1989.
[31] Y. Saab, "An Improved Linear Placement Algorithm Using Node Compaction," IEEE Trans. on CAD, Vol. 15, No. 8, pp. 952-958, 1996.
[32] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, with Applications to Netlist Clustering," IEEE Trans. On VLSI Systems, Vol. 4, No. 2, pp. 240-246, 1996.
[33] Sung-Woo Hur and John Lillis, "Relaxation and Clustering in a Local Search Framework: Application to Linear Placement," Proc. of DAC, pp. 360-366, 1999.
[34] A.B.Kahng, P.Tucker, and A.Zelikovsky, "Optimization of Linear Placements for Wirelength Minimization with Free Sites," Proc. of ASP-DAC, pp. 241-244, 1999.

오 은 경
1997년 동아대학교 컴퓨터공학과 졸업(학사). 2002년 동아대학교 대학원 컴퓨터공학과(공학 석사). 2005년 동아대학교 대학원 컴퓨터공학과(박사과정 수료). 관심분야는 CAD, 컴퓨터 알고리즘

허 성 우
정보과학회논문지 : 시스템 및 이론
제 33 권 제 2 호 참조