論文2001-38SD-12-9

마스크 레이아웃 합성을 위한 벡터화한 변을 사용한 블록 분할 기법

(A Block Disassembly Technique using Vectorized Edges for Synthesizing Mask Layouts)

孫 永 燦 *, 朱 利 亞 **, 劉 尙 大 ***

(Yeong-Chan Son, Ri-A Ju, and Sang-Dae Yu)

요 약

오늘날 집적회로의 집적도가 증가되고 있기 때문에 회로 소자는 기생성분의 영향을 최소화하고 회로의 성능을 감소시키는 요인을 최소화하도록 설계되어야 한다. 그래서 칩을 제작하기 전에 레이아웃으로부터 추출한 회로가 정확한가를 검증하고 시뮬레이션으로 추출된 회로가 설계사양을 만족하는지를 확인해야 한다. 본 논문에서는 스택 구조의 MOSFET의 기하학적인 파라미터와 레이아웃 배선 블록의 분산 RC를 추출할 수 있는 새로운 블록 분할 기법을 제안한다. 폴디드 캐스코드 CMOS 연산 증폭기의 레이아웃에 이 기법을 작용하여 회로를 추출하고, Hspice로 시뮬레이션을 수행하여 전기적 연결관계와 이들 소자의 영향을 검증하였다.

Abstract

Due to the high density of integration in current integrated circuit layouts, circuit elements must be designed to minimize the effect of parasitic elements and thereby minimize the factors which can degrade circuit performance. Thus, before making a chip, circuit designers should check whether the extracted netlist is correct, and verify from a simulation whether the circuit performance satisfies the design specifications. In this paper, we propose a new block disassembly technique which can extract the geometric parameters of stacked MOSFETs and the distributed RCs of layout blocks. After applying this to the layout of a folded-cascode CMOS operational amplifier, we verified the connectivity and the effect of the components by simulating the extracted netlist with HSPICE.

Ⅰ. 서 론

반도체 집적회로 공정기술의 발전으로 레이아웃의 배선 폭이 작아지고 상대적으로 무시되었던 이들 배선의 분산 RC는 회로의 특성을 왜곡시키는 원인으로 작용하고 있다. 가능한 최소의 기생성분이 생성되기 위해서 다중접점으로 레이어를 연결하고 또한 MOSFET의 확산영역의 분산 저항을 최소화하기 위하여 스택구조로 설계하고 있다. 자동으로 레이아웃을 생성하는 경우도 있지만 고성능의 연산증폭기와 같은 회로를 설계하는 경우에는 접점과 소자를 레이아웃하고 라우팅을 직접해야 하는 경우가 많다.

또한 칩 제작에 앞서 설계된 레이아웃으로부터 회로를 추출하고 시뮬레이션을 통하여 회로의 성능을 검증하고 있다. 성능에 대한 기대에 앞서 레이아웃의 논리적 회로가 정확하게 이루어졌는가를 ERC를 수행하여

* 正會員, 浦項1大學 컴퓨터應用科
(Department of Computer Engineering Pohang College)
** 正會員, 慶北大學校 大學院 電子工學科
(Department of Electronic Engineering Kyungpook National University)
*** 正會員, 慶北大學校 電子電氣工學部
(School of Electronic and Electrical Engineering Kyungpook National University)
接受日字:2000年5月3日, 수정완료일:2001年11月13日

레이아웃에 대한 전기적 연결관계를 우선적으로 평가해야 한다. 그 다음으로 기하학적인 구조에 따른 소자의 L/W 외의 여러 가지 파라미터와 배선간에 생성되는 접점 저항과 분산 RC를 추출하여 보다 근접한 출력 특성을 구해야 한다.

현재 유한요소법을 사용하는 SPACE[1], 타일을 기본으로 만들어진 Magic의 회로 추출기[2,3], 그리고 라이브러리 룩업 (lookup) 방식을 이용한 EXCL[4] 등의 회로 추출기가 널리 사용되고 있다. SPACE는 유한요소법을 사용하여 레이아웃의 단자 사이 저항과 커패시턴스 값을 근사화하기 때문에 CMOS SRAM과 같은 기본 셀에 대한 회로 추출 시간이 10분 이상 소요되므로 회로의 크기가 클수록 추출에서 검증까지 많은 시간이 필요하다. Magic의 회로추출기는 한 절점과 다른 인접 단자 사이에 놓인 모든 타일의 면적에 의해서 C를 계산한다. 만약 연결 단자 사이에 분기된 가지가 생성된 경우에 저항 성분을 추출할 때 연결 단자 사이에서 분기된 다른 가지를 무시하고 단자사이의 길이에 의해서 저항 값을 추출하기 때문에 두 절점사이의 추출된 저항 값에는 큰 오차가 발생한다.

EXCL에서는 정형화된 패턴에 대한 라이브러리를 만들고 이 라이브러리를 참조하면서 추출기능을 수행하는데, 그 특징은 레이아웃에서 기하학적으로 생성된 패턴을 라이브러리에서 탐색하고 해당하는 패턴에 대한 이미 계산된 값을 찾아 대응시킨다. 만약 라이브러리에 해당하는 패턴이 없으면 그 패턴에 대한 근사 값을 계산하여 라이브러리에 포함시키는 방식을 사용하고 있다. 그러므로 이 방법은 직각으로 굽은 영역이나 전류분산이 일어나는 영역에 대해서 빠르게 등가회로를 추출할 수 있다. 그러나 라이브러리에 포함되지 않은 패턴에 대해서는 여러 가지의 패턴을 추출과정에서 정형화하여 라이브러리화 해야하는 부가적인 과정을 필요로 하기 때문에 작업 과정이 복잡하고 많은 시간이 필요하다.

본 논문에서는 타일에 의해서 표현되는 Magic의 레이아웃으로 생성되는 도형에 대한 기하학적인 정보를 입력으로 하고, edge-based scanline 알고리즘[5,6]을 이용하여 각 레이어 별로 병합하여 독립된 블록을 생성한다. 이들 독립된 블록의 경계를 나타내는 변을 반시계 방향의 방향성 변 (directional edge)으로 블록을 기술한다. 한 블록의 방향성 변과 인접한 다른 독립 블록과의 전기적 연결을 갖는 변을 사용하여 블록에 대한 등가의 분산 RC와 레이어의 측면 커패시턴스를 추출할 수 있는 블록 분할 알고리즘을 제안한다.

EXCL에서 분기되는 위치는 수직 또는 수평 분할된 사각형 변이 다른 사각형과의 인접형식에 의해서 결정되어 사각형의 등가 RC를 추출하지만, 제안한 알고리즘은 분기가 형성되는 위치를 블록의 변을 기술하는 방향성 변의 한 쌍에 의해서 분기되는 위치가 결정된다. 이 한 쌍은 전체 블록에서 분할되어 독립된 한 개의 블록을 형성하면서 추출과정을 단순화한다. 블록 분할 알고리즘은 Magic의 분기점에서 갖는 저항 값 오차를 극복하고, SPACE가 가지는 추출 시간 문제와 EXCL의 라이브러리 룩업 참조와 비정형화된 형태에 대한 모델링에 따른 문제를 완화한다. 제안된 블록 분할 기법을 사용하여 만들어진 회로 추출기의 출력 회로는 EXCL의 출력결과와 유사하다. 이 추출기를 사용하여 레이아웃으로부터 추출된 netlist와 초기 회로의 netlist에 대한 성능을 평가한다.

Ⅱ. 추출기의 구성

벡터화한 변에 기초한 블록 분할 기법을 이용한 회로 추출기의 전체 구성도는 그림 1과 같다. 입력 데이터는 기하학적인 정보로 표현된 CIF[7] 또는 Magic의 데이터 형식인 타일 정보를 입력으로 받아들인다. 그리고 각 레이어에 대한 시트 저항과 단위 용량, 그리고 CMOS 공정파라미터의 값을 입력으로 하며, 레이아웃 상의 입출력 단자와 전원 단자를 식별하기 위하여 초기의 Spice 파일에서 입출력 단자와 레이아웃과 외부적으로 연결되는 접속 단자에 대한 정보를 도형 정보와는 별개의 입력으로 기술한다. 이 입력은 레이아웃과 외부단자의 연결을 기술하기 위한 정보로써 레이아웃 편집기에서 이들 연결 단자에 대한 레이블 명칭을 기술한다. 또한 입력 데이터 중에는 출력형식을 결정하는 정보가 기술되어야 한다.

Hspice, Spice3, 그리고 PSpice와 같은 여러 종류의 시뮬레이터가 있기 때문에 출력형식으로 시뮬레이터의 종류를 기술해야 하고 분산 RC의 출력 모델을 지정해야 한다. 분산 RC의 출력 모델에는 lumped, T1, T2, T3, π1, π2, π3 등이 있으며, 또한 분산 RC와 접점 저항을 무시하고 스택 구조로 설계된 병렬 MOSFET들을 단일의 MOSFET로 인식하여 논리적 회로를 출력할 수 있는 모델이 포함된다. 다른 모델은 추출 성분의 정

확도에 따라 모든 기생 성분을 추출하지만, 논리적 회로 모델 추출은 설계된 레이아웃이 시뮬레이션을 위해 작성한 초기 회로 기술 파일과 같은 전기적 연결을 가지도록 설계되었는가를 판단할 수 있으므로 기생성분의 영향을 고려하지 않고 논리적 연결관계만을 판단할 때 사용할 수 있는 모델이다.

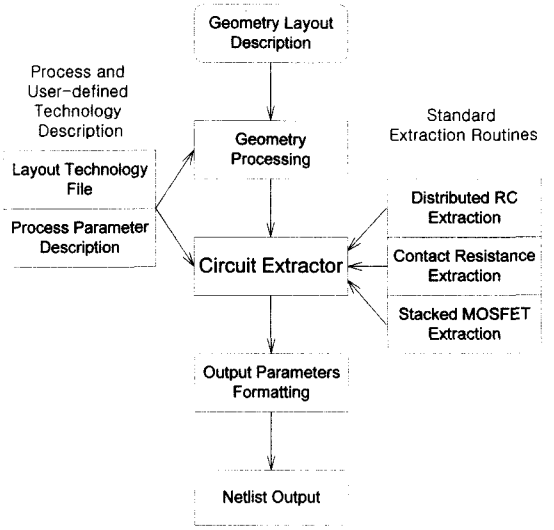


그림 1. 회로 추출기의 구성도
Fig. 1. Diagram of circuit extractor.

본 추출기에서 추출하는 성분은 레이어와 레이어의 접점에 위치한 접점 저항 추출, 단일 또는 스택 구조로 설계된 MOSFET 소자와 관련된 기하학적인 파라미터, 소자와 소자를 연결하는 선에 대한 분산 RC, 그리고 레이어의 측면에 위치하는 커패시턴스 (peripheral capacitance) 등이다. 기하학적인 도형 처리에서는 레이어별로 그룹을 만들어 데이터 구조에 저장하고 분류한다. 위의 입력 데이터를 기초로 제안한 블록 분할 알고리즘을 적용하여 회로추출기에서는 지정된 형식으로 소자와 기생성분의 출력 파일을 생성하고 또한 입력 데이터 중의 입출력 단자들에 대한 시뮬레이션 환경을 출력파일에 포함하여 시뮬레이터에 바로 입력될 수 있도록 구성하였다.

1. 기하학적인 도형 처리 및 블록 병합

Magic에서 레이아웃 설계 규칙에 의해 생성된 타일은 특정 형식으로 각 레이어와 빈 공간에 대한 부분까지 모두를 나타내도록 만들어져 있다. 이러한 타일을 각 레이어별로 분류하고 분류된 타일을 edge-based scanline 알고리즘을 사용하여 병합한다. 하나의 레이어에 대하여 다수의 블록이 구성되며 각각의 블록은 반시계 방향의 변에 의해 연계리스트 (linked-list)로 표현한다. 그림 2(a)는 타일로 표현된 스택구조의 MOSFET의 공유된 드레인과 소스 단자와 연결된 metal1에 대한 타일구조로 표현한 레이아웃이며, 그림 2(b)는 이것을 병합하여 생성된 블록의 외형을 CCW (counter-clockwise)의 벡터 변으로 표현하였다.

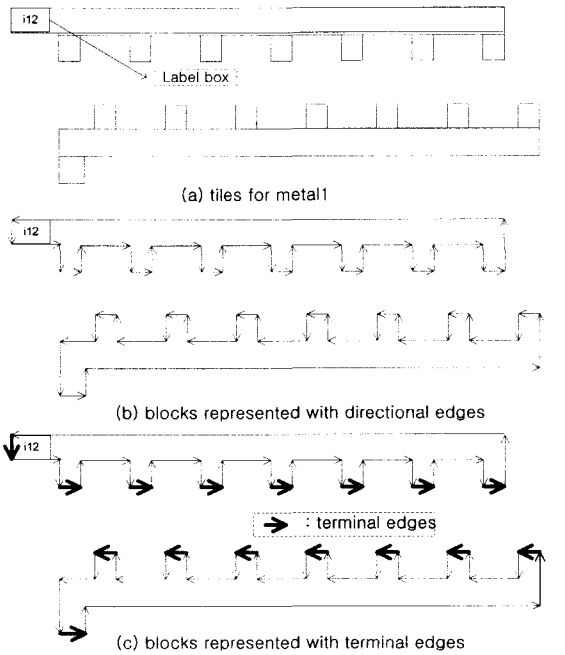


그림 2. Magic에서 타일을 병합한 후 반시계 방향의 방향성 변으로 표현한 블록
Fig. 2. Tiles in Magic and blocks represented with directional edges in CCW after merging these tiles.

한 개의 변은 수직 또는 수평선의 형식이기 때문에 3개의 정수로 표현하며, 처음 두 정수는 3번째 정수와 쌍을 이루어 변의 시작점과 종단점의 (x, y) 좌표 점을 표현한다. 예를 들면, 수평선이면 앞 두 좌표는 변의 좌우의 x좌표이고 3번째 정수는 이 변의 y좌표이다. 그리고 각 변의 좌측은 레이어 블록 영역이며, 우측은 블록의 외부 영역이다. 연계리스트에 포함된 한 개의 변에 대한 데이터 구조는 다음과 같이 변의 방향, 좌표, 그리고 변에 대한 독립적인 절점번호와 이웃한 변을 가리키는 두 개의 포인터로 구성되어 있다.

```
typedef struct edge {
    int a0;  // (a0 a1 b)는 변의 좌표가
    int a1;  // 수평선이면 (a0 a1 b) = (left right y)
    int b;   // 수직선이면 (a0 a1 b) = (bottom top x)
} Edge;
typedef struct edgelist {
    int direction;          // 변의 방향 기술
    int node_number;        // 변의 절점 번호
    Edge coordinate;        // 변의 좌표
    edgelist *next;         // CCW에 위치한 변을 지시
    edgelist *prev;         // CW에 위치한 변을 지시
}  EdgeList
```

백터화한 변으로 표현된 블록을 회로 추출에 사용되기 전에 추출할 블록과 인접할 수 있는 레이어를 우선적으로 구분해야 한다. 예를 들면 metal1 블록의 변과 연결관계를 가지는 블록은 metal1과 다른 레이어 사이를 연결하는 접점 블록이고, MOSFET의 게이트와 인접한 타일은 폴리실리콘과 확산영역의 블록이 있다. 추출할 블록의 변과 인접하는 변을 terminal edge (TE)로 정의하고, TE는 회로추출과정에서 전기적으로 다른 레이어와 연결되어 있으므로 독립된 하나의 절점으로 표현하기 위해서 terminal node (TN)로 정의한다.

그림 2(c)는 metal1 블록의 변 중에서 접점과 연결을 가지는 TE를 나타낸 것이다. TE로 인식되는 다른 경우는 한 변이 신호 단자와 같은 입력단자를 표현하거나 외부 전원 연결을 레이아웃에 나타내기 위하여 레이블을 사용하여 metal1 블록의 변이 레이블 블록의 변과 겹치는 변을 TE로 표현하였다. 그림 2(c)에서 'i12'는 외부 전류 전원이 연결되는 것을 나타내기 위한 레이블인데 이러한 변을 또한 TE로 분류한다.

2. 블록 분할 알고리즘 기술

TE로 표현된 블록을 사용하여 각 블록에 대한 등가의 분산된 RC 추출을 시작한다. 블록을 기술하는 연계리스트에서 TE의 한 변을 선택한다.

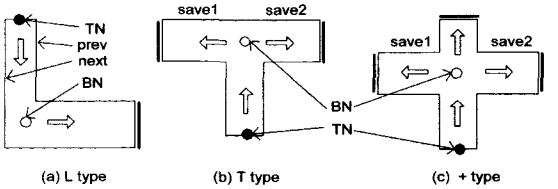


그림 3. 추출에 사용되는 기본 구조
Fig. 3. Basic structures for extraction.

그림 3(a)는 선택된 TE와 연결된 앞뒤의 두 변에서 추출 진행 방향의 변을 next, 반대 방향의 변을 prev로 표시하고, 이들 쌍 (next, prev)을 EP (edge pair)라고 정의한다. Manhattan 기하학 도형을 사용하므로 EP는 수직 또는 수평으로 나란히 위치하고 진행 변의 등가 분산 RC 추출의 기초가 된다.

EP가 진행하면서 다음 진행이 생성될 수 있는 형태는 L, T, 또는 +자 형으로 그림 3과 같으며, 진행 중에 진로가 변경되는 위치를 branch node (BN)로 표기하였다. 기본형에서 그림 3(a)의 경우와 같이 가지가 왼쪽으로 L자 형으로 판단되는 경우는 다음의 조건을 만족한다. 추출 진행방향이 먼저 결정되고 prev와 next 변과 직교하는 prev->prev와 next->next의 진행 방향이 서로 반대로 위치하고 이들 두 변이 다음 진행의 EP를 만들면 이들 EP는 L자형을 구성한다. L자형이 형성되는 곳에서는 새로운 절점 생성은 되지 않는다.

next->next와 쌍을 이루는 변을 save1, 이들 쌍 (next->next, save1)을 NEP (next edge pair)라 하고, prev->prev와 쌍을 이루는 변을 save2, 이들 쌍 (save2, prev->prev)을 PEP (prev edge pair)라고 정의할 때, NEP의 진행 방향은 next->next에 의해서, PEP의 진행 방향은 save2에 의해서 결정된다. NEP와 PEP가 모두 형성되고 save1과 save2가 동일한 변이거나 또는 save1->prev와 save2->next가 동일한 변이면 T자형이 형성된다. 예를 들면, 그림 3(b)에서 NEP는 (next->next, save1)이고 PEP는 (save2, prev->prev)인데 이 경우에는 save1과 save2가 동일한 변이므로 T형의 분기가 형성된다.

그림 3(c)는 save1과 save2가 동일한 변이 아니고 또한 save1->next와 save2->prev가 동일한 변이 아니므로 (save1->next, save2->prev)을 쌍으로 하는 분기가 형성되므로 +자형이 형성된다. Manhattan 기하에서는 +자형의 경우처럼 최대 3개의 가지가 생성될 수 있고, 또한 T자형과 +자형이 형성된 곳에서는 새로운 절점 (BN)이 생성된다.

TE에서 추출 과정을 시작하여 또 다른 TE를 만나면 한 개의 가지에 대한 가지 자르기가 완성되므로 한 개의 부분 블록에 대한 분산 RC 추출을 마친다. Edge-based 블록 분할 알고리즘에 대한 가상 코드는 그림 4와 같다.

그림 4에서 findCounterpartEdge는 매개변수와 EP를 이루는 변을 찾고, 이들 EP를 매개변수로 하여 재귀적

함수인 recursiveCall을 호출한다. 이 알고리즘은 추출 과정에서 탐색되는 변의 수를 최소화한다. 연계리스트로 표현된 블록 정보에서 특정 진행 과정의 EP로 (next, prev)가 선택되었을 경우, recursiveCall을 호출하면서 추출하게될 그 가지에 대한 시작 변 (next)에서 끝 변 (prev)까지의 정보를 가지므로 탐색되는 변의 최대 수는 next에서 prev까지의 변의 개수 이하이다. 그러므로 특정 가지에서 진행을 마치고 동일 레벨의 다음 recursiveCall에서는 한번 탐색한 가지에 대해서 반복 탐색 과정이 발생하지 않는다.

```
void recursiveCall( EdgeList *next, EdgeList *prev )
{
    if a new branch created in only prev side
        save1 = findCounterpartEdge( prev->prev );
        if save1 == next->next                          // L type
            recursiveCall( save1, prev->prev );
        else                                            // T type
            recursiveCall( save1, prev->prev );
            recursiveCall( next, save1->prev );
    else if a new branch created in only next side
        save2 = findCounterpartEdge( next->next );
        if save2 == prev->prev                          // L type
            recursiveCall( next->next, save2 );
        else                                            // T type
            recursiveCall( next->next, save2 );
            recursiveCall( save2->next, prev );
    else if branches are created in both directions     // T type
        save1 = findCounterpartEdge( prev->prev );
        save2 = findCounterpartEdge( next->next );
        recursiveCall( save1, prev->prev );
        recursiveCall( next->next, save2 );
        if the third branch is created                  // + type
            recursiveCall( save2->next, save1->prev );
}
```

그림 4. 블록 분할에 대한 가상 코드
Fig. 4. Pseudo code for block disassembly.

3. 예외적 문제 처리

레이아웃에서 생성될 수 있는 다각형의 블록 형태는 다양하게 존재하므로 제안된 블록 분할 알고리즘에 의해서 처리할 수 있는 레이아웃 상의 특별한 형태의 블록을 그림 5에 나타내었다.

그림 5(a)는 접점이 변의 선상에 위치하는 경우이며, (b)는 접점이 블록과 부분적으로 겹친 경우이고, (c)는 접점이 블록의 내부에 위치하는 경우이며, (d)는 TE가 서로 인접한 변에 위치하고 있고, (e)는 prev->prev 또는 next->next와 대응하는 변을 가지지 않을 때이며, 그리고 (f)는 TE의 prev와 next사이에 다수의 TE나 변이 위치하는 블록이 형성되었을 때를 기준으로 설명한다.

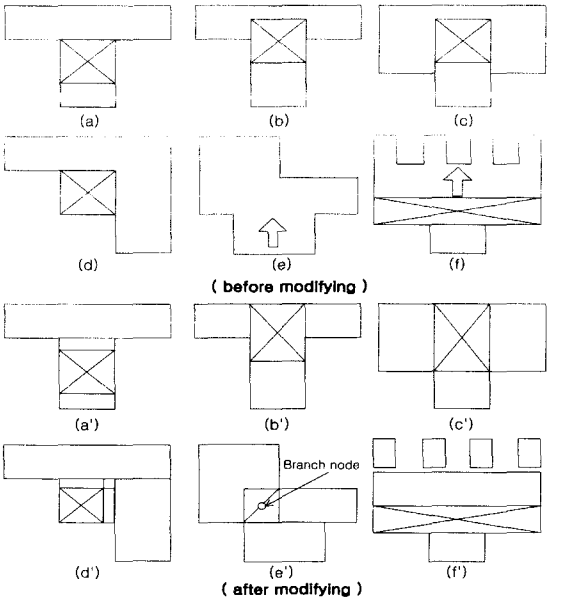


그림 5. 블록 분할의 특수한 경우
Fig. 5. Special cases for block disassembly.

그림 5(a)의 경우는 접점과 인접한 TE가 인접한 변의 폭만큼의 최소의 길이를 덧붙여 그림 5(a')의 블록과 같이 블록을 변경시키고, (b)와 (c)의 경우는 (b')와 (c')의 경우와 같이 한 개의 블록을 두 개의 블록으로 분할한다. (d)의 경우는 인접한 두 TE를 (d')과 같이 분리하고 (a')과 같은 형식으로 변형한다. (e)의 경우는 추출과정에서 처리하는 블록의 형태인데, NEP는 형성되지만 PEP가 형성되지 않는 경우에는 (e')에서와 같이 교차하는 블록의 대각선의 중앙에서 가지 점을 설정하여 기본형의 T자형과 같이 근사화한다. (f)의 형식과 같이 EP의 내부에 빗 모양이 형성되었을 때 (f')의 분할된 것과 같이 다중블록으로 나누고, 각 부분에서 다시 연결되는 기준점은 동일의 절점 번호를 가지도록 하여 현재의 EP의 종단점에서 분기된 형식으로 근사화하였다.

그림 5(a)에서 (d)까지는 추출 전에 조사하여 블록을 변형하고, (e)와 (f)는 추출 중에 생성되는지를 판단한다. 이렇게 분할된 각 부분 블록은 독립 블록에 대한 기본형의 추출과정에서와 같이 적용된다.

4. 소자 추출과 절점 생성

복잡한 구조의 도체를 여러 개의 블록으로 나누면서 연속된 블록은 직렬 RC의 가산으로, 분기되는 위치에서는 분기점을 생성하면서 추출한다. 그림 6은 계단형

과 다중 분기형의 블록에 대한 추출을 도식적으로 나타내고 있으며, 그림 6(a)에서 n1에서 n7까지의 모든 생성 노드가 L자형을 형성하고 있으므로 TN1에서 TN2까지의 분산 RC의 값은 각 절점사이의 직렬 성분의 합으로 계산된다. 그림 6(b)는 n1, n3 그리고 n6 절점에서 L자가 생성되었고, n2와 n5에서 T자형이, 그리고 n4에서는 +자 형이 형성되어 있다. 이 경우의 추출되는 순서는 TN를 만나는 과정별로 나타내면, (TN1→n1→n2→n3→TN2), (n2→n4→n5→TN5), (n5→n6→TN4), (n4→TN3), (n4→TN6)의 순서로 추출과정을 갖는다. 그러므로 n2, n4, n5에서는 새로운 절점이 생성되고 그림 6(b)의 등가회로와 같이 이 절점과 각 TN과의 등가 RC 값이 추출된다.

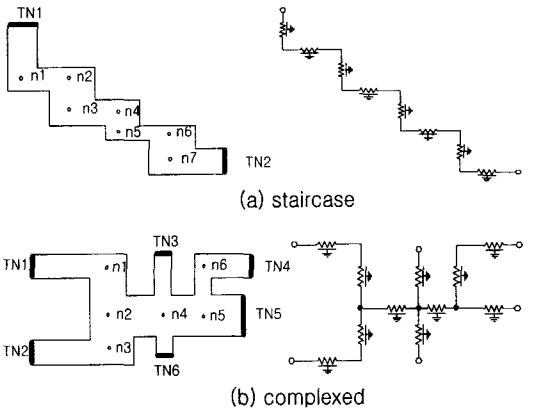


그림 6. 실험상의 추출 블록
Fig. 6. Experimental blocks for extraction.

5. 등가 저항 및 커패시턴스 계산

절점사의의 저항 값은 직선으로 이루어진 두 절점 a, b 사이의 경우 (next, prev)의 폭(W)과 절점사이의 길이(L)가 주어졌을 때, L의 길이가 충분히 긴 경우라면, 두 절점사이의 분산 RC는 다음과 같이 계산할 수 있다. 시트저항 값을 ρ_{sh}로 표현하면, 저항 값은

$$R_{ab} = \rho_{sh} \cdot \frac{L}{W} \tag{1}$$

이고, 평행판에 대한 상수값을 $K_{plate} = \frac{\varepsilon_{oxide}}{depth}$으로 나타내고 레이어의 측면에 대한 상수값을 α_{fringe}로 나타내면 커패시턴스는

$$C_{ab} = K_{plate} \cdot (W \times L) + \alpha_{fringe} \cdot (2W + 2L_{prev}) \tag{2}$$

로 근사화할 수 있다. 두 절점 a에서 b까지 가상 절점이 m개 포함되어 있을 경우 RC 계산은

$$R_{ab} = \sum_{i=1}^{m} \rho \cdot \frac{L_i}{W_i} \tag{3}$$

$$C_{ab} = \sum_{i=1}^{m} (K_{plate} \cdot (W_i \times L_i) + \alpha_{fringe} \cdot (2W_i + L_{i\,\mathrm{prev}} + L_{i\,\mathrm{next}})) \tag{4}$$

로 근사화한다. 여기서 가상 절점이란 절점 a에서 b까지의 진행 중 경로가 90°로 변하거나 폭이 변하는 곳의 위치에서 생성되는 절점이지만 두 절점은 전기적으로 단일 경로를 이루고 있는 경우이다. 특히 edge-based 블록 분할 알고리즘을 이용하여 레이어의 측면 성분의 C는 추출의 진행 성분이 next와 prev의 변으로 이루어졌기 때문에 식 (4)의 (Liprev + Linext)와 같이 EP 변의 길이로 계산할 수 있다.

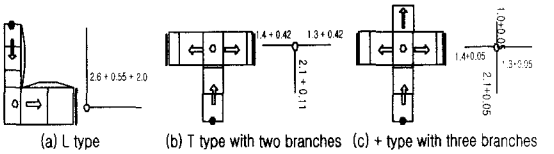


그림 7. 분기 점에서 등가 저항 추출
Fig. 7. Equivalent resistance extraction at the branch point.

L, T, 또는 +자로 형성되는 절점에서의 저항 계산은, 그림 7(a)와 같은 L자 형인 경우는 동일한 시트 저항의 0.55배로 계산하고, T자와 +자로 형성된 곳에서는 그림 7과 같이 근사화된 값[4]을 참조하여 대응시킨다.

6. 접점저항 및 MOSFET 추출

인접해 있는 두 레이어를 연결하는 접점은 일반적으로 금속의 시트 저항 값보다도 상당히 큰 값을 가진다. 이들의 값을 연결 정보에 포함하여 레이아웃에 대한 보다 정확한 근사 모델이 요구된다. MOSFET 스택구조에 있어서 확산 영역의 저항 값을 감소시키기 위하여 그림 8(a)의 금속과 확산 영역의 접점에서와 같이 다중 접점으로 연결하고, 또한 인접한 다른 레이어와의 연결에서도 접점 저항을 감소시키기 위해 다중접점으로 이들을 연결하는 것이 일반적이다. 다중 접점인 경우의 접점 저항의 등가 모델은 공정 파라미터에서 참조한 접점 저항의 값에서 다중 접점 수로 나누어 계산하여 인접한 TN사이에 포함하도록 하였다.

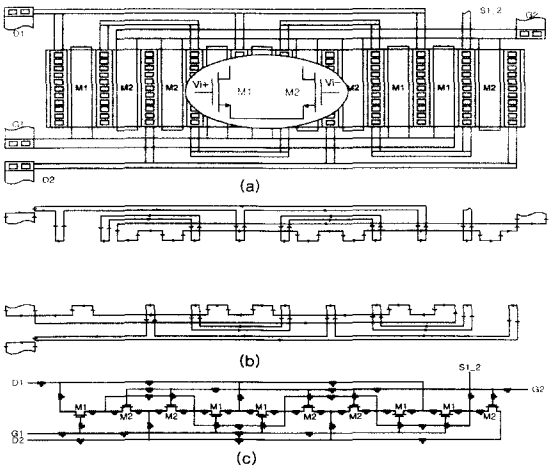


그림 8. 스택 구조의 MOSFET에 대한 등가회로 추출
Fig. 8. Equivalent circuit extraction for stacked MOSFETs.

본 회로 추출기에서 드레인과 소스 단자를 공유시키고 동일한 채널 길이를 가지도록 설계한 스택구조를 인식할 수 있다. 스택구조를 구성하는 배선 성분에 대해서도 분산 RC를 추출하여 출력에 포함한다. 입력 파일 중 시뮬레이션을 위한 환경 설정 파일에서 기술되어진 ACM (Area Calculation Method)[8] 모드에 따라서 모델 파라미터의 값이 계산된다.

그림 8(a)은 차동 증폭기의 차동단에서 소스 영역을 공유한 두 MOSFET의 레이아웃이다. 그림 8(b)는 추출할 블록에 대한 방향성 변으로 나타낸 것이며, 기생 성분을 포함하여 추출된 회로망은 그림 8(c)와 같이 등가 RC에 대한 추출 모델의 등가회로에 대한 값으로 출력된다.

Ⅲ. 실험 결과

본 논문에서 제안한 블록 분할 기법을 사용하여 레이아웃으로부터 MOS의 기하학적인 파라미터와 단자, 그리고 이들을 연결하는 전기적 연결관계 정보를 추출하여 회로추출기의 기능을 검증하고 초기회로와 설계된 레이아웃의 성능을 평가한다. 예제로 그림 9의 조정된 캐스코드를 가지는 폴디드 캐스코드 CMOS 연산증폭기와 그림 13의 2단 연산증폭기를 사용하였으며, 각 회로의 레이아웃을 그림 10과 그림 14와 같이 설계하였다. 그림 11은 그림 10의 레이아웃으로부터 각 레이어별로 병합한 블록이다.

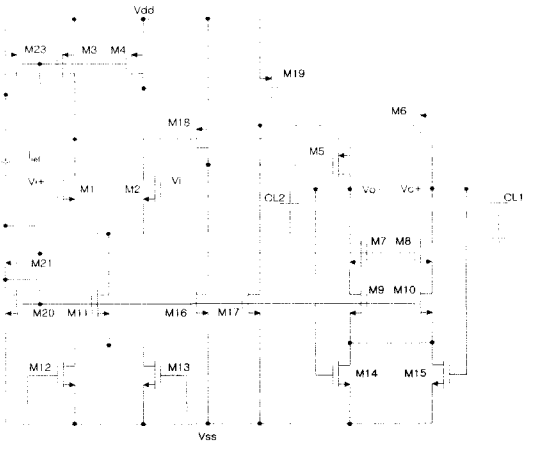


그림 9. 폴디드 캐스코드 CMOS 연산증폭기
Fig. 9. A folded-cascode CMOS operational amplifier.

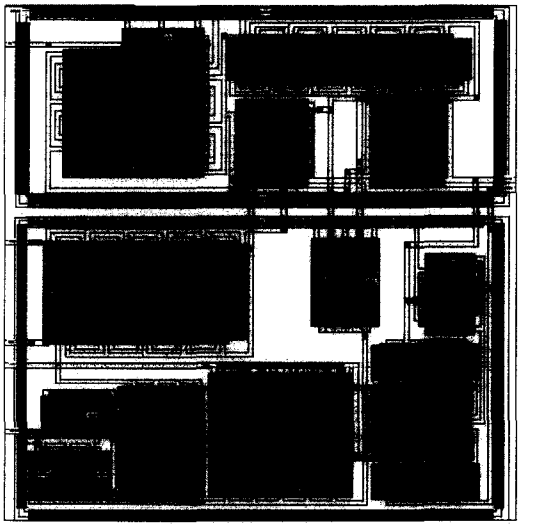


그림 10. 그림 9의 레이아웃
Fig. 10. Layout for Fig. 9.

그림 10의 레이아웃으로부터 추출한 논리적 모델과 RC 추출 모델의 lumped 모델 및 $\pi 3$ 모델로 추출한 결과는 표 1과 같다. 추출된 회로와 초기 회로의 성능을 검증하기 위하여 현대 0.8㎛ CMOS 설계 공정[10]의 HSPICE 레벨 13 BSIM[10]을 사용하여 시뮬레이션한 결과를 그림 12에 표시하였다.

표 2는 그림 13의 초기회로와 설계된 레이아웃의 초기회로와 비교하기 위하여 논리적 모델로 추출한 결과를 교하였다. 초기회로에서 소자의 폭이 25㎛인 경우 λ규칙에 의해 설계된 경우 추출 폭이 24.8㎛의 크기로 설계되었음을 확인할 수 있다. 소자의 이러한 폭의 변

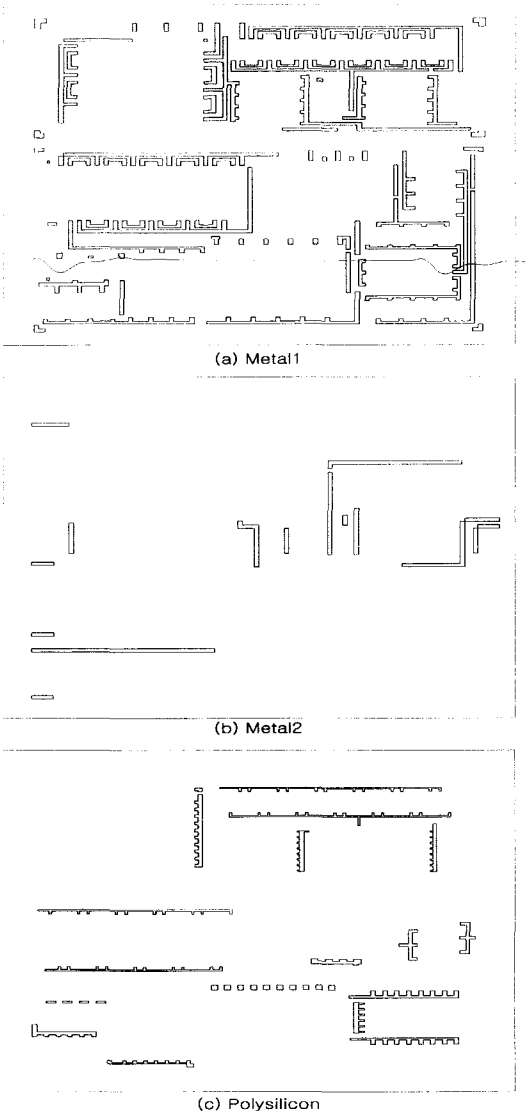


그림 11. 각 레이어 별로 병합된 블록
Fig. 11. Blocks merged for each layer.

화가 회로의 성능에 미소한 변화를 가지게 한다. 논리 회로 모델 추출은 추출된 소자의 개수와 전기적 연결 관계에 있어서 초기회로에 해당하는 레이아웃의 정확성을 판단한다.

표 3은 그림 14의 초기회로와 논리모델, 그리고 pi3 모델로 추출하였을 경우 연산 증폭기의 여러 가지 사양에 대한 시뮬레이션 결과를 나타낸다. 시뮬레이션 결과로부터 논리적 모델의 결과는 소자의 개수가 정확하게 일치하며, 출력 사양은 초기회로의 결과와 큰 차이를 보이지 않는다. 그러나 $\pi 3$ 추출 모델과의 차이를 보면 연산증폭기의 슬루율 (slew rate)이 감소하고, 출력 안정시간이 길어지고 또한 회로의 전력소모가 증가하는 등 상대적으로 모든 특성이 초기회로보다 나빠지게 나타난다.

표 1. 그림 10의 레이아웃으로부터 추출한 결과
Table 1. Elements extracted from layout in Fig. 10.

# of elements	Initial circuit	Extracted model		
		lumped	$\pi 3$	
PMOS	7	39		
NMOS	15	79		
Distributed RC	-	m1	m2	polysilicon
		251	16	232
Element	31	1143	3635	
Node	65	777	1771	

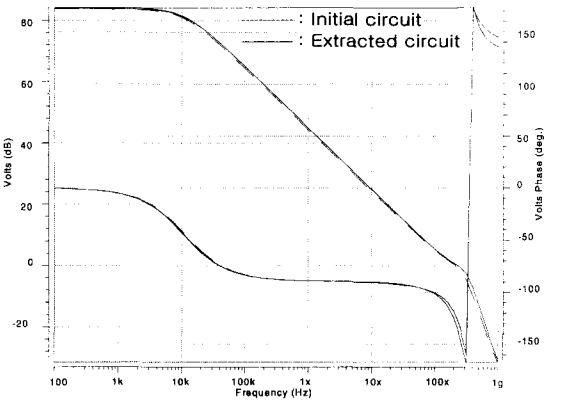


그림 12. 초기 회로와 추출된 회로의 시뮬레이션 결과
Fig. 12. Simulation results of the initial circuit and the extracted circuit.

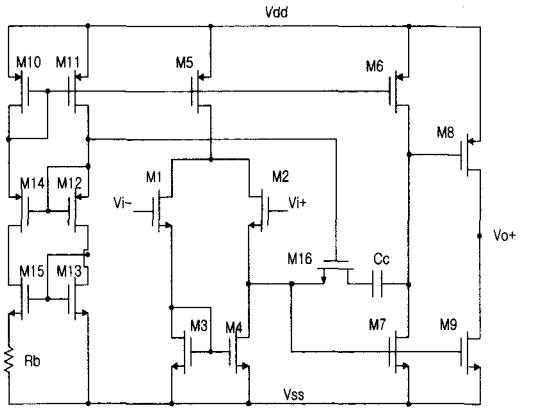


그림 13. 2단 연산증폭기
Fig. 13. Two stage operational amplifier.

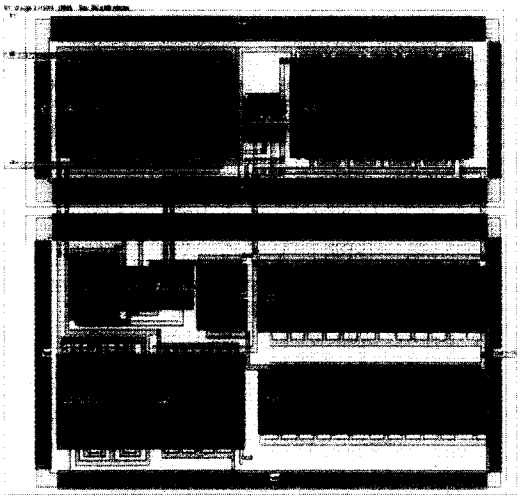


그림 14. 그림 13의 레이아웃
Fig. 14. Layout for Fig. 13.

표 2. 그림 13의 논리 모델 추출
Table 2. Logic model extraction for Fig. 13.

초기 spice 입력 파일	논리 모델 추출
vdd 1 0 dc 2.5v vss 7 0 dc -2.5v vinp 11 0 dc 0v ac 1v ving 9 0 dc 0v Cload 14 0 2pF m10 1 2 2 1 p w=25u l=1.6u m11 3 2 1 1 p w=25u l=1.6u m14 2 3 4 7 n w=25u l=1.6u m12 3 3 5 7 n w=25u l=1.6u m15 4 5 6 7 n w=100u l=1.6u m13 5 5 7 7 n w=25u l=1.6u m5 8 2 1 1 p w=300u l=1.6u m1 10 9 8 1 p w=300u l=1.6u m2 12 11 8 1 p w=300u l=1.6u m3 10 10 7 7 n w=150u l=1.6u m4 12 10 7 7 n w=150u l=1.6u m6 13 2 1 1 p w=300u l=1.6u m8 1 13 14 7 n w=500u l=1.6u m7 13 12 7 7 n w=300u l=1.6u m9 14 12 7 7 n w=500u l=1.6u m16 15 3 12 7 n w=100u l=1.6u Cc 15 13 5pF Rb 6 7 8k .end	vin- 338 0 0v vin+ 359 0 dc 0v ac 1v vdd! 61 0 2.5v vss! 57 0 -2.5v rb 42 57 8k cload 62 0 2p cc 46 47 5p * begin ---------- M1 43 43 57 57 n L=1.6u W=24.8u M2 49 43 42 57 n L=1.6u W=100.0u M3 44 45 49 57 n L=1.6u W=24.8u M4 45 45 43 57 n L=1.6u W=24.8u M5 46 45 66 57 n L=1.6u W=100.0u M6 61 47 62 57 n L=1.6u W=500.0u M7 38 38 57 57 n L=1.6u W=150.0u M8 66 38 57 57 n L=1.6u W=150.0u M9 47 66 57 57 n L=1.6u W=300.0u M10 62 66 57 57 n L=1.6u W=500.0u M11 38 338 14 61 p L=1.6u W=300.0u M12 14 359 66 61 p L=1.6u W=300.0u M13 44 44 61 61 p L=1.6u W=24.8u M14 45 44 61 61 p L=1.6u W=24.8u M15 14 44 61 61 p L=1.6u W=300.0u M16 47 44 61 61 p L=1.6u W=300.0u .end

Ⅳ. 결 론

본 논문에서는 집적회로 레이아웃의 전기적 연결관계와 레이어의 기생 성분을 포함한 회로 정보를 추출하기 위하여 레이아웃을 백터화한 변으로 기하학적인 도형을 블록으로 표현하고 이들 블록에 대하여 분산 RC 모델을 추출하기 위한 블록 분할 알고리즘을 제안하고, 제안한 알고리즘을 적용하여 레이아웃을 검증하는 회로 추출 시스템을 구현하였다.

표 3. 그림 13의 추출 및 시뮬레이션 결과
Table 3. Extraction and simulation results of Fig. 13.

성 능	단위	init	logic	pi3
DC gain	dB	91.47	91.27	91.08
F_t	MHz	25.10	25.11	25.11
ψ	deg	95.9	95.9	87.6
InputCMR	V	-2.5~1.21	-2.5~1.21	-2.5~1.20
Voffset	μV	-19.90	-20.53	-21.10
SR	V/μs	13.50	14.86	12.82
Power	㎽	2.4574	2.4936	2.4865
T_s	μs	92	91	98
OutputSwing	V	-2.5~1.68	-2.5~1.67	-2.5~1.60
CMRR(100Hz)	dB	92.2	92.0	91.9
PSRR+	dB	103.0	102.9	102.4
PSRR-	dB	103.0	102.9	102.4
R_{in}	Ω	1e+20	1e+20	2.1e+15
R_{out}	Ω	289.2	286.2	293.7
V(out)/V(in)	k	38.1	37.2	36.6
시뮬레이션 시간	sec	0.33	0.30	22.68
소자의 갯수	NMOS	10	10	86
	PMOS	6	6	44
	전체	16	16	130
노드의 수	전체	15	15	931

레이아웃 구조가 복잡하더라도 블록 분할에 의해서 중복성 없이 독립된 블록 단위로 등가회로를 추출하고 초기 입력 파일과 연동하여 회로 추출을 행하므로 회로 추출 시간에서 시뮬레이션 시간이 단축된다. 또한 추출 모델을 다양하게 할 수 있도록 하여 레이아웃의 논리적 회로 구성을 확인하거나, 보다 정확한 분산 RC 모델로 추출할 수 있도록 하여 기생 성분이 회로에 미치는 영향을 평가할 수 있도록 하였다. 레이아웃의 논리적 연결 및 비대칭적 구조에 의한 회로의 출력 특성 변화를 시뮬레이션 단계에서 예측할 수 있으므로 보다 신뢰할 수 있는 칩을 제작하기 위한 레이아웃 검증 환경을 제공한다.

참 고 문 헌

[1] A. J. van Genderen and N. P. van der Meijs, "SPACE: A finite element based capacitance

extraction program for sub- micron integrated circuits," NASECODE VI Conference, 1989.

[2] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor, "The Magic VLSI layout system," IEEE DESIGN & TEST, Feb. 1985.

[3] W. S. Scout and J. K. Ousterhout, "Magic's circuit extractor," IEEE DESIGN & TEST, 1986.

[4] Steven P. McCormick, "EXCL : A Circuit Extractor for IC Designs," Proc. of 21st Design Automation Conference, pp. 616-623, 1984.

[5] P. Chapman and K Clark, "The scan line approach to design rules checking : Computational expressions," Proc. of 21st Design Automation Conference, pp. 235-241, 1984.

[6] Erik C. Carlson, "A scanline data structure processor for VLSI geometry checking," IEEE Trans. on CAD, Vol. 6, No. 5, 1987.

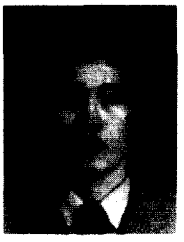
[7] C. Mead and L. Conway, Introduction to VLSI Systems, Addison-Wesley, pp. 115-127, 1980.

[8] HSPICE User's Manual Vol. 2, Elements and Models, HSPICE Version H92, Meta-Software, 1992.

[9] Hyundai Electronics Industries, 0.8㎛ Model Parameter, April 1994

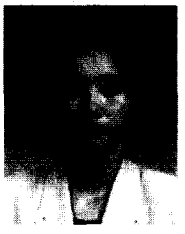
[10] B. J. Sheu, D. L. Scharfetter, P.K. Ko and M.C. Jeng, "BSIM: Berkeley short-channel IGFET model for MOS transistors," IEEE J. Solid-State Circuits, Vol. SC-22, No. 4, pp. 558-566, 1987.

저 자 소 개



孫 永 燦(正會員)

1965년 2월 15일생. 1991년 경북대학교 전자공학과 공학사. 1994년 경북대학교 전자공학과 공학석사. 1996년 경북대학교 전자공학과 박사과정 수료. 1998년 3월~현재 포항1대학 컴퓨터응용과 전임강사. 관심분야 : 집적회로 설계검증, 설계자동화



朱 利 亞(正會員)

1968년 1월 13일생. 1990년 영남대학교 전자공학과 공학사. 1996년 영남대학교 전자공학과 공학석사. 1996년~현재 경북대학교 전자공학과 박사과정. 관심분야 : CAD for VLSI, 아날로그/디지털 집적회로 설계

劉 尙 大(正會員)

1958년 2월 12일생. 1980년 경북대학교 전자공학과 공학사. 1982년 한국과학기술원 전기 및 전자공학과 공학석사. 1998년 한국과학기술원 전기 및 전자공학과 공학박사. 1982년~현재 경북대학교 전자전기공학부 교수. 주관심분야 : 회로설계, 집적시스템, VLSI CAD, 표면음파필터