論文95-32A-1-25

메탈-메탈 매트릭스 레이아웃 형태의 기능모듈 생성

(Functional Module Generation in Metal-Metal Matrix(M^3) Layout Style)

車 榮 俊 *, 林 鍾 錫 **

(Young Jun Cha and Chong Suck Rim)

요 약

메탈-메탈 매트릭스(M^3) 레이아웃은 기존의 다른 형태의 레이아웃에 비하여 최소의 폴리선을 사용함으로써 고속의 MOS 논리회로의 구현에 적합하도록 고안된 레이아웃 형태이다[8]. 본 논문에서는 M^3 레이아웃 형태의 기능모듈 생성방법을 새로 제안한다. 제안한 방법에서는 먼저 주어진 회로의 트랜지스터와 입출력 선들을 M^3 레이아웃 형태로 배치하고 이들간에 배선을 수행하여 기능모듈을 생성한다. 이를 위하여 시뮬레이티드 어닐링에 의한 배치방법을 새로 개발하였고 배선에는 채널배선 방법을 변형하여 사용하였다. 실험 결과 기존에 발표된 같은 형태의 생성방법보다 비슷하거나 작은 면적에 기능모듈을 생성할 수 있었고 게이트 매트릭스와 같은 다른 레이아웃 형태와의 비교에서도 비교적 만족스러운 결과를 얻었다.

Abstract

Metal-Metal Matrix(M^3) layout is a recently proposed layout style which uses minimum amount of poly wires for high speed operation[8]. In this paper we propose a method of generating functional modules in M^3 layout style. In the proposed method the transistors and the input/output lines of the given circuit are first placed in M^3 layout style and then they are interconnected using two metal layers. We develop a new placement method by simulated annealing, and we modify the well known channel routing method for the interconnections. When we applied our method to several logic circuits, the area of the generated layout is smaller than the ones by the previously known method. Our results also compares favorably to the other layout styles like gate matrix layout.

* 正會員, 韓國電子通信硏究所
(Electronics & Telecommunication Research Institute. control system section).
** 正會員, 西江大學校 電子計算學科
(Dept. of Computer Science, Sogang Univ.)

接受日字 : 1993年 9月 6日
※ 이 연구는 '91년도 한국과학재단 연구비 지원에 의한 결과임(과제 번호:913-1106-001-2).

I. 서 론

메탈-메탈 매트릭스(M^3) 레이아웃은 MOS 논리회로를 구현하기 위하여 Kang[8] 이 처음으로 제안한 레이아웃 형태이다. M^3 레이아웃에서는 두 메탈층의 선들을 각각 수직 수평으로 놓여지게 하여 회로의 입출력, 전원 그리고 트랜지스터간의 연결 등에 사용하고, 폴리층은 MOS 트랜지스터의 형성을 위해서만 사용하도록 한다.

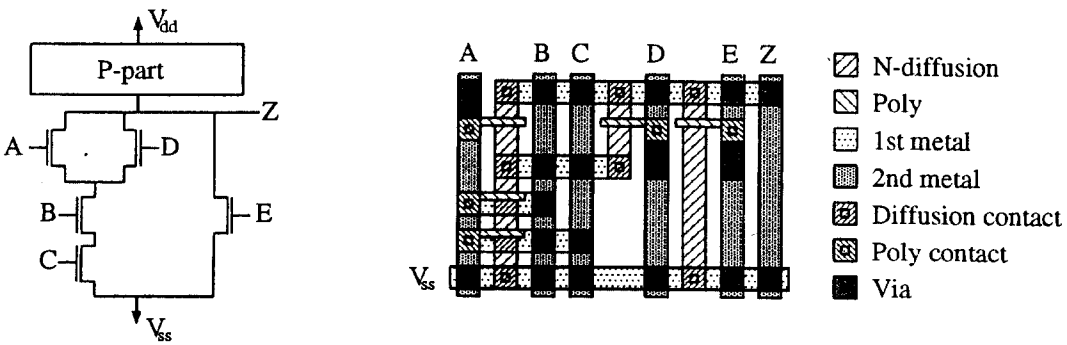
그림 1(b)에 그림 1(a)에 보인 CMOS 논리회로의 N-part를 M^3 레이아웃 형태로 구현한 예를 심볼릭 형태로 보인다. 그림에서 보인 바와 같이 트랜지스터는 메탈선간에 디퓨션 블럭을 매트릭스 형태로 배열하고 그 위에 폴리선을 지나가도록 하여 만들어진다. 각 트랜지스터는 첫째 메탈층의 수평선들을 사용하여 출력, 전원 또는 다른 트랜지스터에 연결된다.

M^3 레이아웃 형태가 제안된 주된 이유로서는 첫째, 대부분의 연결에 메탈층의 선들을 사용함으로써 가능한 적은 양의 폴리선을 사용하여 구현하고자 하는 기능모듈의 속도를 빠르게 할 수 있고, 둘째, P-part와 N-part의 폴리선을 분리할 수 있기 때문에 각 part에 서로 다른 폴리 공정기술을 사용하여 레이아웃의 전기적 특성을 보다 최적화 시킬 수 있다는 점이다.[8] 또한 레이아웃의 입출력 터미날이 메탈층에 존재하기 때문에 요즘 널리 사용되고 있는 더블 메탈 공정기술에 보다 적합하다.

이러한 장점을 가진 M^3 레이아웃은 고속이 요구되는 응용에 필요한 기능모듈 구현에 적당하다고 여겨지나 그 면적을 최소화 하는 것은 유사한 형태의 게이트 매트릭스 레이아웃[10]에 비하여 보다 어렵다고 볼 수 있다. 게이트 매트릭스 레이아웃에서는 수직 폴리선의 갯수가 트랜지스터 게이트로 연결되는 서로 다른 신호선과 출력선으로 고정되어 있기 때문에 주어진 논리회로에 대한 레이아웃의 폭은 일정하다.[14] 따라서, 게이트 매트릭스 레이아웃 형태에서 그 면적을 최소화하는 것은 게이트 매트릭스의 높이를 작게하는 것과 동등하다.[5,14]

반면에 M^3 레이아웃에서는 게이트 매트릭스 레이아웃에서의 수직 폴리선이 둘째 메탈층의 수직선으로 대체되어 트랜지스터가 임의의 위치에 놓일 수 있기 때문에 게이트 매트릭스 레이아웃에서처럼 그의 수평폭이 고정되어 있지 않다. 즉, 하나의 열에 다수의 트랜지스터를 배치하여 이들을 위한 열의 갯수를 작게 할 수 있다. 그러므로 M^3 레이아웃에서는 열과 행의 갯수를 적당히 조정하여 전체 면적의 최소화를 시도하여야 하며 따라서, 게이트 매트릭스 레이아웃 형태의 면적 최소화 문제를 1 차원 문제라고 한다면 M^3 레이아웃 형태의 면적 최소화 문제는 2 차원 문제라고 할 수 있다.

다시 말하면 M^3 레이아웃에서 그 면적을 최소화하기 위해서는 레이아웃의 열과 행의 갯수를 조정하여 그 곱이 최소가 되도록 하여야 한다. 그러나 이들의 갯수를 동시에 조정하는 것은 대단히 어려워 보통 열의 갯수를 먼저 최소화하고 다음 행의 갯수를 최소화하는 방법을 택하고 있으며[1,3], 이러한 방법들은 레이아웃의 수평폭과 수직폭을 독립적으로 최소화 하는데서 오는 전체 면적의 증가를 받아들여야 했다. 본 논문에서는 이러한 단점을 해결하기 위하여 레이아웃의 수직폭



(a) CMOS 논리회로의 예
(a) A CMOS logic circuit.

(b) (a)의 N-part에 대한 M^3 레이아웃
(b) An M^3 layout for the N-part of (a).

그림 1. CMOS 논리회로와 그의 M^3 레이아웃 형태로의 구현
Fig. 1. A CMOS logic circuit and its M^3 layout.

과 수평폭을 동시에 고려하여 전체 면적을 최소화하는 새로운 기능모듈 생성방법을 제안한다.

우리의 기능모듈 생성과정은 배치와 배선 두 단계로 구성된다. 먼저 SPICE 입력 형태의 논리회로가 주어지면 회로의 트랜지스터들을 생성하기 위하여 한 개 이상의 디퓨션 블럭들로 구성된 수직 디퓨션 열과 입/출력 그리고 P 트랜지스터와 N 트랜지스터 간의 연결을 위한 둘째 메탈층의 수직선 등을 생성하고 이들의 상호 위치를 결정한다. 이러한 과정을 편의상 배치라고 부르며 본 논문에서는 이를 위하여 새로 개발한 시뮬레이티드 어닐링[9,12]에 의한 배치방법을 제안한다.

수직 디퓨션 열과 둘째 메탈층의 수직선들의 배치가 완료된 후에는 채널배선 기법[4,7]을 M^3 레이아웃 형태에 맞게 변형시킨 배선 방법을 통하여 배선을 수행한다. 이러한 배선결과로서 가상 그리드[12] 형태의 심볼릭 레이아웃이 만들어지며 여기에 레이아웃의 면적을 보다 작게하기 위한 후처리 과정을 거친후 CIF 형태의 실제 레이아웃 데이타가 생성된다.

서론에 이어 본 논문은 다음과 같이 구성되어 있다. 먼저 다음 2 장에서는 시뮬레이티드 어닐링을 이용한 배치방법에 대하여 설명하고 3 장에서는 배선에 사용되는 변형된 채널배선 방법과 후처리 과정에 대해 기술한다. 그리고 4 장에서는 본 논문에서 제안한 기능모듈 생성방법을 통해 얻은 실험 결과를 보이고 이를 기존의 방법을 통해 얻은 결과와 비교하며 마지막으로 5장에서는 결론을 낸다.

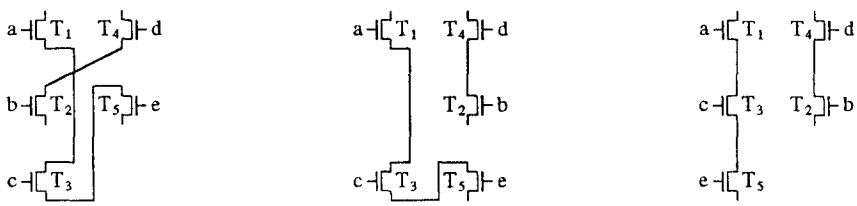
Ⅱ. 배 치

주어진 논리회로에 대한 기능모듈을 M^3 레이아웃 형태로 구현할 때 그 모듈의 폭은 트랜지스터를 형성하기 위한 수직 디퓨션 열과 둘째 메탈층의 수직선들에 의하여 영향을 받는다. 여기서 둘째 메탈층의 수직선들은 모듈의 입출력 그리고 중간출력 등을 위하여 사용되어 그 갯수가 고정되어 있으나 수직 디퓨션 열의 갯수는 각 디퓨션 블럭에서 형성될 트랜지스터를 결정하므로써 임의로 조정이 가능하다. 따라서, 수직선의 갯수가 고정된 게이트 매트릭스 레이아웃 형태와는 달리[14] M^3 레이아웃에서는 수직 디퓨션 열의 갯수를 적당히 조절하면서 이들과 둘째 메탈층의 수직선들과의 상호위치를 동시에 결정하여 모듈의 면적이 최소가 되도록 하여야 한다. 본 장에서는 이러한 목적을 위하여 개발한 시뮬레이티드 어닐링에 의한 배치과정을 소개한다.

시뮬레이티드 어닐링에 의한 배치를 수행하기 위하여 먼저 초기배치를 구한다. 구현하고자 하는 기능모듈의 트랜지스터 회로가 주어지면 이로부터 모든 입력, 출력 그리고 중간출력에 대하여 각각 둘째 메탈층의 수직선을 하나씩 생성하고 이들을 수평으로 왼쪽에서 오른쪽으로 배열한다.

또한, 회로의 P-part와 N-part를 그래프로 변환하여 트랜지스터 형성에 필요한 디퓨션 블럭을 결정한다[15]. 즉, 회로의 P-part에서 두 개 이상의 트랜지스터가 서로 연결된 점과 전원(Vdd), 출력, 그리고 중간출력 등에 대하여 각각 정점을 생성하고, 각 트랜지스터에 대하여 그의 source에 대응되는 정점으로부터 drain에 대응되는 정점으로 이어지는 방향에지(directed edge)를 생성한다. 이렇게 생성된 그래프로부터 이의 모든 에지를 포함하는 경로들(edge disjoint paths)을 구하고 각 경로에 대하여 하나의 디퓨션 블럭을 생성하여 그 경로상에 포함된 트랜지스터를 생성한다.

이와 유사한 과정을 회로의 N-part에 대해서도 반복하며 이렇게 구한 디퓨션 블럭들을 하나씩 수직 디퓨션 열로 하여 이미 배열한 둘째 메탈층의 수직선 사



(a) 현재의 배치 (b) T_2의 이동 (c) T_5의 이동
(a) Current placement. (b) Relocation of T_2. (c) Relocation of T_5.

그림 2. 이동 M_1에 대한 예
Fig. 2. Illustrations for M_1.

이에 배치한다[15]. 그런데 초기배치 과정에서 얻은 이러한 수직 디퓨션 열들은 고정된 것이 아니며 차후 시뮬레이티드 어닐링이 진행됨에 따라 이들의 갯수와 각 열에 있는 디퓨션 블럭 그리고 생성될 트랜지스터에 변화가 있게 된다.

시뮬레이티드 어닐링에 의하여 어떤 문제를 해결하기 위해서는 하나의 해(feasible solution)로부터 다른 해를 구하기 위한 이동 방법, 비용 함수, 냉각 스케쥴 그리고 중단 조건 등을 개발하여야 한다. 본 장의 나머지 부분에서는 우리의 배치 문제를 해결하기 위하여 개발한 이러한 사항들을 세부적으로 설명한다.

1. 이동 방법

시뮬레이티드 어닐링 기법을 적용하기 위하여 필요한 사항 중의 하나는 현재의 해(배치)로 부터 이웃해(새로운 배치)를 얻는 방법이다. 이를 이동 방법이라고 하며 이를 통하여 시뮬레이티드 어닐링 기법에서는 임의의 새로운 배치를 반복적으로 구하여 배치를 점차적으로 개선시킨다. 그런데 M^3 레이아웃에서는 새로운 배치를 얻기 위한 변화 대상으로 둘째 메탈층의 수직선과 수직 디퓨션 열 등 두 가지가 있고, 특히 수직 디퓨션 열에 생성할 트랜지스터들을 조정할 수 있으므로 현재의 배치 상태로부터 여러 다양한 새로운 배치를 얻을 수 있다. 또한, 수직 디퓨션 열에는 둘째 메탈층의 수직선과는 달리 트랜지스터에 관한 정보를 포함하고 있으므로 보다 효율적으로 이동에 의한 비용의 증감을 계산할 수 있어야 한다. 이러한 점을 고려하여 우리는 다음과 같은 10 가지 이동 방법을 고안하였다.

이동 M_1 : 트랜지스터의 이동.

이동 M_2 : 두 트랜지스터의 교환.

이동 M_3 : 수직 디퓨션 열의 생성과 소멸.

이동 M_4 : 둘째 메탈 수직선의 이동.

이동 M_5 : 두 둘째 메탈 수직선의 교환.

이동 M_6 : 수직 디퓨션 열의 이동.

이동 M_7 : 두 수직 디퓨션 열의 교환.

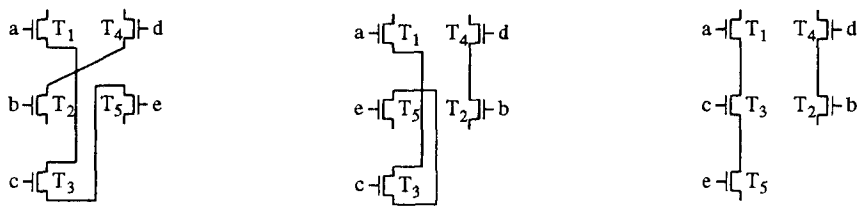
이동 M_8 : 수직 디퓨션 열과 둘째 메탈 수직선의 교환.

이동 M_9 : 트랜지스터의 뒤집기.

이동 M_{10} : 트랜지스터의 동적 선택.

이동 M_1은 회로를 구성하는 한개의 트랜지스터를 임의로 선택하여 이것을 현재와 다른 임의의 수직 디퓨션 열의 임의의 위치에 할당하므로써 새로운 배치를 얻는 이동 방법이다. 그림 2에 이러한 이동 방법에 대한 예를 보인다. 여기서 임의의 수직 디퓨션 열의 각 디퓨션 블럭에서 생성될 트랜지스터들은 직렬로 연결되어 있으며 따라서 그림에서 각 디퓨션 블럭을 직렬 연결 트랜지스터들로 표시한다. 즉, 앞으로 이 절의 모든 그림에서 같은 수직 디퓨션 열의 직렬 연결 트랜지스터들은 하나의 디퓨션 블럭을 나타낸다. 또한, 그림 2에서 보인 바와 같이 하나의 수직 디퓨션 열에 두 개 이상의 디퓨션 블럭이 존재할 수도 있다. 이동 M_1에 대한 그림 2의 예를 설명하면, 그림 2(a)에 보인 현재해에 대하여 트랜지스터 T_2를 다른 디퓨션 블록으로 이동하므로써 그림 2(b)에 보인 바와 같이 두 열간의 연결선이 적은 보다 개선된 배치를 얻을 수 있다. 나아가서 그림 2(c)에 보인 바와 같이 T_5를 다시 이동하여 더욱 개선된 배치를 얻을 수 있다.

이동 M_2는 임의의 두개의 트랜지스터를 선택하여 서로의 위치를 바꿈으로써 새로운 배치를 얻는 이동 방법이다. 그림 3에 이러한 이동 방법에 대한 예를 보인다.

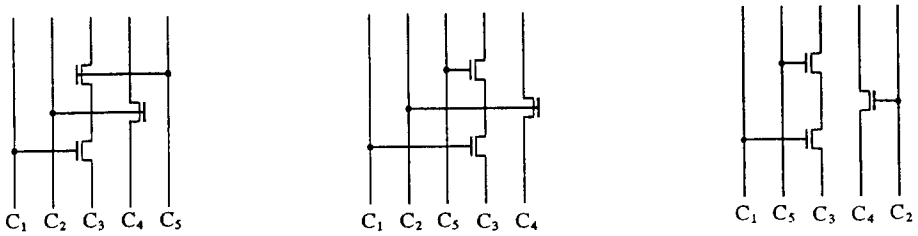


(a) 현재의 배치. (b) T_2와 T_5의 교환. (c) T_3와 T_5의 교환.

(a) Current placement. (b) Exchange of T_2 and T_5. (c) Exchange of T_3 and T_5.

그림 3. 이동 M_2에 대한 예

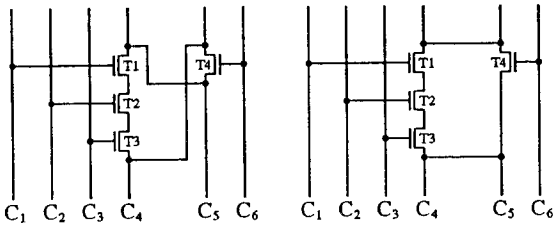
Fig. 3. Illustrations for M_2.



(a) 현재의 해 (b) C_5를 이동 (c) C_2를 이동(C_2와 C_5의 교환)
(a) Current placement. (b) Relocation of C_5. (c) Relocation of C_2(Exchange of C_2 and C_5).

그림 4. 이동 M_4와 M_5에 대한 예
Fig. 4. Illustrations for M_4 and M_5.

그림에서 보인 바와 같이 현재 해로부터 먼저 트랜지스터 T_2와 T_5를 교환하고 다시 T_3와 T_5를 교환하므로서 두 수직 디퓨션 열간의 연결선이 적은 배치를 얻을 수 있다. 이동 M_3는 수직 디퓨션 열을 새로 만들거나 없애므로서 새로운 배치를 얻는 이동 방법이다. 새로 하나의 수직 디퓨션 열을 만들 경우 다른 수직 디퓨션 열에 있는 트랜지스터들을 임의로 선택하여 새로 만든 열로 이동시킨다. 이때 선택되는 트랜지스터의 갯수는 임의로 정하되 다른 수직 디퓨션 열에 있는 트랜지스터들의 평균 갯수보다 적게하여 레이아웃의 높이가 커지지 않도록 한다.



(a) 현재의 배치 (b) T_4를 뒤집은 결과
(a) Current placement. (b) The result by flipping T_4.

그림 5. 이동 M_9에 대한 예
Fig. 5. An illustration for M_9.

한편, 기존의 수직 디퓨션 열을 제거하는경우는 그 열에 있는 트랜지스터들을 다른 수직 디퓨션 열로 이동시켜 새로운 배치를 얻는 이동 방법이다. 이때 트랜지스터들이 이동할 수직 디퓨션 열은 임의로 선택한다.

이동 M_4는 둘째 메탈층의 수직선을 수평으로 이동시켜 새로운 배치를 얻는 이동 방법이다. 이와 유사하게 이동 M_5는 서로 다른 2 개의 둘째 메탈층의 수직선을 교환하는 것이다. 그림 4에 이러한 이동의 예를 보인다. 그림의 예에서 실제로 C_5와 C_2를 차례로 이동한 결과는 C_5와 C_2를 한번 교환한 결과와 동일하다. 이동 M_4, M_5와 유사하게 이동 M_6는 임의로 선택한 하나의 수직 디퓨션 열을 이동시켜 새로운 배치를 얻는 이동 방법이고 이동 M_7은 서로 다른 두개의 수직 디퓨션 블럭을 교환하여 새로운 배치를 얻는 방법이다. 그리고 이동 M_8은 둘째 메탈층의 수직선 하나와 수직 디퓨션 열 하나를 임의로 선택하여 이 둘을 교환하여 새로운 배치를 얻는 이동 방법이다.

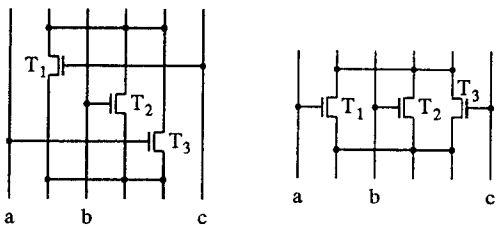
이동 M_4, M_5, M_6, M_7, M_8 들은 실제로 수직 디퓨션 열과 둘째 메탈층의 수직선들을 구분하지 않고 임의의 두 수직선을 선택하여 교환하거나, 하나의 수직선을 다른 위치로 옮기는 등의 두가지 이동 방법에 속한다고 할 수 있다. 그러나 수직 디퓨션 열과 둘째 메탈층의 수직선의 자료구조가 서로 상이하여 이들을 구분하지 않는 단순한 이동방법을 사용할 경우 다음 II장.2 절에서 설명할 비용의 효율적인 계산에 장애가 되므로 이들을 따로 분리하여 이동 방법을 적용하므로써 보다 효율적으로 비용의 증감을 계산할 수 있도록 하였다.

이동 M_9는 임의의 수직 디퓨션 열을 선택하고 이곳에 놓여있는 임의의 갯수의 직렬 연결 트랜지스터를 골라 이들이 놓여진 순서를 뒤집어 새로운 배치를 얻

는 이동 방법이다.

이러한 과정에서 선택된 수직 디퓨션 블럭은 트랜지스터들의 직렬 연결 상태가 끊어져 두 개의 블럭으로 나뉘어지거나 또는 같은 열의 다른 수직 디퓨션 블럭의 트랜지스터들과 직렬연결 상태로 되어 하나의 디퓨션 블럭으로 합쳐질 수도 있다. 그림 5에 이러한 이동 방법에 대한 예를 보인다.

마지막으로 이동 M_{10}은 직렬 또는 병렬 연결된 트랜지스터끼리는 그 게이트 신호를 서로 바꾸어도 회로의 논리에 아무런 영향을 미치지 않는다는 사실을 이용하여 새로운 배치를 얻는 이동 방법이다. 이동 M_{10}에 의한 새로운 배치는 이전 배치에 비하여 두 트랜지스터의 게이트 신호명만 바뀔 뿐이지만 열간의 연결선 또는 레이아웃의 높이에 영향을 미칠 수 있다. 예를들어 그림 6(a)에 보인 배치는 연결선이 대단히 복잡하다.



(a) 현재의 배치 (b) T_1과 T_3의 게이트 연결선을 교환한 결과

(a) Current Placement (b) The result by placement. exchanging gate connections of T_1 and T_3.

그림 6. 이동 M_{10}에 대한 예
Fig. 6. An illustration for M_{10}.

그러나 트랜지스터 T_1, T_2, T_3가 병렬 연결되었으므로 T_1과 T_3의 게이트 신호를 서로 바꿀 수 있고 이를 수행하면 그림 6(b)에 보인 바와 같이 개선된 배치를 얻을 수 있다.

2. 비용 함수

시뮬레이티드 어닐링 기법을 적용하기 위하여 필요한 비용 함수는 M^3 레이아웃 회로의 형태에 알맞게 선택되어져야 한다. 그런데 M^3 레이아웃의 면적을 작게 하기 위해서는 레이아웃의 폭과 높이를 동시에 고려하여야 하고 이를 위하여 수직 디퓨션 열의 갯수, 수직 디퓨션 블럭에 놓여질 트랜지스터들의 상호 위치, 열간의 연결을 위하여 필요한 수평 트랙의 갯수 등 많은 사항을 비용 함수에 반영하여야 한다. 따라서, 우리는 다음과 같은 5 가지의 비용을 정의 하였다.

비용 C_1 : 레이아웃 면적에 대한 예측값.
비용 C_2 : 디퓨션 abutment 의 수(음수 값).
비용 C_3 : 연결선의 길이의 합.
비용 C_4 : 연결선의 형태에 따른 실제적인 레이아웃의 높이.
비용 C_5 : VDD(GND)를 위한 수평선의 같은 행 배치.

이러한 다섯 가지의 비용에 각각의 가중치를 곱하여 더한 것이 전체 비용 함수가 된다. 즉, 각각의 가중치를 W_1, W_2, W_3, W_4, W_5 라고 하면 전체 비용 함수는 다음과 같다.

$$\text{전체 비용} = W_1C_1 + W_2C_2 + W_3C_3 + W_4C_4 + W_5C_5$$

앞으로 이러한 각각의 비용에 대하여 설명을 한다.

비용 C_1은 생성할 레이아웃의 전체 면적을 예측한 값으로 현재 배치 상태의 폭과 높이의 곱으로 다음과 같이 나타내어 진다.

$$C_1 = (\text{현재 레이아웃의 폭}) \times (\text{현재 레이아웃의 높이}).$$

여기서 레이아웃의 폭은 둘째 메탈층의 수직선들과 수직 디퓨션 열의 갯수의 합으로 정의한다. 그런데 만약 이들의 갯수가 레이아웃의 P-part와 N-part에서 서로 다르다면 이중 큰 값을 택하여 레이아웃의 폭으로 결정한다. 그리고 레이아웃의 높이는 회로의 네트리스트를 이용하여 레이아웃의 열간의 연결에 필요한 첫째 메탈층의 수평선들을 대략 구하고 이들로부터 P-part와 N-part의 최대밀도(density)를 각각 구하여 이 두 값을 더한 값으로 정의한다.

현재의 배치 상태에 존재하는 모든 디퓨션 블럭의 집합을 $\{B_1, B_2, \ldots, B_b\}$라고 하고 B_i에서 만들어질 트랜지스터의 갯수를 f_i라고 하자. 그러면 비용 C_2는 다음과 같이 정의된다.

$$C_2 = -\sum_{i=1}^{b}(f_i - 1)$$

하나의 수직 디퓨션 열에는 두 개 이상의 디퓨션 블럭이 놓일 수 있는데 이 경우 실제 레이아웃을 만드는데 있어서 상하로 인접한 이러한 두 블럭을 배치할 때 설계규칙에서 정의한 만큼의 간격을 두어야 한다. 따라

서 하나의 수직 디퓨션 열에 서로 다른 디퓨션 블럭이 많이 존재할수록 레이아웃의 높이는 커지는 경향이 있다. 비용 C_2는 이러한 간격 없이 배치되는 트랜지스터 쌍의 전체 갯수로서 이 값이 작을수록 구현할 모듈의 높이를 작게할 뿐 아니라 차후 배선을 쉽게하여주는 효과를 준다.

비용 C_3는 다음과 같이 정의된다.

C_3 = 각 네트에 대한 bounding box의 half perimeter의 합.

즉, 비용 C_3는 현재의 배치에서 각 네트의 배선에 사용하는 선의 길이를 그 네트의 bounding box의 half perimeter로 간주하고 이들의 합을 구하여 개략적인 전체 배선길이로 예측한 값이다.

비용 C_4는 연결선의 형태를 고려하여 구현될 기능모듈의 높이를 좀더 구체적으로 예측한 값이다. 예를들어 그림 7에서 보인 바와 같은 배치가 주어졌다고 가정하자. 만약 두 개의 수직 디퓨션 열간의 연결을 둘째 메탈층의 수직선을 추가로 사용하거나 배선을 꺾어서 연결 한다면 레이아웃의 폭이 넓어지고 다른 네트의 배선을 어렵게 하는 경향이 있어 레이아웃의 면적은 그리 작아지지 않는다. 따라서 우리의 방법에서는 이러한 연결에 주로 한 개의 첫째 메탈층의 수평선을 사용하는데, 이 경우 두 열에서의 연결점의 수직 높이의 차이로 인하여 전체 모듈의 높이가 커지게 된다. 즉, 그림의 경우에 레이아웃의 높이는 6 개의 트랜지스터를 하나의 열에 수직으로 배치한 높이와 같다.

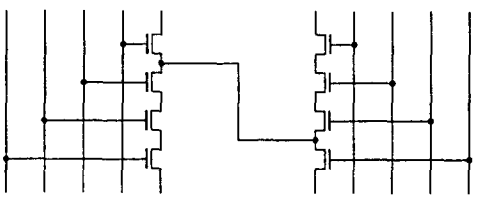


그림 7. 비용 C_4에 대한 설명.
Fig. 7. An illustration for cost C_4.

네트 n에 의하여 서로 연결 되어야 할 수직 디퓨션 열들을 D_1, D_2, ..., D_k라고 하고 열 D_i에서의 연결점 위에 존재하는 트랜지스터의 갯수를 f_i, 아래에 존재하는 트랜지스터의 갯수를 g_i라고 하자. 만약 연결선을 고려하지 않고 각 수직 디퓨션 열에 존재하는 트랜지스터의 갯수로만 레이아웃의 높이로 간주하면 $\max_{1 \le i \le k}\{f_i + g_i\}$ 가 현재 고려하고 있는 수직 디퓨션 열에 의한 레이아웃의 높이로 볼 수 있다. 그러나 네트 n의 배선을 위한 수평 연결에 의하여 레이아웃의 높이는 $\max_{1 \le i \le k}\{f_i\} + \max_{1 \le i \le k}\{g_i\}$ 라고 예측하는 것이 보다 정확하다. 그런데 네트의 연결을 고려한 예측값이 연결을 고려하지 않은 예측값보다 항상 더 크므로 이 두 값의 차이를 가능한 적게함으로써 레이아웃의 실제 높이를 정확히 예측할 수 있도록 할 필요가 있는데 이를 위하여 다음과 같은 비용 C_4를 전체 비용에 추가한다.

$$C_4 = \sum_{n \in \text{전체네트}} (\max_{1 \le i \le k}\{f_i\} + \max_{1 \le i \le k}\{g_i\} - \max_{1 \le i \le k}\{f_i + g_i\})$$

비용 C_5는 전원과의 배선을 쉽게하기 위한 비용이다. 본 논문에서 고려한 M^2 레이아웃에서는 Vdd를 위한 첫째 메탈층의 수평선은 P-part의 가장 위쪽에 그리고 Gnd를 위한 첫째 메탈층의 수평선은 N-part의 가장 아래에 배치되도록 한다. 따라서 모든 트랜지스터의 전원과의 연결점은 P 트랜지스터의 경우 그것이 존재하는 수직 디퓨션 열의 가장 위에 배치되어야 하며 N 트랜지스터의 경우 그것이 존재하는 수직 디퓨션 열의 가장 아래에 배치되어야 한다. 따라서, 이 조건을 만족하지 않는 모든 트랜지스터의 갯수를 비용 C_5로 하여 시뮬레이티드 어닐링의 수행시 전원과 연결되어야 할 모든 트랜지스터가 전원에 직접 연결될 수 있도록 한다.

이렇게 정의한 비용들은 이동을 수행한 후마다 신속히 수정할 수 있도록 앞의 2.1 절에서 언급한 바와 같이 이동 방법을 세분화하는 등의 기법을 사용하여 프로그램을 구현하였는데 이에 대한 자세한 설명은 지면상 생략한다. 마지막으로 전체 비용을 구하기 위한 가중치는 실험에 의하여 결정하였다. 실험 결과 5 개의 가중치가 다음과 같은 관계가 있을 때 좋은 결과를 내었으며 실제로 우리가 사용한 가중치는 $W_1 = 10$, $W_2 = 1$, $W_3 = 1$, $W_4 = 10$, $W_5 = 10000$ 이다.

$W_5 \gg (W_1 \approx W_4) > (W_2 \approx W_3)$

3. 냉각 스케줄과 중단 조건

시뮬레이티드 어닐링 방법에 의한 배치를 성공적으로 수행하기 위해서는 온도를 적절하게 낮추기 위한 냉각 스케쥴과 각 온도에서 평형상태에 도달하였음을 판단하거나 시뮬레이티드 어닐링을 언제 끝낼것인가를 판단하기 위한 중단조건 등을 결정하여야 한다. 기본적으로 우리의 배치방법에서는 초기 온도를 이 온도에서 약 90% 정도의 이웃해를 새로운 해로 받아들이도록 설정한다. 그리고 한 온도 T에서 평형상태에 도달하면

다음 온도 T_{new}를 다음과 같은 식에 의하여 결정한다.

$$T_{new} = \alpha \times T$$

여기서, α는 0.9로 한다.

앞의 II장.1 절에서 정의한 이동 방법의 발생 빈도를 조정하기 위하여 각 이동 M_i에 대해 변수 p_i를 두고 $(0.0 \le p_i \le 1.0)$ M_i의 발생확률을 $p_i / \sum_{i=1}^{10} p_i$로 정한다. 그런데 각 이동 방법의 발생 확률은 시뮬레이티드 어닐링이 진행됨에 따라 변수 p_i, $1 \le i \le 10$, 값을 조정하여 수시로 달라지도록 한다. 초기에는 이동 M_3 와 M_{10}을 제외한 모든 다른 이동 방법의 발생 확률을 같게 설정하고(즉, $p_i = 1.0$, $i = \backslash\ 3, 10$) M_3와 M_{10}의 발생 확률은 이들의 반으로 정한다(즉, $p_3 = p_{10} = 0.5$). 이와 같이 이동 M_3 와 M_{10}의 발생 확률을 다른 이동 방법에 비하여 낮게 설정하는 이유는 실험에 의하여 관찰한 결과, 보다 빠른 시간에 전원선의 위치를 우리가 원하는 곳(즉, Vdd는 레이아웃의 위에, 그리고 Gnd는 레이아웃의 아래에)으로 배치할 수 있기 때문이다.

시뮬레이티드 어닐링의 수행중에 각 이동 방법의 발생 확률은 다음과 같이 수시로 조정된다. 만약 현재 이동 M_i를 시도 하였고 이 시도가 받아들여 졌다면 $p_i = p_i + 0.05$로 조정하여(만약 그 결과가 1.0보다 클 경우에는 $p_i = 1.0$으로 한다) 좀 더 많은 시도를 유도한다. 그런데 만약 이 시도가 받아들여지지 않았다면 $p_i = p_i - 0.05$로 조정하여(만약 그 결과가 0.0보다 작을 경우에는 $p_i = 0.0$으로 한다) 이 이동 방법의 시도를 억제한다. 이러한 확률조정을 계속하되 현재 온도가 변수 p_i 값이 초기화 된 온도의 1/5이 되면 p_i 값을 다시 위에서 설명한 방법으로 초기화 시킨후 시뮬레이티드 어닐링의 수행을 반복한다.

한 온도에서 평형 상태에 도달하였는지를 판단하는 기준은 다음과 같다. N_t를 입력으로 주어진 논리회로의 트랜지스터의 총 갯수라고 할 때 $M_{v=20N_t}$, $A_{c=10N_t}$ 라고 하자. 만약 어떤 온도 T에서 A_c 만큼의 해가 새로운 해로 받아들여졌거나 혹은 M_v 만큼의 이동이 이루어졌다면 평형 상태에 도달하였다고 간주한다. 즉, 이러한 조건을 만족하면 온도를 낮추고 다시 시뮬레이티드 어닐링을 계속한다.

시뮬레이티드 어닐링의 전체 과정은 온도 T가 0.1에 도달하였을 때 끝내도록 한다. 그런데 이미 최적의 해에 거의 도달하여 이동에 따른 이웃해를 받아들이기가 힘들 때 무의미하게 반복되는 이동을 줄이기 위해서 다음과 같은 가속방법을 사용하여 수행시간의 단축을 유도한다. 어떤 온도에서 M_v 만큼의 이동 후에도 받아들여진 이동의 횟수가 $A_c/4$ 이하일 때 $M_v = 9M_v/10$ 그리고 $A_c = 9A_c/10$으로 낮추어 다음 온도부터는 최대 이동 시도 횟수를 작게한다.

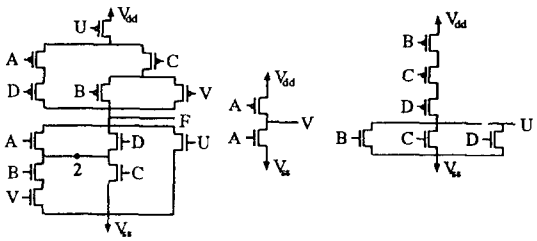
마지막으로, 온도가 어느 정도까지 충분히 내려가면 그 때부터는 중간 배치 결과를 전체 비용이 감소할 때마다 저장해 놓는다. 이것은 나중에 시뮬레이티드 어닐링이 끝났을 때 지금까지의 배치결과중에서 최소의 비용을 갖는 배치를 선택하기 위함이다.

III. 배선 및 레이아웃 생성

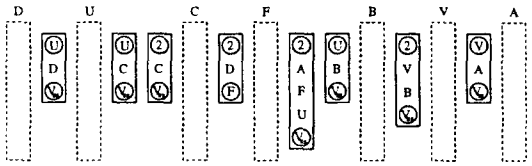
앞의 2 장에서 설명한 방법에 의하여 배치를 수행한 후 남은 일은 수직 디퓨션 열의 필요한 위치에 폴리선을 추가하여 트랜지스터를 만들고 첫째 메탈층의 수평선을 사용하여 필요한 배선을 수행함으로써 완전한 레이아웃을 생성하는 것이다. 이를 위하여 기존의 채널배선 방법[4,7]을 수정 적용하여 심볼릭 레이아웃을 구성하고 이를 바탕으로 CIF 형태의 레이아웃 결과를 생성하는데 본 장에서는 이러한 과정을 자세히 설명한다.

그림 8(b)에 그림 8(a)에 보인 논리회로의 N-part를 우리의 배치방법에 의하여 배치한 결과를 보인다. 그림에서 점선의 수직 상자는 둘째 메탈층의 수직선을 나타내며 실선의 수직 상자는 수직 디퓨션 블럭을 의미한다. 실선의 수직 상자안의 원들은 다른 곳과의 연결이 필요한 트랜지스터의 드레인 또는 소스를 나타내며 그 내부의 라벨(label)은 이 연결점이 포함된 네트의 이름이다. 또한 수직 상자안의 원으로 둘러싸이지 않은 라벨은 그 위치에 트랜지스터를 위한 게이트가 있어야 함을 의미하며 이 게이트는 같은 라벨을 갖는 다른 연결점과 연결된다. 따라서 하나의 네트는 서로 연결이 필요한 둘째 메탈층의 수직선, 트랜지스터의 드레인 또는 소스를 위한 연결점 그리고 트랜지스터의 게이트 등으로 구성된 집합이다. 여기서 편의상 네트를 구성하는 각 원소를 터미날이라고 부른다.

주어진 배치에 대한 배선을 수행하는데 있어서 각 수직 디퓨션 열에서는 터미날간의 위치에 제한이 존재한다. N-part의 경우 어떤 트랜지스터의 드레인을 연결하기 위한 배선은 게이트를 위한 폴리선보다 위에 놓여져야 하며 소스를 연결하기 위한 배선은 같은 폴리선의 아래에 놓여져야 한다(P-part의 경우에는 이와 반대이다). 즉, 하나의 수직 디퓨션 열에 있는 터미날들의 배선은 배치결과에서 주어진 상하 위치대로 구현되어야 한다.



(a) CMOS 논리회로[2]
(a) A CMOS logic circuit[2].



(b) (a)에 보인 회로의 N-part에 대한 배치 결과
(b) The placement result for the N-part of the circuit in (a).

그림 8. CMOS 논리회로와 그의 배치결과
Fig. 8. A CMOS logic circuit and its placement result.

이러한 수직제한 조건을 나타내기 위하여 배치결과가 w 개의 열로 구성되어 있다면 N-part와 P-part 각각에 대한 배치상태를 각 열에 대하여 왼쪽에서 오른쪽으로 w 개의 링크드 리스트(linked list) L_1, L_2, ..., L_w를 구성한다. 만약 i 번째 열이 둘째 메탈층의 수직선이라면 L_i는 이를 대표하는 터미날 하나를 그의 원소로 가지고 있다. 한편 i 번째 열이 수직 디퓨션 열이라면 L_i는 N-part의 경우 Vss로부터(P-part의 경우 Vdd로부터) 먼데서 가까운 쪽으로 수직 디퓨션 블럭에 존재하는 터미날을 그 순서대로 원소로 하는 링크드 리스트이다.

본 논문에서 정의한 심볼릭 레이아웃 S는 정사각형의 타일(tile)을 필요한 갯수만큼 사방형으로 배열한 형태이며 이의 i 번째 행과 j 번째 열에 존재하는 타일을 S(i,j)로 표시한다. 각 타일에는 첫째 메탈층, 둘째 메탈층, 비아(via), 콘택(contact), N 디퓨션, P 디퓨션, 폴리 등의 의미를 부여할 수 있으며 실제로 $S(i,j).m_1$, $S(i,j).m_2$, $S(i,j).via$, $S(i,j).con$, $S(i,j).ndif$, $S(i,j).pdif$ 그리고 $S(i,j).poly$ 등의 플래그(flag)를 필요에 따라 1로 함으로써 그 의미를 부여한다. 그런데 만약 모든 플래그가 0이면 그 타일은 빈 공간을 의미하고 어떤 타일이 폴리와 디퓨션이면 이는 트랜지스터의 게이트를 의미한다. 또한 $S(i,j).\neq t$에는 이 타일을 배선에 사용한 네트의 이름 또는 번호를 기록하며 만약 빈 공간이라면 $S(i,j).\neq t$에 -1을 기록한다. 어떤 배치결과가 w 개의 열로 구성되어 있다면 심볼릭 레이아웃의 폭은 2w이며 이의 2i-1 번째 열은 배치의 i 번째 열에 대응된다. 여기서 배치결과의 열에는 왼쪽에서 오른쪽으로 1, 2, 3,.... 순으로 인덱스(index)가 부여되어 있고 심볼릭 레이아웃의 열에는 왼쪽에서 오른쪽으로 0, 1, 2,.... 순으로 인덱스가 부여되어 있다고 가정한다. 그리고 심볼릭 레이아웃의 높이는 배선에 필요한 전체 수평트랙의 갯수와 같으며 이 값은 배선이 완료된 후에 결정된다. 또한 앞으로 설명할 방법에 의하여 심볼릭 레이아웃을 생성할 때 배치 결과의 둘째 메탈층의 수직선에 대응하는 심볼릭 레이아웃의 열에 존재하는 타일들은 이미 둘째 메탈층이라는 의미와 네트 번호가 부여되어 있다고 가정한다.

어떤 네트 n에 속한 터미날들을 배치결과의 왼쪽에서 오른쪽으로 그들이 놓여진 순서대로 t_{c1}, t_{c2}, ..., t_{ck}라고 하자. 여기서 c_1, c_2, ..., c_k는 각각 이 터미날들이 속한 열의 위치를 나타낸다. 네트 n을 심볼릭 레이아웃 S의 r 번째 트랙에 할당한다는 말은 S의 r 번째 행의 관련된 타일에 필요한 의미를 줌으로써 폴리선을 사용하여 이 네트에 속하는 트랜지스터의 게이트들을 만들고 첫째 메탈층의 수평선과 비아(via) 또는 콘택(contact)을 사용하여 필요한 터미날간을 연결하는 등의 배선을 수행하는 것을 의미한다. 만약 네트 n의 어떤 터미날 t_{cj}가 링크드 리스트 L_{cj}의 맨 앞에 존재한다면 이를 자유 터미날이라고 하고 네트 n의 모든 터미날이 자유 터미날이면 n을 자유 네트라고 한다. 만약 네트 n이 자유 네트이고 타일 $S(r,c_1)$, $S(r,c_1+1)$, ..., $S(r,c_k)$ 모두가 빈 공간이라면 n을 S의 r 번째 트랙에 할당할 수 있다는 것을 쉽게 알 수 있다. 그러나 이중 일부 타일이 빈 공간이 아니라면 네트 n 전체를 할당할 수는 없다.

```
procedure track_assignment(S, n, r, {t_cp, t_cp+1, ...
                                    , t_cp+q-1})
  for i=0 to q-1 do
    case
    : t_cp+i == 둘째 메탈층의 수직선:
      S(r,2c_p+i-1).via = S(r,2c_p+i-1).m1 =1;
      S(r,2c_p+i-1).net =n; s=r,2c_p+i-1;
    : t_cp+i == 트랜지스터의 드레인 또는 소스를 위한 연결
                점:
      S(r,2c_p+i-1).con = S(r,2c_p+i-1).ndif =
      S(r,2c_p+i-1).m1=1;
      S(r,2c_p+i-1).net =n; s =2cp+i-1;
    : t_cp+i == 트랜지스터의 게이트:
      S(r,2c_p+i-1).ndif =S(r,2c_p+i-1).poly =1;
      if S(r,2c_p+i-1).net == n or S(r,2c_p+i-1).net
      == -1 then
      S(r,2c_p+i-1).con =S(r,2c_p+i-1).m1
      =S(r,2c_p+i-1).poly=1;
      S(r,2c_p+i-1).net =S(r,2c_p+i-1).net =n;
      s = r,2c_p+i-2;
      else
      S(r,2c_p+i).con =S(r,2c_p+i).m1= S(r,2c_p+i).poly
      =1;
      S(r,2c_p+i-1).net =S(r,2c_p+i ).net =n;
      s = r,2c_p+i ;
      end if
    end case
    if i ≠ q-1 then
      for j = s to r,2c_p+i+1-1 do
      S(r,j).m1 =1; S(r,j).net =n;
      end for
      end if
      if t_cp+i ≠ 둘째 메탈층의 수직선 then
      t_cp+i 를 L_cp+i 에서 제거한다;
    end if
  end for
  end track_assignment
```

그림 9. 네트의 일부를 심볼릭 레이아웃의 r 번째 트랙에 할당하는 과정(N-part)

Fig. 9. A procedure to assign a part of nets to the l-th track of the symbolic layout(N-part).

자유 네트 n의 터미날의 일부인 q 개의 터미날 t_{cp}, $t_{cp}+1, \ldots, t_{cp+q-1}$를 고려하자. 만약 타일 $S(r, c_p)$, $S(r, c_p+1), \ldots, S(r, c_{p+q-1})$ 모두가 빈 공간이고 최소한 하나의 터미날이 둘째 메탈층의 수직선이거나 트랜지스터의 드레인 또는 소스를 위한 연결점이라면(이러한 두 조건을 할당 조건이라고 한다) 네트 n의 부분 네트 $\{t_{cp}, t_{cp+1}, \ldots, t_{cp+q-1}\}$ 는 트랙 r에 할당할 수 있으며 N-part의 경우 그 과정은 그림 9에 보인바와 같다.

그런데 이러한 할당 과정은 하나의 네트 전체를 현재 배선하는 트랙에 할당할 수 있는 경우에도 물론 그대로 적용할 수 있다. 네트 n이 자유 네트가 아니라고 하자. 만약 n의 일부 터미날들이 자유 터미날이고 이들이 위의 할당 조건을 만족한다면 이 네트를 부분 자유 네트라고 한다. 심볼릭 레이아웃을 얻기 위하여 사용한 배선 방법은 잘 알려진 왼쪽에지우선(left-edge-first) 배선 방법을 변형한 것이며[4,7] 레이아웃의 첫번째 트랙부터 시작하여 아래로 배선을 수행한다. 현재 r 번째 트랙에 네트를 할당한다고 가정하자. 먼저, 모든 자유 네트에 대하여 변형된 왼쪽에지우선 알고리즘[7]을 적용하여 가능한 많은 네트 또는 그 일부를 그림 9에 보인 과정에 의하여 배선한다[15].
다음, 모든 부분 자유 네트에 대하여 자유 네트와 같은 과정을 수행한다. 그런데 어떤 네트를 할당하는 도중에 만약 현재 트랙의 윗부분이 어느정도 비어 있다면 이 네트를 꺾어서 배선한다[15].

그림 10에 그림 8(b)의 배치결과를 배선한 심볼릭 레이아웃을 보인다.

지금까지 설명한 방법으로 심볼릭 레이아웃을 구성한 후에는 이를 실제 레이아웃으로 변환하여야 한다. 이를 위하여 먼저 비아와 콘택의 위치를 조정하여 가능한 작은 면적의 레이아웃을 얻을 수 있도록 한다. 그림 10을 보면 짝수 번째 열에는 폴리와 첫째 메탈층의 선을 연결하는 콘택들이 존재한다. 이러한 콘택은 레이아웃의 면적을 크게하는 요인이 되므로 이들을 가능한 많이 홀수 번째 열로 이동하여야 한다. 이를 위하여 먼저 심볼릭 레이아웃을 조사하여 만약 같은 네트에 속한 폴리 게이트에 해당하는 타일이 동일한 트랙에서 서로 이웃해서(즉, 2i-1, 2i+1, ...) 존재한다면 이들 각각에 인접해서 존재하는 콘택중 가장 왼쪽에 있는 것을 제외한 나머지를 폴리로 대치함으로써 불필요한 콘택을 제거한다.

일반적으로 대부분의 설계규칙에서는 폴리와 첫째 메탈층을 연결하는 콘택을 둘째 메탈층과 겹쳐서 놓는 것을 허용한다. 따라서 짝수 번째 열에 있는 각 콘택에 대하여 같은 트랙의 좌우로 둘째 메탈층의 수직선을 탐색하여 이를 놓을 수 있는 장소가 발견되면 이를 이동하고 관련된 첫째 메탈층의 선을 조정한다. 그림 11에 이러한 조정의 한 예를 보인다.

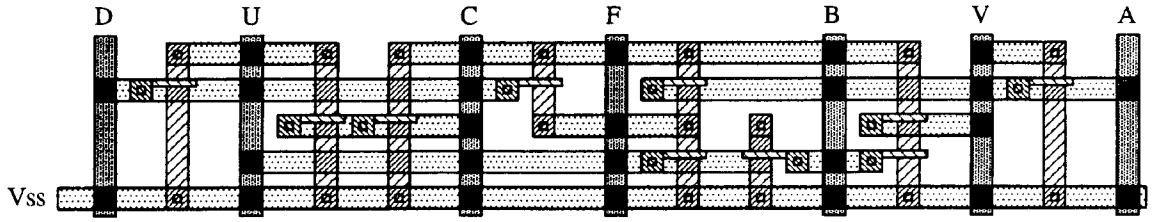
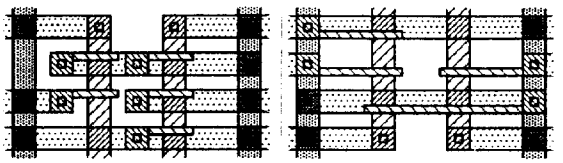


그림 10. 그림 8(b)의 배치결과에 대한 심볼릭 레이아웃
Fig. 10. A symbolic layout for the placement result in Fig. 8(b).



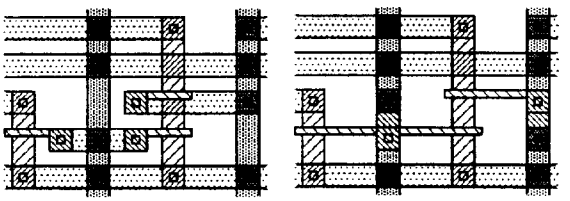
(a) 조정 전 (b) 조정 후
(a) Before adjustment. (b) After adjustment.

그림 11. 콘택의 위치 조정의 예
Fig. 11. An example of contact position adjustment.

그런데 이러한 콘택의 위치 조정을 위하여 비아도 이동할 필요가 종종 있는데 그 이유는 대부분의 설계 규칙에서 비아와 콘택의 겹침을 허용하지 않기 때문이다. 하나의 콘택을 이동하고자 하는 장소에 같은 네트에 속하는 비아가 존재할 경우 레이아웃의 상하 좌우를 탐색하여 이 비아를 이동할 장소를 찾는다. 만약 이러한 장소가 발견되면 비아를 그 곳으로 이동하고 다음 콘택을 원하는 장소로 이동한다. 그림 12 에 이러한 조정의 한 예를 보인다.

마지막으로 이러한 콘택들의 위치조정과정이 완료되면 심볼릭 레이아웃의 짝수번째 열중 콘택이 존재하지 않는 열은 존재할 필요가 없으므로 레이아웃에서 제거하고 회로의 중간 출력선에 해당하는 둘째 메탈층의 수직선은 위 아래의 불필요한 부분을 최대한 제거한다.

지금까지 기술한 과정에 의하여 생성된 심볼릭 레이아웃은 가상 그리드를 사용한 기존의 심볼릭 레이아웃과 마찬가지이다[12]. 따라서 이로부터 주어진 설계규칙에 따라 심볼릭 레이아웃의 위에서 아래로 그리고 좌에서 우로 인접한 열 그리고 행간의 최소 간격을 계산한 후 실제 매스크 데이타를 CIF 형태로 생성한다. 여기서 우리가 사용한 설계 규칙은 서울대 반도체 공동연구소의 1.5 μm 표준 CMOS 공정[6] 에 의한 것이며, 만약 다른 설계 규칙을 원하면 모듈 생성기의 설계규칙 화일을 바꾸어 사용하면 된다.



(a) 조정 전 (b) 조정 후
(a) Before adjustment. (b) After adjustment.

그림 12. 비아의 이동을 동반한 콘택의 위치 조정의 예
Fig. 12. An example of contact position adjustment with viaadjustment.

Ⅳ. 실험 결과

지금까지 기술한 방법에 의한 M^3 레이아웃 형태의 자동 기능모듈 생성기를 C 언어를 사용하여 구현하였다. 본 장에서는 이러한 자동 기능모듈 생성기를 통하여 생성된 여러 기능모듈들을 보이고 그들을 평가한다. 이를 위하여 사용한 컴퓨터는 삼보 200 워크스테이션이다.

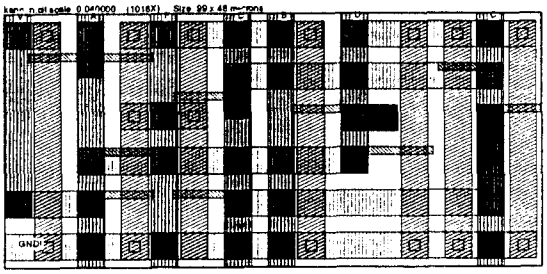


그림 13. 그림 8(a)의 회로의 N-part를 Gee 등[2]이 제안한 방법으로 구현된 결과

Fig. 13. Gee's result for the N-part of the circuit in Fig.8(a).

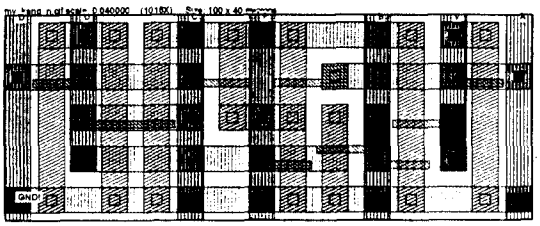


그림 14. 그림 8(a)의 회로의 N-part를 우리의 기능모듈 생성기로 구현한 결과

Fig. 14. Our result for the N-part of the circuit in Fig. 8(a).

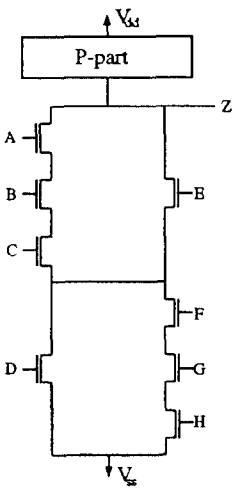


그림 15. CMOS 논리회로

Fig. 15. A CMOS logic circuit.

먼저 그림 13에 앞의 그림 8(a)에 보인 CMOS 회로의 N-part를 Gee 등[3]이 제안한 방법에 의하여 구현된 결과를 보인다. 이 결과는 참고문헌 [3]에 보인 결과를 우리의 설계 규칙에 맞도록 수정한 것이다. 그림 8(a)의 같은 회로의 N-part를 우리의 기능모듈 생성기로 구현한 결과를 그림 14에 보인다.

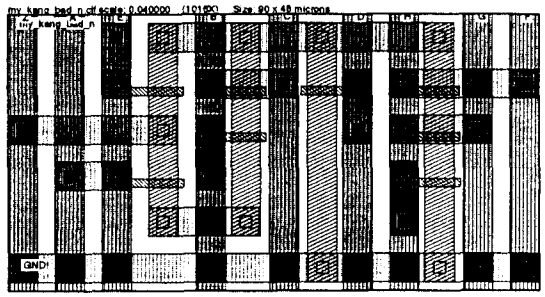


그림 16. 그림 15의 회로의 N-part를 우리의 기능모듈 생성기로 생성한 결과

Fig. 16. Our result for the N-part of the circuit in Fig. 15.

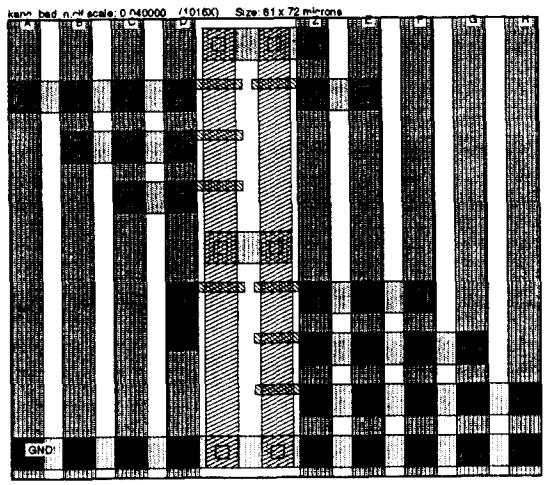


그림 17. 그림 15의 회로를 Gee 등[3] 의 방법으로 구현한 결과

Fig. 17. Gee's result for the N-part of the circuits in Fig. 15.

우리의 방법에 의한 레이아웃의 면적은 $100 \times 40 = 4,000\mu m2$ 로서 $99 \times 48 = 4,752\mu m2$ 인 Gee 등의 결과보다 15% 작다. Gee 등[3] 이 제안한 방법으로는 좋은 결과를 기대할 수 없는 그림 15 회로의 N-part를 우리의 기능모듈 생성기로 구현한 결과와 그들의 방법

으로 구현한 결과를 그림 16과 그림 17에 각각 보인다.

그림에서 보인 바와 같이 우리 결과의 면적은 90x48 =4,320μm 으로 그림 17의 면적 81x72=5,832μm에 비하여 74%에 불과하다. Gee 등[3] 의 방법은 transmission gate가 포함된 회로를 구현할 수 없다. 그러나 우리의 기능모듈 생성기는 임의의 형태의 회로에서도 그 기능모듈을 구현할 수 있다. 예를 들어 그림 18에 보인 회로는 "2:1 Selector"인데[2] 이의 구현 결과는 그림 19에 보인 바와 같다.

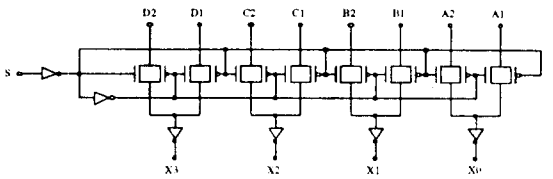


그림 18. Transmission gate가 포함된 논리 회로[2]

Fig. 18. A Logic circuit that has transmission gates[2].

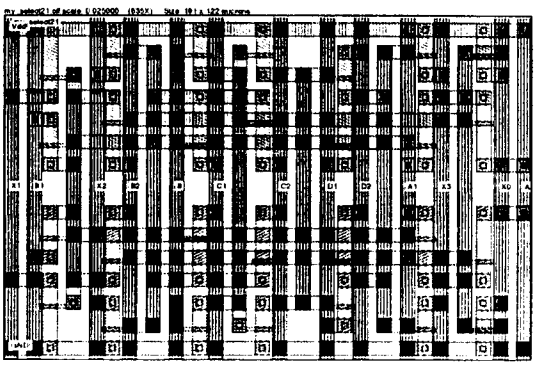


그림 19. 그림 18에 보인 회로에 대한 생성 결과

Fig. 19. Generation result for the circuit in Fig. 18.

끝으로, M^3 레이아웃 형태를 기존의 다른 레이아웃 형태, 즉, 게이트 매트릭스 레이아웃 형태[10,14]와 Uehara와 Van Cleemput의 레이아웃 형태[13] 등과 비교한다. 먼저, 그림 20에 보인 1비트 가산기 회로를 게이트 매트릭스 레이아웃 형태로 구현한 것을 그림 21(a)에 보이고 우리의 기능모듈 생성기로 생성한 결과를 그림 21(b)에 보인다. 그림에서 보인 바와 같이 M^3 레이아웃 형태가 게이트 레이아웃 형태보다 약간 큰 면적을 차지한다. 하지만, 참고문헌 [8]에서 지적한 바와 같이 게이트 매트릭스 레이아웃에서는 그 입출력에 폴리층을 사용하고 있으므로, 현재 주로 사용되고 있는 더블 메탈 공정기술에 효율적으로 사용하기 위해서는 폴리 입출력을 둘째 메탈층으로 바꿀 필요가 있어 추가의 면적이 필요하며 전체 면적은 우리의 M^3 레이아웃형태의 면적보다 약 3% 크다.

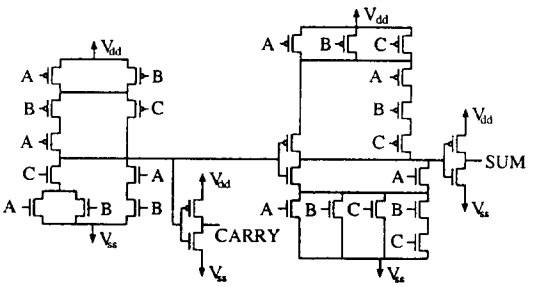


그림 20. 1 비트 가산기 논리 회로

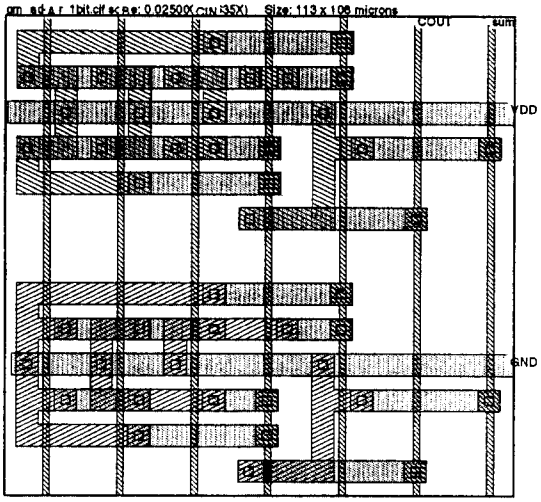
Fig. 20. A one bit full adder.

표 1. 모듈 생성에 소요된 CPU 사용시간

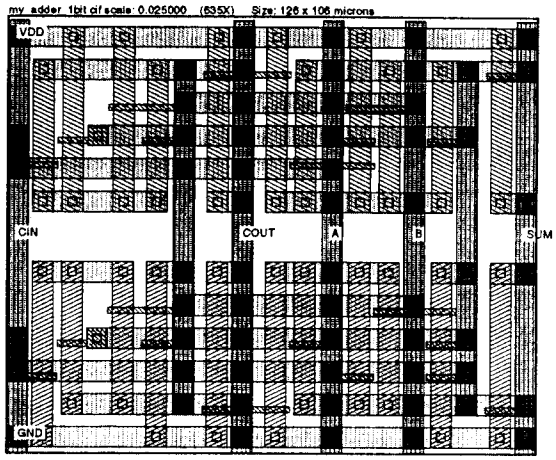
Table1. CPU time for our module generation.

실행환경	예		CPU 사용시간(min:sec)
SDT-200	그림 8.	kang1	2:15
	그림 15.	kang2	1:38
	그림 18.	select21	5:59
	그림 21.	adder21	6:28
	그림 22.	dff_hayes	3:55

Uehara와 Van Cleemput의 레이아웃 형태는 일반적으로 표준셀 레이아웃 형태에서의 셀라이브러리 구성에 자주 사용되는 형태이다[12,13]. 그림 22(a)에 참고문헌 [11]에 보인 D 플립플롭에 대한 결과를 우리의 설계규칙에 맞도록 수정한 형태를 보인다. 또한, 같은 회로를 우리의 기능모듈 생성기를 통하여 생성한 결과를 그림 22(b)에 보인다. 이 경우에는 M^3 레이아웃 형태의 면적이 폴리 입출력을 둘째 메탈층으로 바꾸는 경우를 고려하지 않아도 Uehara와 Van Cleemput 레이아웃 형태의 면적보다 작다. 마지막으로 지금까지 보인 모듈 생성 결과에 소요된 CPU 사용시간을 표 1에 보인다.

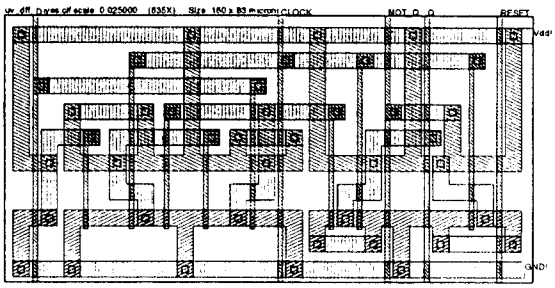


(a) 게이트 매트릭스 형태로의 구현.
(a) Layout in gate matrix style.

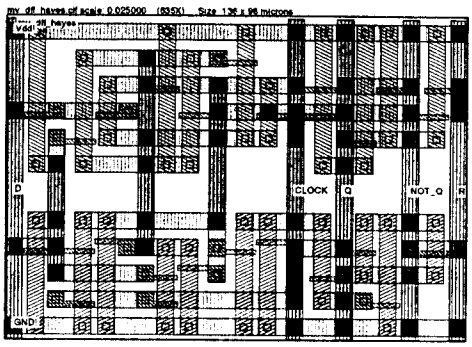


(b) M^3 레이아웃 형태로의 구현.
(b) Layout in M^3 layout style.

그림 21. 게이트 매트릭스 레이아웃과의 비교
Fig. 21. Comparison result with gate matrix layout.



(a) Uehara와 Van Cleemput의 레이아웃 형태로의 구현[11]
(a) Result in Uehara and Van Cleemput's layout style[11]



(b) M^3 레이아웃 형태로의 구현
(b) Result in M^3 layout style.

그림 22. Uehara와 Van Cleemput[13]의 레이아웃 형태와의 비교
Fig. 22. Comparison result with the Uehara and Van Cleemput[13] layout style.

Ⅳ. 결 론

본 논문에서는 M^3 레이아웃 형태의 자동 기능모듈 생성을 위한 새로운 배치 및 배선 방법을 제안하였다. 제안한 방법에서는 먼저 주어진 논리회로의 트랜지스터들과 중간 출력선 그리고 입출력선들을 시뮬레이티드 어닐링 기법을 사용하여 M^3 레이아웃 형태로 배치한다. 다음, 변형된 왼쪽에지우선 배선 알고리즘을 사용하여 배치된 결과를 심볼릭 레이아웃 형태로 만들고 비아와 콘택의 위치를 면적이 작도록 조정하는 작업을

거쳐 CIF 형태의 레이아웃 데이타를 생성한다.이러한 방법을 C 언어로 구현하여 삼보 SDT-200 워크스테이션에서 그 성능을 시험하였다. 시험결과 우리의 M^3 레이아웃 형태의 자동 기능모듈 생성기는 기존에 발표된 같은 형태로의 생성방법에 의한 결과보다 작은 면적에 기능모듈을 구현할 수 있었다. 또한, 같은 회로를 게이트 매트릭스 레이아웃 형태와 같은 다른 형태로의 생성결과와 비교할 때 우리의 생성결과는 그 면적에 있어서 이들의 면적보다 비슷하거나 작았다. 앞으로의 연구 내용으로 현재 개발된 기능모듈 생성기는 생성할 기능모듈의 aspect ratio를 조정할 수 없으며 터미날의 위치 또한 제한을 둘 수 없는데 이들을 보완할 수 있는 방법이 필요하다. 또한, M^3 레이아웃 형태의 최대 장점인 짧은 폴리선을 그대로 적용하면서도 트랜지스터의 드레인과 소스의 면적이 작은 새로운 레이아웃 형태의 개발도 시도할 가치가 있다.

참 고 문 헌

[1] B. S. Carlson, C. Y. R. Chen and U.Singh, ``Optimal Cell Generation for Dual Independent Layout Styles,'' IEEE Trans. on CAD, Vol. 10. No. 6, pp. 770-782, June 1991.

[2] CMOS Channelless Gate Arrays, 1989 Data Book, Fujitsu Microelectronics Inc., 1989.

[3] P. Gee, M. Y. Wu, S. M. Kang and I. N. Hajj, ``Metal-Metal Matrix(M^3) CMOS Cell Generator with Compaction,'' Proc. ICCAD, 1987, pp. 184-187.

[4] A. Hasimoto and J. Stevens, ``Wire Routing by Optimizing Channel Assignment Within Large Apertures,'' Proc. 18th Design Automation Workshop, pp. 155-168, 1971.

[5] D. K. Hwang, W. K. Fuchs and S. M. Kang, ``An Efficient Approach to Gate Matrix Layout,'' IEEE Trans. on CAD, Vol. CAD-6, No. 9, pp. 802-809, Sept. 1987.

[6] ISRC 교육부 MPC(Multi-Priject Chip) 안내, 서울 대학교 반도체 공동 연구소, 1992

[7] A. Mukherjee, Introduction to nMOS/CMOS VLSI Systems Design Prentice Hall, 1986.

[8] S. M. Kang, ``Metal-Metal Matrix(M^3) for High-Speed VLSI Layout,'' IEEE Trans. on CAD, Vol. CAD-6, No. 5, pp. 886-891, Sept.1987.

[9] S. Kirkpairick, C. D. Gelalt and M. P. Vecchi, ``Optimization by Simulated Annealing,'' Sience, Vol. 200, pp. 671-680, May 1983.

[10] A. D. Lopez and H-F. S. Law, ``A Dense Gate Matrix Layout Methods for Mos VLSI,'' IEEE Trans. on Electronic Devices, Vol. ED-27, pp. 1671-1675, Aug. 1980.

[11] R. L. Maziasz and J. P. Hayes, ``Exact Width and Height Minimization of CMOS Cells,'' Proc. 28th DAC, pp. 487-493, 1991.

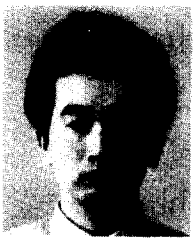
[12] T. Preas and M. J. Lorenzetti, Physical Design Automation of VLSI Systems , The Benjamin/Cummings Pub. Comp., Inc., 1988.

[13] T. Uehara and W. M. Van Cleemput, ``Optimal Layout of CMOS Functional Arrays,'' IEEE Trans. Comput., Vol. C-30, pp. 305-312, May 1981.

[14] O. Wing, S. Huang, and R. Wang, ``Gate Matrix Layout,'' IEEE Trans. on CAD, Vol. CAD-4, pp.220-231, July 1985.

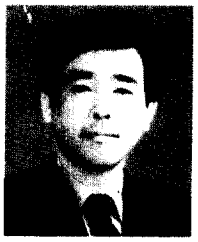
[15] 임종석외 3 인, 메탈-메탈 매트릭스 레이아웃에 관한 연구, 최종 보고서, 한국과학재단, 1993

저 자 소 개



車 榮 俊(正會員)

1992年 2月 서강대학교 전자계산학과 학사. 1994年 2月 서강대학교 전자계산학과 석사. 1994年 3月 ~ 현재 한국전자통신연구소 교환기술연구 제어시스템연구실. 주관심 분야는 자동설계, 운영체제, ATM 시스템 등임.



林 鍾 錫(正會員)

1981年 서강대학교 전자공학과 학사. 1983年 한국 과학 기술원 전기 및 전자공학과 석사. 1989年 Univ. of Maryland, College Park, 전기공학과 박사. 1983年 3月 ~ 1990年 8月 한국전자통신연구소 연구원. 1990年 9月 ~ 현재 서강대학교 부교수. 주관심 분야는 CAD 알고리즘 및 VLSI설계 등임.