論文92-29A-1-7

임의의 각도를 갖는 VLSI 레이아웃에서의 회로 및 심볼릭 추출

(Circuit and Symbolic Extraction from VLSI Layouts of Arbitrary Shape)

文 寅 鎬*, 李 容 在**, 黃 善 泳*

(In Ho Moon, Yong Jae Lee, and Sun Young Hwang)

要 約

본 논문에서는 임의의 각도를 가지는 계층적인 레이아웃으로부터의 회로 및 심볼릭 추출을 수행하는 레이아웃 시스템의 구현에 관해 서술하고자 한다. 회로 및 심볼릭 추출기는 테크놀리지 화일에 추출 규칙을 사용자가 기술할 수 있게 함으로써, MOS 또는 바이폴라의 여러가지 공정의 레이아웃을 처리할 수 있도록 하였다. 구현된 시스템에 제안한 알고리듬, 즉 심볼릭 추출 과정의 다각형의 사다리꼴 분해 알고리듬및 사다리꼴의 template을 이용한 심볼릭 경로 추출의 새로운 효율적인 알고리듬을 제안한다. 본 회로 및 심볼릭 추출기는 서강 레이아웃 디자인 및 검증 시스템인 SOLID의 종합 환경을 구축하기 위한 일환으로서 개발되었다.

Abstract

This paper presents the design of a layout processing system that performs circuit and symbolic extraction from hierarchical designs containing arbitrarily shaped layout. The system is flexible enough to deal with various technologies, MOS or bipolar, by providing extraction rules in the form of technology files. In this paper, new efficient algorithms for trapezoidal decomposition of polygon and symbolic path extraction using trapezoidal template are proposed for symbolic extraction. Circuit and symbolic extractor is developed as an integrated design environment of SOLID system.

I. 서 론

최근 CAD 툴의 발달에 따라 VLSI 레이아웃 설계시 컴퓨터를 이용하여 자동으로 레이아웃을 생성시키는 반주문형 방식이 널리 사용되고 있다. 그러나, 대부분의 자동 레이아웃 생성 시스템들이 rectilinear 레이아웃 만을 생성하기 때문에 사용 면적의 비효율성을 감수해야 하며, 특히 선폭이 일정치 않고 많은 굴곡을 가지는 고성능의 아날로그 회로의 레이아웃 생성에는 많은 어려움을 가지고 있다. 따라서, 45°등의 임의의 각도를 사용하여 레이아웃의 사용 면적의 효율성을 증가시키고, 고성능의 회로를 설계함으로써 경쟁력 있는 레이아웃을 생성하기 위하여 full-custom 방식의 레이아웃 기법은 여전히 설계의

*正會員, **準會員, 西江大學校 電子工學科
(Dept. of Elec. Eng., Sogang Univ.)

接受日字 : 1991年 8月 1日

중요한 위치를 차지하고 있다.

반주문형 방식으로 설계된 레이아웃의 경우 다른 설계 규칙을 가지는 새로운 테크놀리지로의 레이아웃 이전을 용이하게 할 수 있으나, 수작업에 의해 설계된 레이아웃의 경우는 레이아웃 이전에 많은 어려움을 가지게 된다. 일반적인 수작업의 레이아웃 이전 과정은 레이아웃으로 부터 스틱 다이아그램의 심볼릭 추출을 수행하여 심볼릭 레이아웃을 발생한 후 레이아웃 컴팩션 과정을 통하여 새로운 테크놀리지로의 레이아웃 이전이 용이함과 동시에 설계자가 레이아웃의 설계 규칙에 대한 정보가 없이도 단순히 스틱 다이아그램을 사용하여 편리하게 설계를 수행할 수 있는 장점을 가지고 있어서 full-custom 방식에 도입되어 최근에 많이 사용되고 있다.

이러한 심볼릭 레이아웃을 지원하기 위하여 최근 많은 심볼릭 추출기들이 발표되었다.[2][4][7] 그러나, 대부분의 심볼릭 추출기들이 rectilinear 패턴만을 처리하고 STICKIZER[2]만이 45°의 패턴을 처리할 수 있으며, 소자 및 컨택의 추출에 있어 시스템 내부에 저장된 모델에 대해서만 추출이 가능하므로 임의의 각도를 가지는 레이아웃 및 다양한 테크놀리지로의 적용에는 많은 문제점을 가지고 있다.

본 논문에서는 이러한 문제점들을 해결하기 위하여 사용자가 기술하는 테크놀리지의 추출 규칙 정보를 입력으로 임의의 각도를 가지는 계층적인 레이아웃으로 부터 회로 및 추출 심볼릭 추출을 수행하기 위한 효율적인 알고리듬 및 이의 구현에 관해 기술하고자 하며, 특히 심볼릭 추출 과정에서 사용되는 다각형의 사다리꼴 분해 알고리듬 및 심볼릭 경로 추출 알고리듬을 제안한다. 회로 추출은 레이아웃으로 부터 회로 및 전기적인 파라미터를 추출하며, 기생 저항의 추출은 심볼릭 추출을 수행한 후 심볼릭 경로상에서 저항값을 구하게 된다. 심볼릭 추출의 경우는 추출 과정에서 레이아웃의 연결선의 굴곡 및 소자의 폭과 길이등이 전기적인 특성을 유지하여야 한다. 임의의 각도를 가지는 레이아웃으로 부터 심볼릭 추출을 수행할 때 다음 단계에서 사용할 레이아웃 컴팩션 시스템이 만일 rectilinear의 심볼릭 레이아웃만을 처리하는 경우는 임의의 각도를 90°로 변형시켜 추출을 수행하게 되지만, 임의의 각도를 처리할 수 있는 레이아웃 컴팩션 시스템[1][5]의 경우는 임의의 각도를 심볼릭 레이아웃에 계속 유지하게 된다. 임의의 각도를 가지는 심볼릭 레이아웃을 90°로 변형시키는 과정은 수직 에지 또는 수평 에지 이외의 임의의 각도를 형성하는 에지에 대하여 이를 수직 에지 또는 수평 에지가 되도록 에지의 버텍스 좌표를 이동시킴에 의해 변형된다. 심볼릭 추출 과정은 노드를 형성하는 각각의 다각형을 사다리꼴로 분해하여 각 사다리꼴에 대하여 사다리꼴의 template을 사용하여 사다리꼴의 중심선을 따라가면서 심볼릭 경로를 발생시키게 된다.

회로 및 심볼릭 추출기는 그림1에 나타나 있는 서강 레이아웃 설계 및 검증 시스템인 SOLID(sogang layout verification and design system)를 구축하기 위한 일환으로서 개발되었으며, 회로 추출과 심볼릭 추출을 동시에 수행하게 되고 테크놀리지 화일에 추출 규칙을 사용자가 기술할 수 있도록 함으로써 MOS 및 바이폴라의 여러가지 공정의 레이아웃을 처리할 수 있도록 하였다.

본 논문의 구성은 Ⅱ장에서 추출을 위한 테크놀리지 화일에 대해 기술하였고, Ⅲ장에서는 회로 추출 과정을, Ⅳ장에서는 심볼릭 추출 과정을 기술하였다. Ⅴ장에서는 구현된 회로 및 심볼릭 추출기의 실험 결과를 기술하였다.

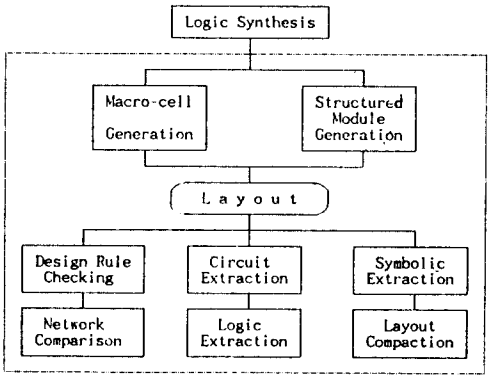


그림 1. 서강 레이아웃 디자인 및 검증 시스템
Fig. 1. Sogang layout design and verification system.

Ⅱ. 테크놀리지 화일

회로 및 심볼릭 추출은 사용자가 작성한 테크놀리지 화일의 추출 규칙에 따라 수행된다. 이 화일은 그림2에서와 같이 입력 레이어 정보, 설계 규칙[7] 그리고 추출 규칙으로 구성된다. 입력 레이어 정보의 R은 단위 저항값을 C는 단위 면적당 용량값을 나타내며, 'layer-type'은 G(general), C(contact), W(well) T(text)중의 하나로 표시하며, 존재하지 않는 경우는 G로 인식하게 된다. 추출 규칙의 앞 부분은 입력

레이어로부터 Boolean operation(AND, OR, MINUS, XOR) 및 sizing을 이용하여 추출에 필요한 새로운 레이어를 발생시키는 규칙을 나타낸다.

```
/* input layer description */

INPUT layer-name layer-number [layer-type]
      R=valuel  C=valus2

/* design rule description */

WIDTH       layer          min_value
SPACE       layer          min_value[N]
MARGIN      layer1 layer2 min_value
CO-MAGIN    layer1 layer2 min_value
SEPARATE    layer1 layer2 min_value
ENCLOSURElayer1 layer2 [min_value]
INTERSECT layer1 layer2 min_value

/* extraction rule description */

AND     in-layer1 in-layer2 out-layer
OR      in-layer1 in-layer2 out-layer
MINUS   in-layer1 in-layer2 out-layer
XOR     in-layer1 in-layer2 out-layer
SIZE    in-layer [out-layer} BY value

ATTACH text-layer TO layer
CONNECT layer1 TO layer2 [BY contact-layer]
DEVICE model-name type device-layer term-layer1
       term-layer2 [term-layer3 ···]
```

그림 2. 테크놀리지 화일의 syntax
Fig. 2. Syntax of technology file.

"ATTACH"는 레이아웃상의 'text-layer'의 text 데이타와 이를 포함하는 'layer'의 데이타를 연결하며, VDD 및 GROUND 노드의 지정, 그리고 특정 노드의 이름을 부여하는데 사용된다. "CONNECT"는 두 레이어 또는 세 레이어가 겹칠때의 전기적인 연결을 나타낸다. "DEVICE"는 레이아웃상에서 추출하고자 하는 소자들을 나타내며, 'type'은 PMOS, NMOS, NPN, PNP, DIODE, CAPACITOR, RESISTOR 등의 소자 타입을 지정하고, 'device-layer'는 소자를 나타내는 레이어를, 'term-layer'는 소자의 각 단자를 나타내는 레이어를 의미한다. 'CONNECT' 및 'DEVICE' 내에 사용되는 레이어들 중 'contact-layer'와 'device-layer'이외의 모든 레이어를 'conduct-layer'로 정의하며, 최종적으로 추출과정에서는 이 세가지의 레이어만이 사용된다. 사용자가 입력하는 테크놀리지 화일을 사용함으로써 모든 공정에 대한 회로 및 심볼릭 추출이 가능하다.

Ⅲ. 회로 추출

회로의 추출은 자동으로 생성된 레이아웃 또는 full-custom 방식의 레이아웃으로 부터 회로 및 전기적인 파라미터를 추출함으로써 최종 레이아웃의 회로와 설계자가 의도한 회로가 일치하는지를 검증하는데 사용될 수 있으며, 전기적인 파라미터를 이용하여 회로의 성능이 주어진 조건에 만족하는지를 재 시뮬레이션하기 위한 back-annotation에 사용된다. 이는 게이트 레벨의 로직 및 행위 추출을 위한 입력으로도 사용될 수 있다.[9] 회로의 추출 과정은 레이어 발생 및 노드 병합, 그리고 RC 추출의 세 단계로 구성된다. 회로 추출의 출력으로는 SPICE 입력 화일 포맷의 회로 정보를 발생시킨다.

1. 레이어 발생

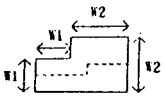
테크놀리지 화일의 추출 규칙에 기술된 Boolean operation 및 resizing의 순서에 따라 소자 및 연결 정보의 추출에 필요한 새로운 레이어들을 차례로 발생시키게 된다. 따라서, 소자 및 컨택의 추출은 레이어 발생을 통하여 자동적으로 수행된다. 레이어 발생을 위한 자료 구조로는 polygon-based quad-tree[8]를 사용하고 있으며, siliced-edge trace 알고리듬[8]을 사용하여 Boolean operation 및 sizing을 이용한 레이어 발생을 수행한다.

2. 연결 정보 추출

소자의 추출이 완료된 후 각 소자의 단자간의 연결 정보를 추출하게 된다. 연결 정보의 추출 과정은 노드 병합과 각 소자의 단자 노드 지정의 두 단계로 진행 된다. 노드 병합은 크게 네가지의 과정으로 이루어진다. 첫번째 단계에서는 각 다각형마다 고유 노드 번호를 지정하고 다음 단계에서, 각 'conduct-layer'에 대해 겹치는 다각형들간의 노드 병합 과정을 거쳐 'CONNECT'로 지정된 레이어간의 노드 병합 과정을 수행한다. 'CONNECT'를 통한 노드 병합은 서로 다른 레이어에 대해 전기적으로 연결되는 경우 이들의 레이어를 동일 노드로 인식하기 위하여 사용되며, 단순히 서로 다른 두 레이어간의 노드 병합 및 'contact-layer'를 통한 세 레이어간의 노드 병합이 이루어 진다. 마지막으로, 노드 병합과정의 내용을 저장하고 있는 노드 병합 테이블을 사용하여 최종적으로 존재하는 노드들의 reordering을 수행한다. 노드 병합 과정이 완료된 후, 각 소자의 단자에 연결된 노드를 지정함으로써 연결 정보의 추출이 완료된다.

3. RC 추출

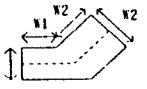
레이아웃으로 부터 기생 저항과 기생 용량의 추출은 테크놀리지 화일로부터 단위 저항값과 단위 면적당 용량값을 읽어들여 각각의 노드에 대해 계산을 수행하게 된다. 기생 용량값은 노드를 형성하는 다각형의 면적을 계산하여 단위 면적당의 용량값을 곱하여 구하게 된다. 보다 정확한 값을 필요로할 때에는 fringing field와 coupling에 의한 용량값까지 고려하게 된다. 기생 저항값은 단순히 면적의 함수가 아니므로 노드를 형성하는 다각형으로 부터 저항값을 직접 구하기 위해서는 많은 시간과 어려움이 따르기 때문에, 대부분의 추출 시스템에서 다각형을 사각형 또는 사다리꼴로 분해를 수행한 후 각각의 분해된 형태에 해당하는 저항값을 구하고 전체적으로 모두 더함으로써 근사적으로 구하게 된다[3]. SOLID의 회로 추출기에서는 그림3에서와 같이 세가지의 패턴 및 폭의 ratio(W_2/W_1)의 변화에 따른 저항값을 테이블로 미리 만들어 놓고, 심볼릭 추출 과정을 통하여 발생되는 각각의 노드에 대한 심볼릭 경로를 이용하게 된다. 심볼릭 경로를 이용하여 저항을 구하는 과정은 그림4에서와 같이 경로의 시작점으로 부터 끝점까지 따라가면서 경로의 폭이 변하거나 bending이 생기는 경우 테이블을 참조하여 해당되는 저항값을 구하게 되고, 그 외의 직선 부분들에 대해서는 경로의 폭 및 길이를 고려하여 쉽게 저항값을 구하게 되며, 이들을 모두 더함으로써 기생 저항값을 구하게 된다. 테이블을 참조하는 과정에서 폭의 ratio가 등록되어 있지 않은 경우는 interpolation 방식에 의해 근사 저항값을 사용하게 된다.



(a)

ratio(W_2/W_1)	1	1.5	2	3
resistance	2.0	2.1	2.25	2.5

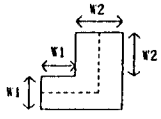
(b)



(c)

ratio(W_2/W_1)	1	1.5	2	3
resistance	2.2	2.3	2.4	2.6

(d)



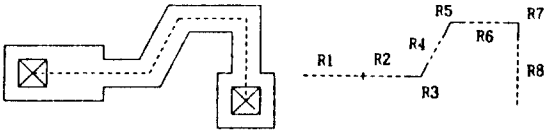
(e)

ratio(W_2/W_1)	1	1.5	2	3
resistance	2.5	2.55	2.6	2.75

(f)

그림 3. 패턴의 ratio별 저항값 테이블
(a) 패턴 A (b) 패턴 B (c) 패턴 C
(d) 패턴 A의 ratio별 저항값 테이블
(e) 패턴 B의 ratio별 저항값 테이블
(f) 패턴 C의 ratio별 저항값 테이블

Fig. 3. Resistance table according to ratio for patterns.
(a) pattern A, (b) pattern B,
(c) pattern C,
(d) resistance table according to ratio for pattern A,
(e) resistance table according to ratio for pattern B,
(f) resistance table according to ratio for pattern C.



(a)　　(b)

그림 4. 심볼릭 경로를 이용한 저항값 추출의 예
(a) 노드 및 심볼릭 경로의 예
(b) 저항값 추출

Fig. 4. An example of resistance extraction using symbolic path.
(a) an example of node and symbolic path,
(b) resistance extraction.

Ⅳ. 심볼릭 추출

심볼릭 추출은 레이아웃으로 부터 스틱 다이아그램을 추출함으로써, 다른 테크놀리지로의 레이아웃 이전 (technology migration)을 위해 주로 사용되며, 레이아웃 컴팩션의 전처리 과정으로 사용된다[4][7]. 스틱 다이아그램은 각 회로 소자들의 크기 및 상대적인 위치와 각 노드의 심볼릭 경로 및 폭의 정보로 구성된다. 심볼릭 추출에서 가장 중요한 위치를 차지하고 있는 심볼릭 경로 추출의 경우 현재까지 가장 널리 사용되는 알고리듬으로서는 thinning method [2], Lee router[4]와 패턴 router[7]가 있다. 패턴 router의 경우 다각형을 사각형으로 분해하면서 모든 사각형들이 준비된 wire template에 속할때까지 분해한 후 심볼릭 경로를 찾아나간다. SOLID의 심볼릭 추출기에서는 임의의 각도를 가지는 레이아웃까지 처리하기 위하여 일반적인 패턴 router의 사각형 분해 및 사각형 wire template을 사용하는 대신 이를

확장하여 사다리꼴 분해 및 사다리꼴 wire template을 사용한다. 즉, 다각형을 사다리꼴로 모두 분해한 후, 각각의 사다리꼴에 대해 차례대로 사다리꼴의 wire template을 이용하여 심볼릭 경로를 찾아내는 새로운 알고리듬을 제안한다.

심볼릭 추출 과정은 다각형의 병합과 다각형의 사다리꼴 분해, 그리고 심볼릭 경로 추출의 세 단계로 구성된다. 첫 단계는 회로 추출시 발생된 노드 병합 테이블을 이용하여 각각의 노드에 대해 각 interconnection 레이어 (metal, poly, diffusion등)의 overlap된 다각형들을 병합한다. 둘째 단계는 각각의 병합된 다각형을 사다리꼴로 분해하고, 마지막으로 lower-left로 부터 upper-right의 방향으로 각각의 사다리꼴의 중심선을 따라 심볼릭 경로를 찾아낸다.

1. 다각형의 사다리꼴 분해

다각형을 사다리꼴로 분해하기 위해서는 다각형의 n개의 버텍스들을 lower left 버텍스로 부터 반시계 방향으로 0에서 (n−1)까지의 버텍스 번호를 지정한 후 그림5에서와 같이 Y축 및 X축으로 버텍스의 소팅을 미리 수행한 후 YX의 버텍스 연결 리스트를 생성한다. 분해 과정에서는 그림6에서와 같이 Y축의 방향으로 진행이 되며, 반복적으로 그림6에서와 같이 5개의 버텍스를 찾아낸 후 사다리꼴을 출력하게 된다. YX의 버텍스 연결 리스트는 계속 변형, 생성 또

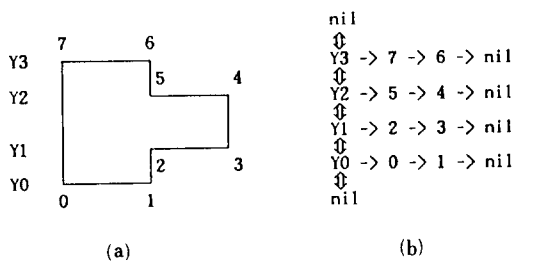


그림 5. 다각형 버텍스 소팅
(a) 다각형의 예
(b) 소팅된 버텍스 연결 리스트

Fig. 5. Vertex sorting of polygon.
(a) an example polygon,
(b) linked-list of sorted vertices.

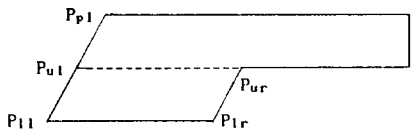


그림 6. 사다리꼴 분해를 위한 5개 버텍스의 지정

Fig. 6. Determination of 5 vertices for decomposition.

```
Decomposed_Polygon_into_Trapezoid(polygon)
POLYGON *polygon;
{
  L=make_yx_linked_list_of_vertices(polygon);/L:vertex_
    list */
  while(vertex_list L!=NULL){
    Lowest_Y=the set of lowest vertices;
    Pll=the leftmost vertex in Lowest_Y;
    Plr=the next leftmost vertex in Lowest_Y;
    Ppl=the previous vertex of Pll;  /*Ppl->vertex=Pll-
        >vertex-1*/
    Pul=the lower-left vertex in vertex_list L, satisfying
        as follows;
        Yul>Yll, and min(Xll, Xpl)<=Xul<=Xlr;
    Pur=the lower-right vertex in vertex_list L, satisfying
        as follows;
        Yur>Yll, and Xul<=Xur<=max(Xlr, Xnr);
    if (Pul is a point on edge Epl){
      Ppre_of_ul=the previous vertex of Pul;
      /*Ppre_of_ul->vertex=Pul->vertex-1*/
      if (Yul==Ypre_of_ul)
        Remove Pul from vertex_list L;
    }
    else{
      Add new_vertex to edge Epl at the intersecting point
        by the line Y=Yul;
      Insert new_vertex into vertex_list L;
      Pnl=new_vertex;
    }
    if (Pur is a point on edge Elr){
      Pnext_of_ur=the next vertex of Pur;
      /*Pnext_of_ur->vertex=Pur->vertex+1*/
      if (Yur==Ynext_of_ur)
        Remove Pur from vertex_list L;
    }
    else{
      Add new_vertex to edge Elr at the intersecting point
        by the line Y=Yur;
      Insert new_vertex into vertex_list L;
      Pur=new_vertex;
    }
    Remove the vertices(Pll, Plr);
    Output one trapezoid;  /*using Pll, Plr, Pul, and Pur*/
  }
}
```

그림 7. 다각형의 사다리꼴 분해 알고리듬

Fig. 7. Trapezoidal decomposition algorithm.

는 삭제되면서, 이 연결 리스트가 NULL이 될때까지 반복적으로 수행된다. 그림7은 다각형의 사다리꼴 분해를 수행하는 알고리듬을 나타낸다. $X_{11}(Y_{11})$은 P_{11}의 X(Y) 좌표를 나타내며, E_{11}은 P_{11}로 부터 다음 버텍스 (즉, P_{11}—>vertex+1의 버텍스 번호를 가지는 버텍스)까지의 에지를 나타낸다. 그림8은 그림 7의 알고리듬을 이용하여 다각형을 사다리꼴로 분해한 경우의 예를 보여준다. 다각형의 사다리꼴 분해 알고리듬은 O(n)의 시간 복잡도를 가지고 수행된다. 이는 분해 과정에 버텍스의 sorting 과정이 포함되어 있기는 하지만 버텍스의 순서가 random한 것이 아니라 다각형을 형성하는 반시계 방향의 버텍스 순서를 가지기 때문에 O(n)의 sorting을 수행할 수 있다.

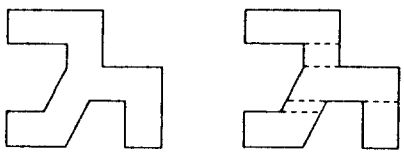


그림 8. 다각형 사다리꼴 분해 예
(a) 다각형의 예
(b) 사다리꼴로 분해된 결과

Fig. 8. An example of trapezoidal decomposition.
(a) an example of polygon,
(b) result of trapezoidal decomposition.

2. 심볼릭 경로 추출 알고리듬

다각형으로 부터 심볼릭 경로를 추출하는 과정은 다각형의 사다리꼴 분해를 수행한 후 lower-left에서 upper-right 방향으로 차례대로 각각의 사다리꼴의 중심선을 따라 진행함으로써 다각형의 심볼릭 경로를 찾게 된다. 사다리꼴의 중심선을 따라가는 과정은 그림9에서와 같이 사다리꼴의 심볼릭 경로를 위한 template를 사용하여 진행하게 된다. 사다리꼴의 위와 아래 두 선폭 중 작은 선폭을 기준으로 컨택의 확장 규칙 (extension rule)을 만족하는 최소 선폭 이하인 경우는 그림9(a)의 template을 적용하게 되고, 그 이상인 경우는 왼쪽을 그림9(b)의 template중의 하나를 오른쪽을 그림9(c)의 template중의 하나의 조합으로 심볼릭 경로를 찾아가게 된다.

현재 처리중인 사다리꼴의 심볼릭 경로로부터 처리되지 않은 사다리꼴로의 심볼릭 경로가 존재할 가능성이 있을 때 연결이 일어날 지점을 'active_point'로 정의한다. 그림10에서와 같이 현재 처리중인 사다리꼴의 바로 윗 부분에 다른 사다리꼴이 존재할때 두 사다리꼴의 X축 방향의 겹치는 폭이 설계 규칙의 최소 선폭인 경우 또는 컨택의 확장 규칙을 만족하는 최소 선폭이내의 경우는 하나의 active_point가 존재하게 되며, 그림10(b)에서와 같이 겹치는 선폭이 그 이상인 경우는 두개의 active_point가 존재하게 된다.

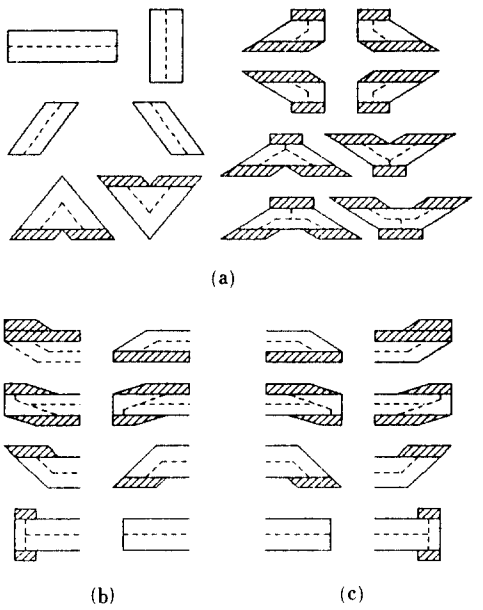


그림 9. 사다리꼴의 심볼릭 경로 templates.
(a) 바로 적용 가능한 templates
(b) 왼쪽 끝을 위한 templates
(c) 오른쪽 끝을 위한 templates

Fig. 9. Templates for symbolic path of trapezoid.
(a) immediately applicable templates,
(b) templates for left-end,
(c) templates for right-end.

심볼릭 경로를 찾아가는 과정에서 그림11에서와 같이 심볼릭 경로에 대응하는 방향성 그래프를 동시에 형성하게 된다. 그래프에서 하나의 전기적인 노드에 대해 컨택 및 소자의 터미널과 연결되는 점, 그리고 bending이 일어나는 점이 그래프의 노드가 되며, 그래프 노드들간의 연결 관계가 그래프의 에지가 되고, 각 에지는 해당 wire의 폭을 웨이트로 가지게 된다. 그래프의 노드는 GENERAL_NODE,BRANCH_NODE, CONTACT_NODE, TERMINAL_NODE의 네가지 타입중의 하나를 가지게 된다. GENERAL_NODE는 레이아웃의 bending을 나타내는 노드들 중 fanin과 fanout이 각각 하나인 경우, BRANCH_NODE는 fanin 또는 fanout이 두개 이상인 경우를 나타내

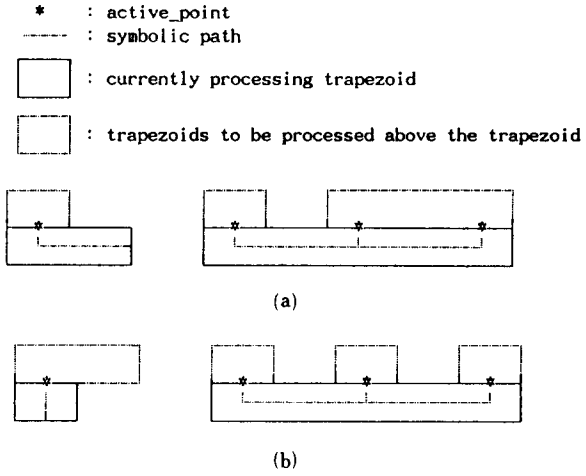


그림10. Active_point의 예
Fig. 10. Examples of active_point.

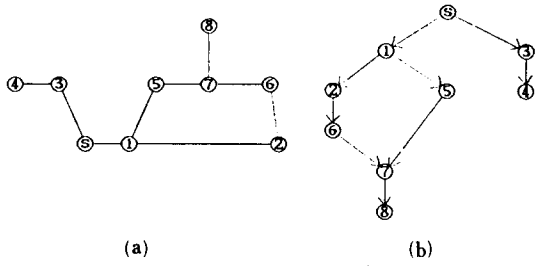


그림11. 심볼릭 경로의 방향성 그래프
(a) 심볼릭 경로 (b) (a)의 방향성 그래프
Fig. 11. An example of directed graph for symbolic path.
(a) symbolic path,
(b) directed graph of (a).

며, CONTACT_NODE는 컨택이 존재하는 위치의 노드, TERMINAL_NODE는 소자의 터미날이 존재하는 위치의 노드를 나타낸다. 생성된 그래프는 모든 사다리꼴에 대한 심볼릭 경로를 찾아낸 후 심볼릭 경로의 최적화 과정에 사용된다.

사다리꼴 wire templated와 active_point를 이용하여 다각형으로 부터 심볼릭 경로를 추출하는 과정은 그림12에서와 같이 각 사다리꼴에 대해 크게 세가지의 단계로 진행된다. 각각의 사다리꼴에 대해 먼저 그림12(a)에서와 같이 해당 template을 선정하여 중심선을 따라 사다리꼴 자체의 심볼릭 경로를 찾아내고 심볼릭 경로의 폭을 계산한다. 다음 단계로 그림

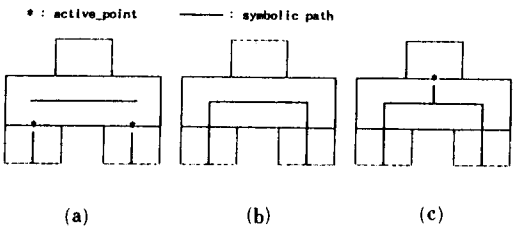


그림12. 각 사다리꼴에 대한 심볼릭 경로의 추출 과정
(a) template의 선정 및 사다리꼴의 심볼릭 경로 추출
(b) active_point의 확장
(c) 새로운 active_point의 발생
Fig. 12. Extraction process of symbolic path for each trapezoid.
(a) template selection and symbolic path extraction of trapezoid,
(b) extension of active_point,
(c) generation of new active_point.

12(b)에서와 같이 사다리꼴의 하부 X영역 아래에 active_point들이 존재하는 경우 각각의 active_point를 포함하는 심볼릭 경로를 사다리꼴의 심볼릭 경로까지 Y방향으로 확장함과 동시에 active_point를 두 심볼릭 경로가 만나는 점으로 이동시킨 후, 더이상 active_point가 되지 않을 때는 이를 제거시킨다. 마지막 단계로 그림12(c)에서와 같이 사다리꼴의 상부 X영역 위에 존재하는 사다리꼴들에 대해 새로운 active_point들을 심볼릭 경로상에 추가 시키게 된다. 그림13에 이를 수행하는 심볼릭 경로 추출 알고리듬을 pseudo C 언어로 나타내었고, 그림14에서는 이 알고리듬을 이용한 심볼릭 경로 추출의 예를 보였다. 알고리듬에서 프로시져 Optimize_Symbolic_Path()는 Ⅳ.4의 심볼릭 경로에 대한 최적화 과정을 수행한다. 이 알고리듬은 하나의 다각형으로부터 분해된 사다리꼴의 수를 n이라할 때 O(n)의 시간 복잡도를 가지고 수행된다.

3. 심볼릭 경로의 최적화

모든 사다리꼴에 대한 심볼릭 경로의 추출이 완료된 후, 추출된 심볼릭 경로에 대한 최적화 과정을 방향성 그래프상에서 진행하게 된다. 이 최적화 과정에서는 형성된 방향성 그래프에 대해 다각형 내의 컨택 및 소자의 핀들에 대한 연결 정보를 사용하여 심볼릭 경로상에 이들의 위치에 노드가 존재하면 노드의 타입을 수정하게 되고 존재하지 않으면 새로운

```
Find_Symbolic_Path(trapezoids, cnt_dev)
TRAPEZOID *trapzoids; /*linked-list of decomposed
                        trapezoids
                          from lower-left to upper-right*/
CNT_DEV  *cnt_dev;  /*linked-list of contacts and devices
                        inside the polygon*/
{
 width=minimum wire width of layer;
 extension_width=minimum wire width of contact+
                 2 *minimum extension distance from
                 contact;
 for (each trapezoid) {
   Determine a template;
   Generate a symbolic path along center line using the
   template and evaluate the width;
   for(each active_point between X_ll and X_lr) {
     Extend the symbolic path from active_point to center
     line;
     if (no trapezoid above current active_point)
       Remove active_point;
     else
       Move active_point to center line;
     }
     for(each trapezoid T above current trapezoid
     between X_ul and X_ur)
       if(active_point exists below T) {
         y2=lower Y of T;
         Add new symbolic path from active_point to y2;
         Move active_point to y2;
       }
       else {
         x1=max(lower-left X of upper trapezoid,
                lower-left X of current trapezoid);
         x2=min(lower-right X of upper trapezoid,
                lower-right X of currnet trapezoid);
         y1=Y of center line;
         y2=upper Y of current trapezoid;
         if((x2-x1)>=extension_width) {
           x1=(x1+x2)/2;
           Add new symbolic path from (x1, y1) to(x1, y2);
           Make new active_point at (x1, y2);
         }
         else{
           x1+=width/2; x2-=width/2;
           Add new symbolic path from(x1, y1) to (x1, y2)
             and from (x2, y1) to (x2, y2);
           Make new active_point at (x1, y2) and (x2, y2);
         }
       }
   }
 }
 /*After the symbolic path extraction, a Directed_Graph
   is estaished. */
 Optimize_Symbolic_Path(Directed_Graph, cnt_dev);
}
```

그림13. 심볼릭 경로 추출 알고리듬
Fig. 13. Algorithm of symbolic path extraction.

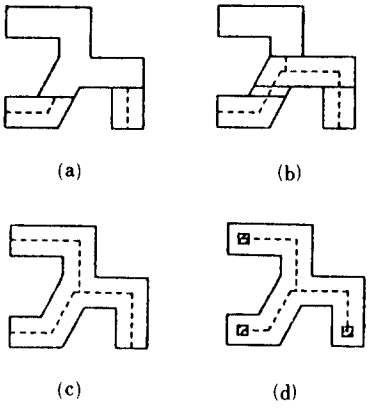


그림14. 심볼릭 경로 추출의 예
(a) 첫째와 둘째 사다리꼴 수행 후
(b) 셋째와 넷째 사다리꼴 수행 후
(c) 완성된 심볼릭 경로
(d) 컨택을 고려한 심볼릭 경로

Fig. 14. An example of symbolic path extraction.
(a) after processing 1st and 2nd trapezoids,
(b) after processing 3rd and 4th trapezoids,
(c) completed symbolic path,
(d) symbolic path extraction with contact.

노드를 심볼릭 경로상에 추가하며, 이들의 위치가 심볼릭 경로상에서 벗어난 경우는 새로운 노드와 에지를 추가하게 된다. 컨택의 경우 해당 레이어의 확장 규칙에 해당하는 거리내의 노드들중 하나의 fanin 노드 또는 하나의 fanout 노드만을 가지는 GENERAL_NODE 타입의 노드들을 제거하게 된다. 또한, 그림15(a)에서와 같이 오목 또는 볼록한 형태의 bending이 존재하는 부분의 노드 및 에지들에 대해 bending을 제거하더라도 에지를 확장시켰을 때 동일한 형태의 레이아웃을 얻을 수 있는 경우는 방향성 그래프상에서 해당 노드 및 에지들을 제거 또는 위치를 수정하게 된다. 그리고, 그림15(b)와 15(c)에서와 같이 방향성 그래프내에서 feedback loop 또는 reconvergent 경로상에서 각 에지의 폭을 확장시켰을 때 겹치거나 abut되는 경우, 이에 관련되는 두개 이상의 경로에 포함되는 노드 및 에지들을 하나의 경로로 병합시키는 기능을 수행한다.

4. 심볼릭 추출

심볼릭 추출은 회로 추출 과정의 레이어 발생정보 및 노드 병합 테이블을 입력으로 하여 진행된다. 추출은 각 노드의 각 'conduct_layer'에 대해 노드 병합 테이블을 이용하여 다각형의 병합을 수행한 후, 각각의 병합된 다각형에 대해 다각형의 내부에 존재

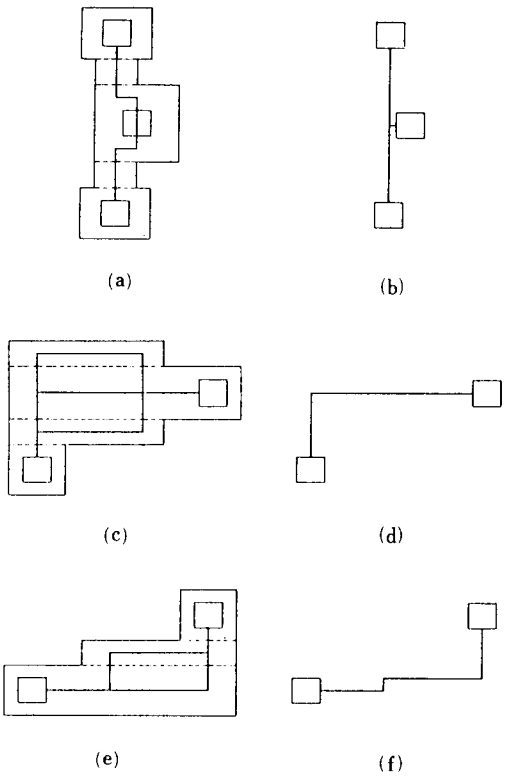


그림15. 심볼릭 경로의 최적화 예
(a)(b)(c) 심볼릭 경로의 예
(d)(e)(f) (a)(b)(c)에 대한 최적화 과정후의 심볼릭 경로

Fig. 15. Examples of optimization for symbolic path.
(a)(b)(c) examples of symbolic path,
(d)(e)(f) optimized symboltc path from (a)(b)(c).

하는 컨택 및 병합된 다각형에 연결된 소자들을 찾아낸다. 그 후, 병합된 다각형을 사다리꼴로 분해한 후, 사다리꼴의 연결 리스트 및 병합된 다각형이 포함하는 컨택 및 소자들의 정보를 이용하여 심볼릭 경로를 찾아낸다. 그림16에 심볼릭 추출의 전반적인 과정을 pseudo C 언어로 나타내었다. 심볼릭 추출 알고리듬에서 프로시져 Circuit_Extraction()은 회로 추출을 수행하며, Merge_Polygon()은 전기적으로 연결된 여러개의 다각형들을 하나의 다각형으로 병합시키는 기능을 수행하고,[8] Calculate_RC()는 각각의 노드에 대해 심볼릭 경로 추출 과정에서 생성된 방향성 그래프를 사용하여 기생 저항 및 기생 용량을 계산한다.

```
Symbolic_Extraction()
{
  Node=Circuit_Extraction();
  for (each node n in Node) {
    for (each 'conduct_layer'){
      L=Merge_Polygons(); /*L:set of marged polygons*/
      for (each merged polygon P in L) {
        C=Find contacts and devices on the polygon P;
        T=Decompose Polygon_into_Trapezoid(P);
        /*T:linked-list of decomposed trapezoids*/
        Find_Symbolic_Path(T, C);
      }
    }
    Calculate_RC(G);/*G:Dircted_Graph for generated
                symbolic path */
  }
}
```

그림16. 심볼릭 추출 알고리듬
Fig. 16. Symbolic extraction algorithm.

5. 계층적 추출

회로 및 심볼릭의 계층적 추출을 수행하기 위해 셀들의 계층적 관계를 트리 그래프 형태로 변형한 후, root 노드의 셀들로부터 leaf 노드의 셀까지 depth-first 탐색으로 추출을 수행한다. 계층적 추출을 위해서는 root 노드로부터 각 parent 노드의 셀에서 instantiation된 child 노드의 셀에 대한 변환(transformation) 메트릭스 정보의 계층적 연산을 수행하여야 하며, 각 instantiation된 child 노드의 셀과 parent 노드의 셀간의 동일 노드들을 찾아 계층적 netlist를 구성하여야 한다. 계층적 netlist를 구성하기 위해서 각 parent 노드의 셀에서 instantiation된 child 노드의 셀의 bounding box에 overlap 또는 abut된 노드들을 찾아 child 노드의 셀에 전달한다. Child 노드의 셀에서는 추출을 완료한 후, 전달된 노드들의 다각형 또는 심볼릭 경로의 좌표들을 자신의 좌표계로 변환한 후, 이 노드들에 연결되는 child 노드의 셀의 노드들을 parent 노드의 셀로 전달한다. 이 과정을 recursive하게 root 노드로부터 leaf 노드까지 수행함으로써 계층적 netlist를 얻게 된다. 이 과정에서 전달된 노드의 다각형에 의해 새로운 소자가 child셀에서 생성되는 경우는 child 셀을 또 하나의 새로운 셀로 인식하여 전체 레이아웃의 셀 리스트에 등록하게 된다.[6] 또한, instantiation된 셀들간의 overlap된 영역들에 대해서는 그 영역내의 다각형들을 flattening 함으로써 잠정적인 instantantion 셀로 만들어 이 셀에 대해 추출을 수행한 후 새로운 소자가 발생하는 경우

는 마찬가지로 이 셀을 전체 레이아웃의 셀 리스트에 등록하게 된다.[6] 그림17에 계층적 추출의 개략적인 알고리듬을 pseudo C 언어로 나타내었다.

```
Hierarchical_Extraction(cell, transported_polygons, matched_
  nodes)
CELL  *cell;
POLYGON  *transpoted_polygons;
int  matched_nodes[];
{
  CELL  *subell, *tmp_cell;
  POLYGON  *transporting_polygons;

  if (cell is not yet extracted)
    Circuit_and_Symbolic_Extraction(cell);
  if (flag=Check if new device is generated by transported_
     polygons() )
    Make new cell and add the new cell to cell list() ;
  for (subcell=each instantiated cell) {
    for (each overlapped area among instantiated cells) {
      tmp_cell=Make a temporary cell by flattening all
               polygons inside the overlapped area();
      Circuit_and_Symbolic_Extraction(tmp_cell);
      if (new device is generated)
        Add tmp_cell to cell list();
    }
    transporting_polygons=Find polygons abutted or
                          overlapped with bounding box of
                          subcell();
    Transform coordinate of transporting_polygons into
      subcell's coordinate();
    Hierarchical_Extraction(subcell, transporting_polygons,
      matched_nodes);
    Build connectivity of hierarchical netlist matched_nodes()
  }
  Find matched_nodes for each transported_polygons();
}
```

그림17. 계층적 추출 알고리듬
Fig. 17. Hierarchical extraction algorithm.

```
INPUT pwell 1 W
INPUT diff 2
INPUT poly 3
INPUT cnt 4 C
INPUT metal 5
INPUT text 6 T

WIDTH diff      2.0
WIDTH poly      2.0
WIDTH metal     3.0
WIDTH cnt       2.0
MARGIN cnt diff 1.5
MARGIN cnt poly 1.0
MARGIN cnt metal 1.0

/*Other design rules are omitted.*/

MIUNS boundary pwell  bulk
AND   diff     poly   gate
MINUS diff     poly   sdpn
AND   gate     pwell  ngate
MINUS gate     pwell  pgate
AND   cnt      poly   cmpoly
MINUS cnt      compoly cmdiff

ATTACH text TO metal
CONNECT metal TO poly BY compoly
CONNECT metal TO sdpn BY cmdiff
DEVICE p PMOS pgate poly sdpn bulk
DEVICE n NMOS ngate poly sdpn pwell
```

그림18. 테크놀리지 화일의 예
Fig. 18. An example technology file.

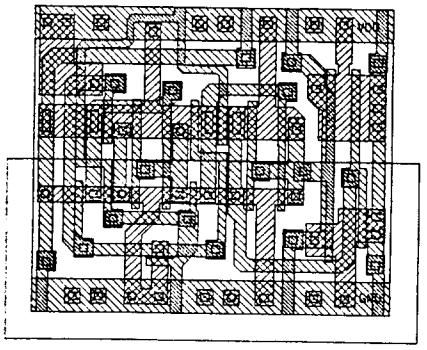
표 1. 회로 및 심볼릭 추출의 수행 시간
Table 1. Elapsed time for circuit and symbolic extraction.

cell	#tr	#node	#inst	shape	elapsed time
n805_datadec	121	68	0	rectilinear	3.2 sec
n805_sysgen	124	103	4	rectilinear	4.8 sec
timer	253	244	11	rectilinear	9.8 sec
df01	22	13	0	arbitrary	1.0 sec
mux4	24	23	0	arbitrary	0.8 sec

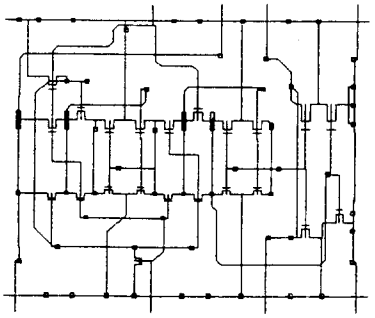
V. 실험 결과

회로 및 심볼릭 추출기의 성능을 평가하기 위하여 수작업에 의해 설계된 레이아웃을 사용하여 추출을 수행하였다. 그림18은 추출에 사용된 CMOS 공정의 추출 규칙을 나타내었으며, 표1에는 SUN4/40 상에서 수작업으로 설계된 MOS 레이아웃인 n805_datadec, n805_sysgen, timer 및 45° 패턴을 포함하는 df01과 mux4의 레이아웃에 대한 회로 추출 및 심볼릭 추출을 동시에 수행하는데 소요된 시간을 나타내었다. #tr은 트랜지스터의 갯수를 나타내며, #inst는 instantiation된 셀의 갯수를 나타낸다. 표1에 나타난 결과와 같이 기존의 심볼릭 추출기들이 처리하지 못하던 45°패턴 및 임의의 각도를 처리하고, 테크놀리지의 추출 규칙을 사용함으로써 모든 테크놀리지의 레이아웃으로 부터 회로 및 심볼릭 추출을 수행하면서도

수행 속도면에서 우수함을 알 수 있다. 그림19는 심볼릭 추출에 사용된 45°의 패턴을 포함하는 레이아웃인 df01에 대한 마스크 레이아웃 및 추출에 의해 발생된 스틱 다이아그램을 나타내었다.



(a)



(b)

그림19. 심볼릭 추출의 예
(a) 레이아웃 예
(b) (a)로부터 발생된 스틱 다이아그램

Fig. 19. An example of symbolic extraction.
(a) an example of layout,
(b) stick diagram extracted from (a).

Ⅵ. 결 론

임의의 각도를 가지는 계층적인 레이아웃으로 부터 회로 및 심볼릭 추출을 수행하는 시스템을 개발하였으며, 심볼릭 추출 과정의 다각형의 사다리꼴 분해 및 다각형으로부터의 심볼릭 경로를 찾아내는 새로운 효율적인 알고리듬을 제안하였다.

기존에 발표된 회로 추출기 및 심볼릭 추출기들의 경우 rectilinear의 패턴만을 처리할 수 있어 임의의 각도를 가지는 레이아웃에는 적용이 불가능하며, 시스템에 내장된 소자 컨택의 모델만을 사용함으로써 특정 테크놀리지에만 적용이 가능하였다. 이러한 문제점들을 해결하기 위하여 SOLID의 회로 및 심볼릭 추출기는 사용자가 추출 규칙을 테크놀리지 화일에 기술할 수 있게 함으로써 full-custom 방식의 MOS 및 바이폴라 레이아웃의 모든 테크놀리지에 대한 회로 및 심볼릭 추출을 가능하게 하였으며, 45°는 물론 임의의 각도를 가지는 레이아웃을 처리할 수 있도록 구현하였다.

심볼릭 추출의 과정에 사용되는 다각형의 사다리꼴 분해 및 사다리꼴 template를 이용한 심볼릭 경로 추출에 대해 효율적인 알고리듬을 제안하였으며, 실험을 통하여 45°는 물론 임의의 각도를 가지는 다각형과 도넛츠 패턴의 다각형에도 적용이 가능하여 모든 형태의 레이아웃에 대한 심볼릭 추출에 매우 적합함을 보였고, 수행 시간면에서 임의의 각도를 처리하면서도 우수한 속도를 나타내었다. 회로 및 심볼릭 추출기는 서강 레이아웃 설계 및 검증 시스템인 SOLID의 한 분야로서 SUN 워크스테이션상에서 C 언어를 사용하여 개발되었다.

參 考 文 獻

[1] P. Dood, J. Wawrzynek, E. Liu, and R. Suaya, "A two-dimensional topological compactor with octagonal geometry," *in Proc.* 28th DAC, June 1991, pp. 727-731.

[2] J. Dufourd, "The STICKIZER: A layout to symbolic converter," *in Proc.* ICCAD, Nov. 1989, pp. 534-537.

[3] M. Horowitz and R.W. Dutton, "Resistance extraction from mask layout data," *IEEE Trans. on Computer-Aided Design,* vol. CAD-2, no. 3, July 1983, pp. 145-150.

[4] B. Lin and A.R. Newton, "KAHLUA: A Hierarchical Circuit Disassembler," *in Proc.* 24th DAC, June 1987, pp. 311-317.

[5] D. Marple, M. Smulders, and H. Hegen, "An efficient compactor for 45° layout," *in Proc.* 25th DAC, June 1988, pp. 396-402.

[6] M.E. Newell and D.T. Fitzpatrick, "Exploitation of hierarchy in analyses of integrated circuit artwork," *IEEE Trans.* on Computer-Aided Design, Vol. CAD-1, no. 4, Oct. 1982, pp. 192-200.

[7] M. Sato, N. Ohba, H. Watanabe, and S. Saito, "Stick diagram extraction program SKELETON," *in Proc.* ICCAD, Nov. 1988, pp. 318-321.

[8] 문인호, 김현정, 오성환, 황선영, "Sliced-Edge Trace 알고리듬을 이용한 계층적 Incremental DRC 시스템," 대한전자공학회 논문지, 제28권 A편 제1호, 1991년 1월, pp. 60－73.

[9] 이용재, 문인호, 황선영, "상위 단계에서의 설계 검증을 위한 회로 추출 시스템," 대한전자공학회 추계 종합학술대회 논문집, vol. 13, no. 2, 1991년 11월, pp. 619－622.

著 者 紹 介

文 寅 鎬(正會員)

1962年 11月 15日生. 1985年 2月 한양대학교 전자공학과 졸업. 1990年 2月～현재 서강대학교 전자공학과 대학원 석사과정. 1984年 12月～현재 삼성전자 반도체 연구소 CAD실 주임 연구원. 주관심분야는 Design Verification, Layout Synthesis, CAD 시스템, Computer Architecture 등임.

●

李 容 在(準會員)

1967年 3月 19日生. 1990年 2月 서강대학교 전자공학과 졸업. 1990年 2月～현재 서강대학교 전자공학과 대학원 석사과정. 주관심분야는 Design Verification, Simulation, CAD시스템, Computer Architecture 등임.

黃 善 泳(正會員)

1976年 2月 서울대학교 전자공학과 졸업. 1978年 2月 한국 과학원 전기 및 전자공학과 공학석사 취득. 1986年 10月 미국 Stanford 대학 공학박사학위 취득. 삼성반도체(주) 연구원, Stanford 대학 CIS 연구소 연구원. Fairchild Semiconductor 기술 자문. 1989年 3月～현재 서강대학교 전자공학과 교수. 주관심분야는 CAD 시스템, Computer Architecture 및 Systems Design, VLSI 설계 등임.