論文 92-29A-9-9

솔거 : 45° Corner-stitching에 의거한 레이아웃 설계 시스템

(SOLGER : A Layout Design System Based on 45° Corner-stitching)

金 宰 範,* 鄭 成 太,* 李 在 晃,* 全 洲 植*

(Jae Bum Kim, Sung Tae Jung, Jae Hwang Lee, and Chu Shik Jhon)

要 約

본 논문에서는 종합적인 레이아웃 설계 시스템인 「솔거」에 대하여 설명한다. 솔거는 공통 레이아웃 데이타베이스를 중심으로, 강력한 기능을 제공하는 레이아웃 편집기, 방대한 양의 설계정보를 일관성 있게 관리하는 설계 정보 관리자, 점진 및 계층적 방식의 설계규칙 검사기, 노드 추출기 및 전기적 규칙 검사기, technology에 관련된 정보를 정의할 수 있게 하는 공정기술 편집기, 사용자가 정의한 프로시듀어를 활용할 수 있도록 하는 명령언어 처리기 등으로 구성된다. 또한 본 논문에서는 솔거 시스템 구성요소의 근간이 되는 45° corner-stitching 구조를 제안한다. 그리고 사용자는 솔거가 지원하는 다양한 기능과 다중 윈도우를 이용한 설계 환경 및 메뉴 방식의 사용자 인터페이스를 이용하여 레이아웃 설계를 수행할 수 있다. 현재 솔거는 레이아웃 설계에 실제 활용되고 있다.

Abstract

In this paper, we introduce an integrated layout design system, SOLGER. Our system incorporates useful design tools : a powerful layout editor, a coherent access mechanism for large volumes of design data, an incremental design rule checker for hierarchical design environment, node extractor and electrical rule checker, a technology capture which is used for defining technology-specific information, and a procedural design environment for user customization. Also, we present a modified corner-stitching data structure which allows 45°-angled bilateral edges. Users are provided with a multi-window design environment and a menu-driven interface. SOLGER is being used for VLSI designs practically.

I. 서 론

최근의 상용화된 CAD 소프트웨어들은 이전의 독립적인 형태에서 각 회로 설계 단계들을 유기적으로 연결하는 보다 진보된 형태로 발전하고 있다. 즉, VLSI 회로 설계 단계에 있어서의 편집, 시뮬레이션, 검증 및 분석 등은 물론 각 단계 간의 유기적인 연결을 통하여 합성, 추출, 회로 비교 등의 기능뿐아니라 방대한 양의 설계 정보들을 효율적으로 관리하는 기능들도 제공하고 있다. 이러한 경향은 이전의 CAD 소프트웨어들이 가지고 있던 여러가지 문제점들, 예를 들면 여러 tool들 간의 인터페이스 부재로 인한 비효율성과 같은 문

*正會員, 서울大學校 컴퓨터工學科
(Dept. of Computer Eng., Seoul Nat'l Univ.)
接受日字 : 1991年 4月 3日

제점들을 극복하고 CAD 소프트웨어들의 실용성을 높이기 위하여 필수적이라 볼 수 있다.

본 연구팀은 이러한 취지에서 기존의 레이아웃 편집기인 Magic 시스템[1]을 기반으로 종합적인 레이아웃 설계 시스템인 「솔거」를 개발하였다. 본 연구팀은 기존의 레이아웃 설계 시스템[1, 2, 3, 4]을 분석하고 레이아웃 설계자의 요구사항을 수렴하여 레이아웃 설계에 필요한 여러가지 기능을 추출하여 이를 구현하였다. 또한 이들 기능들을 일관된 인터페이스로 연계시킴으로써 이들 기능을 통합한 종합적인 레이아웃 설계 환경을 제공한다. 그림 1은 솔거의 시스템 구성을 나타낸다.

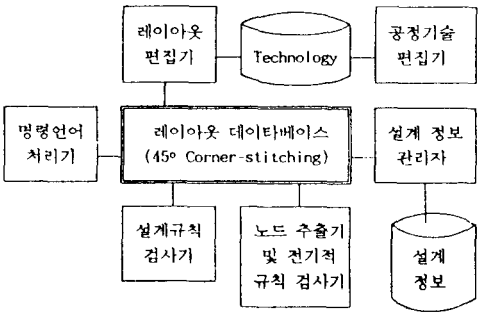


그림 1. 솔거 시스템 구성도
Fig. 1. Configuration of SOLGER system.

그림 1에서 보듯이 솔거[5, 6]는 공통 데이타베이스를 중심으로 강력한 기능을 제공하는 레이아웃 편집기, 방대한 양의 설계 정보를 일관성있게 관리하는 설계 정보 관리자, 점진 및 계층적 방식의 설계규칙 검사기, 노드 추출기 및 전기적 규칙 검사기, technology에 관련된 정보를 정의할 수 있게 하는 공정기술 편집기, 사용자가 정의한 프로시듀어를 활용할 수 있도록 하는 명령언어 처리기 등으로 구성된다. 또한 레이아웃을 표현하는 자료구조로는 연결 리스트 구조, 다차원 이진나무구조[7], quad 나무구조[8], corner-stitching[9] 등이 있는데 본 논문에서는 레이아웃 설계 시스템에 적합하다고 판명된 corner-stitching 자료구조를 채택한다. 기존의 corner-stitching 자료구조는 90° 도형만을 처리할 수 있으나 90° 도형만으로는 레이아웃이 조밀하지 못하고 회로의 지연 시간이 길어질 수 있다는 단점이 있다. 따라서 본 논문에서는 이러한 단점을 보완한 45° corner-stitching 자료구조[10]를 제안하며, 공통 레이아웃 데이타베이스는 이를 바탕으로 구성된다.

본 논문의 Ⅱ장에서는 솔거 시스템 구성요소의 근간이 되는 45° corner-stitching 자료구조에 대하여 자세히 설명하며, Ⅲ장에서는 솔거 시스템을 구성하는 각각의 구성요소에 대하여 설명한다. 또한 Ⅳ장에서는 본 논문의 결론을 내리고 앞으로의 연구방향을 제시한다.

Ⅱ. 레이아웃 자료구조

1. 기존의 레이아웃 자료구조

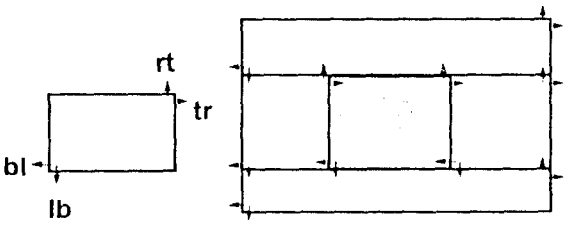
레이아웃 표현을 위한 자료구조로 연결 리스트 구조, 다차원 이진나무구조, quad 나무구조, corner-stitching 자료구조 등이 제안되었다. 이들 자료구조에 대하여 레이아웃 편집 시스템에서 기본적으로 사용되는 알고리듬의 시간 복잡도가 표 1에 나타나 있다. 솔거에서는 여러가지 소프트웨어들이 하나의 레이아웃 데이타베이스를 공유하므로 이 레이아웃 자료구조는 여러가지 소프트웨어에 모두 효율적이어야 한다. 레이아웃 편집기에서는 기본적인 작업으로서 점 탐색, 영역 탐색, 노드 탐색, 삽입, 삭제 작업이 사용되고, 설계규칙 검사기에서는 이웃 탐색, 영역 탐색 작업이 사용되며, 노드 추출기에서는 노드 탐색, 이웃 탐색, 영역 탐색 작업이 사용된다. 여기에서 이웃 탐색과 영역 탐색은 각 소프트웨어들에서 공통적으로 사용되는 작업임을 알 수 있다. 따라서 솔거에서는 표 1에 나타나 있듯이 이 두가지 작업에 모두 효율적인 corner-stitching 자료구조를 채택하였다.

표 1. 레이아웃 자료구조에 대한 기본 알고리듬의 시간 복잡도
Table 1. Time complexities of the basic algorithms for layout data structure.

자료구조 / 알고리듬	연결 리스트	다중이진 나무구조	Quad 나무구조	Corner Stitching
점 탐 색	N	log N	log N+T	sqrt(N)
이웃탐색	N	log N	log N+T	1
영역탐색	N	n+logN	T+n	sqrt(N)+n
노드탐색	n×N	n×logN	n(logN+T)	n
삽 입	1	log N	log N	sqrt(N)+n
삭 제	N	log N	log N+T	sqrt(N)+n

N : 전체 오브젝트 갯수
n : 작업영역 내의 오브젝트 수
(노드 탐색의 경우 노드내의 오브젝트 수)
T : 임계 값

Corner-stitching 자료구조는 레이아웃을 '타일'이라고 하는 사각형으로 표현한다. 이 corner-stitching 자료구조의 특징은 빈 공간을 나타내는 공백 타일과 특정 레이어로 고정된 고정 타일을 모두 데이타베이스에 저장한다는 것이다. 또한 그림 2에 나타나 있듯이 모든 타일들이 모서리 부분에서 연결되어 전체 레이아웃을 구성하고 있으며 타일은 최대 수평통합 방식[1]으로 통합되어 저장되므로 모든 타일은 서로 중복되지 않고 레이아웃 평면 상의 한 점은 한 타일에만 속하게 된다. 이와 같은 특성으로 인하여 다른 레이아웃 자료구조 보다 이웃 탐색이 빠르게 수행된다.



(a) 타일의 구조 (b) 레이아웃의 구조

그림 2. Corner-stitching 자료구조
Fig. 2. Corner-stitching data structure.

그러나, 기존의 corner-stitching 자료구조는 맨하탄 형태의 레이아웃만을 허용하므로 레이아웃의 구조 및 설계규칙이 단순하여 설계가 용이하지만 레이아웃이 조밀하지 못하다는 단점을 가지고 있다. 임의의 각을 허용하면 레이아웃의 크기를 줄일 수 있고, 배선의 길이가 짧아져 신호 지연 시간이 감소되나 반면에 레이아웃 편집기의 복잡도가 너무 커져서 수행속도가 느려진다. 따라서 솔거 시스템에서는 시스템의 성능에 크게 영향을 미치지 않으면서 레이아웃의 효율을 상당히 향상시킬 수 있도록 45° 도형을 허용한다.

2. 45° corner-stitching 자료구조

45° corner-stitching 자료구조 역시 타일을 기본 자료구조 단위로 레이아웃을 형성하며 타일이 겹치는 것을 허용하지 않는다. 또한 최대 수평 통합 방식을 통해 타일을 형성하므로 타일의 좌우변은 X축에 대하여 45°, 90°, 135°의 각도를 가지며 윗변과 아랫변은 X축에 평행하게 된다. 즉, 모든 타일은 사다리꼴의 범주내에 속하며 가능한 형태가 그림 3에 나타나 있다. 하나의 타일을 표현하기 위한 내부 자료구조는 다음과 같다.

```
struct tile
  {
    int ti_body;
    struct tile    *ti_lb;
    struct tile    *ti_bl;
    struct tile    *ti_tr;
    struct tile    *ti_rt;
    Point ti_ll;
    int ti_status;
```

위의 자료구조에서 표 1의 알고리즘들을 효율적으로 구현할 수 있도록 하기위해서 ti_lb, ti_bl, ti_tr, ti_rt는 다음과 같이 기존의 corner-stitching 자료구조와는 다른 타일을 가리키도록 수정되었다.

· ti_lb(ti_rt)는 현재 타일의 좌(우)측변과 동일한 직선상에 있는 좌(우)측변을 가지고 현재 타일의 아랫(윗)변의 y값을 윗(아랫)변의 y값으로 가지는 타일을 가리키거나, 이 두 조건을 만족하는 타일이 없는 경우 아랫(윗)변에 인접하는 타일중 제일 좌(우)측에 있는 타일을 가리킨다.

· ti_bl(ti_tr)은 현재 타일의 좌(우)측변과 인접한 타일 중, 제일 하(상)단에 있는 타일을 가리킨다.

ti_ll은 타일을 포함하는 최소의 직사각형을 그렸을 경우의 좌측 하단점을 나타내는 좌표값을 가지고 있다. ti_status는 타일의 형태를 나타내는 것으로 최상위 1비트는 사용되지 않으며, 그 다음 1비트는 타일이 직사각형인지 사다리꼴인지를 나타내며 그 다음 3비트씩은 각각 좌우측변의 기울기를 나타낸다. 이 필드의 값이 0이면 기울기가 무한대임을 나타내고 1이면 기울기가 1임을 나타내고 2이면 기울기가 −1임을 나타낸다. 그림 3에 9가지 형태의 타일과 각각의 ti_status 값이 나타나 있다.

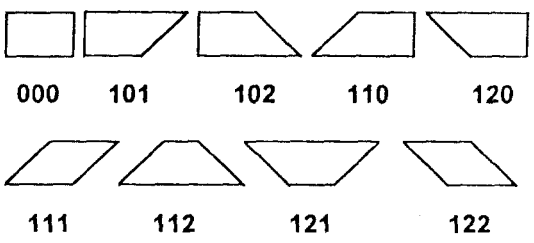


그림 3. 허용되는 타일의 형태들
Fig. 3. Tile patterns.

3. 기본 알고리듬

본 절에서 설명하는 기본 알고리듬들은 기존의 corner-stitching 자료구조에서의 알고리듬과 유사하지만 세부적으로는 많은 부분이 사다리꼴 처리를 위하여 수정되었다.

1) 점 탐색

점 탐색은 특정 위치가 포함되는 타일을 탐색하는 작업으로, 수직 수평으로 반복적으로 이동하며 어느 타일에서나 탐색 작업이 시작될 수 있다. 점 탐색의 수행 과정은 다음과 같고, 이 탐색 작업 과정의 예가 그림 4에 나타나 있다.

i) 먼저 주어진 y 좌표값을 포함하는 수직 영역을 가진 타일이 발견될 때까지 ti_lb 또는 ti_rt stitch를 따라 아래 또는 윗 방향으로 수직 이동한다.

ii) 수평 영역이 x 좌표값을 포함하되 주어진 y 좌표값에서 x 좌표값을 포함하는 타일이 발견 될 때까지 또는 수평 이동으로 인해 수직 영역이 y를 포함하지 않는 타일이 발견될 때까지 ti_bl 또는 ti_tr stitch를 따라 왼쪽 또는 오른쪽으로 수평 이동한다.

iii) y좌표값이 어긋난 타일이 발견될 경우 주어진 점을 포함하는 타일이 발견될 때까지 단계 i), ii)를 반복 수행한다.

주어진 레이아웃 상의 타일의 수를 N이라 할때, 이 알고리듬의 복잡도는 최악의 경우, 예를 들면 모든 타일이 한 줄로 연결되어 있는 경우, 0(N)이다. 그러나 실제적으로 이러한 레이아웃은 있을 수 없으며 대부분의 경우 레이아웃 사각형 형태로 구성되므로 이 알고리듬의 복잡도는 $0(N^{1/2})$이라 할 수 있다. 또한 레이아웃 편집 작업은 지역적으로 수행되므로 이전에 탐색된 타일을 시작으로 할 경우 점 탐색에 소요되는 시간을 더욱 줄일 수 있다.

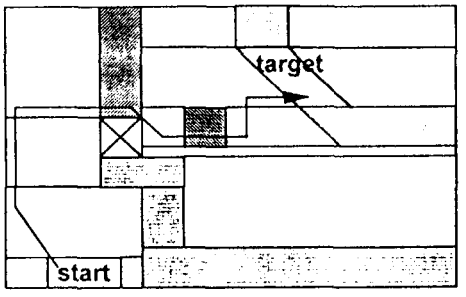


그림 4. 점 탐색 수행 과정 예
Fig. 4. An example of the point search.

2) 이웃 탐색

가장 빈번히 사용되는 작업중의 하나는 인접한 타일을 찾는 이웃 탐색 작업이다. Corner-stitching 자료구조에서는 이웃한 타일이 corner-stitch에 의해 연결되어 있으므로 이웃 탐색은 이들을 따라가면서 탐색하면 된다. 따라서 이웃 탐색의 복잡도는 0(1)이라 할 수 있다.

3) 영역 탐색

영역 탐색은 주어진 영역 내의 모든 타일을 찾는 작업으로 왼쪽에서 오른쪽, 윗쪽에서 아랫쪽으로 타일을 검색한다. 영역 탐색의 수행 과정은 다음과 같고 영역 탐색 과정의 예가 그림 5에 나타나 있는데 타일에 붙여진 번호는 타일이 탐색된 순서를 나타낸다.

i) 먼저 점 탐색 방법을 이용하여 영역의 좌측 상단점을 포함하는 타일을 탐색한다.

ii) 주어진 영역의 왼쪽변을 포함하는 모든 타일에 대해 단계 iii)을 수행한다.

iii) 타일의 오른쪽 변에 인접한 타일중에서 아랫변의 y좌표값이 현재의 타일보다 크고 주어진 영역에 포함된 타일을 recursive하게 탐색한다.

주어진 영역에 겹치는 타일을 찾는 작업의 수행시간은 주어진 영역에 전체 또는 일부가 포함되는 타일의 수에 비례한다. 그리고 과정 i)에서 점탐색을 통하여 시작 타일을 찾아야 하므로 영역 탐색의 복잡도는 0(n+sqrt(N))이다. 여기에서 N은 레이아웃 데이타베이스 내의 전체 타일 수이며 n은 주어진 영역에 겹치는 타일의 수이다.

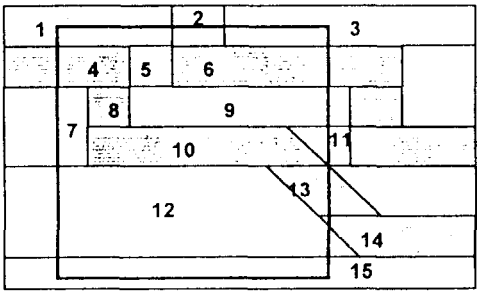


그림 5. 영역 탐색 수행과정 예
Fig. 5. An example of the region search.

4) 노드 탐색

노드 탐색은 전기적으로 연결된 타일을 찾는 작업으로 레이아웃 편집기와 노드 추출기에서 사용된다. 노드 탐색을 수행하는 과정은 다음과 같다.

i) 주어진 타일의 각 변을 따라가면서 이웃한 타일들 중에서 전기적으로 연결된 타일을 찾는다.

ii) 과정 i)에서 발견된 타일들에 대해 과정 i), ii)를 recursive하게 수행한다.

이웃한 타일은 하나의 corner-stitch만을 따라가면 찾을수 있으므로, 전기적으로 연결된 타일의 수를 n이라 할때, 노드 탐색 알고리듬의 복잡도는 0(n)이다. 다른 자료구조에 비해 corner-stitching 자료구조에서 이 알고리듬의 복잡도가 낮은 이유는 이웃 탐색이 빠르기 때문이다.

5) 삽입과 삭제

삽입은 새로운 고정 타일을 레이아웃 데이타베이스에 첨가하여 레이아웃 데이타베이스를 새로이 구성하는 과정이다. 삭제는 주어진 영역에 공백 타일을 삽입하는 작업이다. 타일 삽입 알고리듬은 다음과 같고 삭제 과정도 이와 유사하다.

i) 생성하고자 하는 고정 타일 영역내에 존재하는 타일들을 영역 탐색을 이용하여 찾는다.

ii) 발견된 타일이 영역내에 완전히 포함되지 않은 경우에는 타일을 분리하여 영역 밖에 위치하는 타일과 영역내에 위치하는 타일을 생성한다.

iii) 영역내에 위치하는 타일들 중에서 공백 타일을 고정 타일로 변경한 다음 최대 수평통합을 수행한다.

Ⅲ. 시스템 구성요소

1. 레이아웃 편집기

솔거의 레이아웃 편집기는 Magic의 레이아웃 편집기를 바탕으로하여 구현하였다. 45° plowing, 45° stretch 기능 등을 제외한 대부분의 기능을 사다리꼴을 지원할 수 있도록 개선하였으며 45° 배선 기능과 같은 새로운 기능도 추가하였다. 이에 더불어 설계를 더욱 편하게 할 수 있도록 사용자 인터페이스를 개선하고 여러가지 유용한 기능을 추가하였다.

1) 사용자 인터페이스

솔거에서는 다중 윈도우 기능을 이용하여 화면을 여러 편집 공간으로 나누어 동시에 여러가지 작업을 수행할 수 있다. 대부분의 기능들을 아이콘으로 표현된 메뉴를 사용하여 수행할 수 있도록 함으로써 명령을 암기하고 입력하는 번거로움을 없앴으며, 메뉴들은 다중 윈도우 기능을 이용하여 독립적인 윈도우 상에 표시함으로써 화면을 효과적으로 사용할 수 있도록 하였다. 또한 다중 윈도우 방식의 장점을 십분 활용하여 환경 설정 윈도우, 도움말 윈도우, 설계 정보 관리 윈도우, 컬러 지정 윈도우 등 다양한 윈도우를 제공함으로써 사용자의 편의를 도모하였다.

2) 추가 기능

솔거의 레이아웃 편집기에는 효율적인 레이아웃 편집을 위해 여러가지 기능이 추가되었다. 계층 구조 윈도우는 설계중인 레이아웃에 계층 구조가 있을 경우 이를 트리 구조로 보여줌으로써 레이아웃의 계층 구조를 쉽게 파악하고 설계 영역 이동을 쉽게 해준다. 또한 포트 개념을 도입하여 어떤 셀(cell)의 내부를 보이지 않게 하는 대신에, 그 셀의 외부 연결점인 포트만을 보여줌으로써 셀간의 배선 작업을 쉽게 하였다. Magic에서는 레이아웃의 일부를 선택하여(selection), 그에 대해 삭제, 회전 복사 등 여러가지 작업을 할 수 있었다. 이를 발전시켜, 솔거에서는 선택된 레이아웃의 일부, 즉 여러개의 객체들을 하나의 객체와 같이 취급되도록 하는 그룹 기능을 제공한다. 이외에도 여러가지의 유용한 기능들이 추가 되었다.[5,6]

2. 설계 정보 관리자

일반적인 레이아웃 편집기들은 설계된 셀을 화일의 형태로 저장하는 것이 보통이다. 그러나 설계한 셀의 갯수가 늘어남에 따라 관리가 어려워진다. 또한 하나의 셀을 설계함에 있어서도 계속적인 수정 과정을 거쳐 하나의 완성된 셀을 설계하게 되는데 이 과정에서 많은 시행착오를 거치는 것이 일반적이다. 즉, 설계된 셀의 내용을 수정하였다가 문제가 발생하여 다시 원래의 셀이 필요할 수가 있다. 이러한 경우 설계자가 다른 이름으로 셀들을 저장하여 대비할 수 있으나 매우 번거로운 일이 아닐 수 없다. 솔거에서는 이러한 점을 고려하여 방대한 설계 정보를 쉽게 검색할 수 있고 하나의 셀에 대하여 설계가 진행됨에 따라 버전이 다른 셀들을 관리할 수 있도록 데이타베이스 기법을 도입하여 설계 정보를 관리하는 기능을 제공한다. 따라서 설계자가 솔거 시스템을 사용하여 설계한 모든 셀들은 설계 정보 관리자를 통해 데이타베이스에 저장되고 다시 사용할 때에도 설계 정보 관리자를 통하여 접근하도록 하였다. 이러한 방법을 사용함으로써 설계자에게 설계 정보에 대한 일관성있는 접근 통로를 제공하게될 뿐만 아니라 설계된 셀의 특성 정보등을 이용하여 간단한 형태의 질의를 처리할 수 있도록 하였다. 또한 설계 정보 관리자를 통하여 솔거를 사용하는 다수의 설계자들에 대하여 설계 정보를 공유하면서도 각 설계자의 독립성을 유지시켜 줄 수 있다. 솔거 시스템의 설계 정보 저장을 위한 데이타베이스는 셀 단위의 계층적 데이타 표현을 기반으로 하며, 솔거 시스템 고유의 양식으로 기술된 마스크 정보 화일이나 네트리스트 화일 등을 저장한다. 그리고 솔거 시스템으로 설계되지 않은 Calma GDSII 형식이나 CIF로 저장된 셀들도 저장이

가능하다.

설계 정보 관리자는 솔거의 메뉴 또는 명령을 통하여 수행되며 윈도우를 통하여 편리하게 사용할 수 있도록 되어 있다. 사용자의 간단한 질의를 처리하기 위하여 셀의 이름, 설계자의 이름, logic 이름, 공정기술, 설계 날짜, 셀의 크기 그리고 셀에 대한 설명 등을 기술하는 field들을 가지고 있어 사용자가 원하는 셀들을 쉽게 찾아 볼 수 있도록 하고 있다. 사용자는 셀 이름의 일부만으로도 셀을 검색할 수 있으며 사용자 질의에 의해 선택된 셀들은 윈도우를 통해 나열되어 사용자가 선택할 수 있게 되어 있다. 그리고 설계된 셀에 대한 설명을 첨가하는 기능도 가지고 있어 다른 설계자에 의해 설계된 셀을 사용할 경우에도 셀의 기능을 쉽게 알 수 있다. 또한 솔거 고유의 형식이 아닌 다른 형식으로 저장된 셀을 위하여 셀의 저장 형식, 외부에서 사용하는 셀 이름 등의 정보도 지정할 수 있게 되어 있으며 셀의 등록 및 삭제 등의 기능을 제공한다.

3. 명령언어

일반적으로 레이아웃 편집기는 설계자와의 대화형 작업을 통하여 작동한다. 그러나 유사한 작업이 반복되는 경우와 같이 이러한 대화형 작업이 부적합한 경우도 있다. 이러한 단점을 보완하기 위한 방편으로 기존의 레이아웃 편집기들은 각자 고유한 형식의 명령언어를 제공하고 있다. 예를 들면 Calma GDSII 시스템에서는 GPLII를 제공하고 있으며 Valid의 EDSIII에서는 EPL을 제공하고 있다. 이러한 기존의 명령언어들은 설계자의 반복되는 작업을 간소화 하거나 설계자의 필요에 맞도록 시스템을 구성(customization)하는 등의 목적으로 사용될 수 있다. 이러한 기능은 설계자에게 procedural한 형태의 설계 환경을 제공하기 위한 것으로 각 CAD tool들 마다 독특한 형태의 프로그래밍 언어를 제공하고 있다. 기존 레이아웃 편집기에서 제공하는 언어들을 살펴보면 대부분 기존의 범용 프로그래밍 언어와 마찬가지로 'if-then-else', 'for-loop', 'while-loop', 'switch' 등의 제어문과 변수 선언 기능을 제공하고 있다. 실제로 GPLII는 APL과 ALGOL에 기반을 두고 구성되었으며 일반적 제어기능 이외에 레이아웃 편집기와 관련된 각종 함수들을 제공하고 있다. EPL의 경우 레이아웃 자료구조와 관련된 'add-els', 'delel', 'getel', 'mselect' 등의 함수와 수치 계산용 함수, 그리고 화일 입출력 함수 등의 각종 utility 함수들이 제공되고 있다. 따라서 레이아웃 편집기에서 제공하는 명령언어와 기존의 프로그래밍 언어와의 차이는 레이아웃 편집기에서 사용하는 레이아웃 정보를 조작하는 함수들에 있다고 할 수 있다.

솔거에서는 기존의 레이아웃 편집기에서 제공하는 명령언어가 레이아웃 정보를 조작하는 함수 부분을 제외하면 기존의 프로그래밍 언어와 매우 유사하다는 점에 착안하여 기존 프로그래밍 언어의 하나인 C 언어를 명령언어로 활용하는 방법을 채택하였다. C 언어는 현재 가장 널리 쓰이는 언어 중의 하나로 프로그램 개발, 특히 UNIX 환경하에서의 프로그램 개발에 적합한 언어이다. 새로운 명령언어를 정의하기 보다 기존의 프로그래밍 언어인 C를 그대로 사용함으로써 얻을 수 있는 장점으로는 C 언어가 이미 널리 사용되고 있는 언어라는 점에서 별도의 훈련 과정이 없이도 쉽게 사용할 수 있다는 점과 기존의 라이브러리 함수들은 물론이고 솔거를 개발하는 과정에서 작성된 많은 내장 함수들을 그대로 사용할 수 있다는 점을 들 수 있다.

C 언어를 명령언어로 사용하기 위해서는 사용자가 작성한 프로그램이 실행중인 솔거 내부에서 실행될 수 있어야 한다. 이를 위하여 C 언어 인터프리터를 구하는 방법도 생각할 수 있으나 기존의 C 컴파일러를 그대로 이용하는 방법을 택하였다. 즉 사용자가 작성한 프로그램을 기존의 C 프로그램과 동일한 방법으로 컴파일하여 오브젝트 코드를 생성하고 솔거 내부에는 이 오브젝트 코드를 실행할 수 있도록 하여 주는 기능만 가지고 있다. 이러한 방법이 가능한 것은 C 컴파일러가 생성하는 오브젝트 코드가 재배치 가능한 정보를 가지고 있으므로 이를 이용하여 실행중인 프로그램과 링크하여 수행될 수 있기 때문이다. 시스템에 따라 차이가 있을 수 있으나 UNIX에서의 오브젝트 화일은 header, text, data, reloc_info, symtab의 5개의 영역[12]으로 나누어 진다. Header는 오브젝트 코드의 각 영역의 크기와 기타 정보를 가지고 있으며 text영역은 실제 수행될 명령어 코드를 가지고 있다. 이 영역에 사용된 명령어들의 주소값 부분은 이 프로그램이 수행될 때 적재되는 메모리의 주소값에 따라 결정되는, 즉 재배치 되어야 하는 부분이 된다. 따라서 오브젝트 코드의 실행을 위해서는 반드시 재배치 과정을 거쳐야만 한다. Data 영역은 프로그램 내에서 선언한 전역 변수들이 차지하는 영역이다. Reloc_info 영역은 text와 data 영역중 재배치되거나 external 참조를 사용하는 부분에 대한 정보로서 최종적인 실행 화일을 만들기 위한 링크과정에서 사용된다. Symtab은 프로그램 내에 사용된 심볼들의 위치와 기타 정보를 나타내는 것으로 역시 링크 과정에서 사용된다. 따라서 이러한 구조를 갖는 오브젝트 화일을 실행중인 프로그램에서 실행시키기 위해서는 실행중인 프로그램 내부에 오브젝트 코드를 읽어들일 영역을 확보하여야 하며 일반 프로그램의 링크 과정과 마찬가지의 링크 과정을 거쳐

오브젝트 코드의 재배치를 해야만 한다. 실행중인 프로그램 내부에 오브젝트 코드를 읽어들이기 위한 메모리는 동적 메모리 할당을 통해 확보할 수 있으며 적재되는 내용은 오브젝트 화일의 text와 data 영역이 된다. 적재된 오브젝트 코드의 재배치는 오브젝트 화일의 reloc_info와 symtab 그리고 실행중인 프로그램의 실행화일에 있는 심볼 테이블을 이용하여 이루어 질 수 있다. 따라서 실행중인 프로그램의 실행화일에는 반드시 심볼 테이블이 있어야 한다. 예를 들어 사용자가 선언한 함수 ul에서 실행중인 프로그램 내부의 함수 f1을 호출하는 경우 사용자 프로그램의 reloc_info에는 f1을 호출하는 부분이 external 참조임을 표시하고 있다. 따라서 이를 처리하기 위해서는 실행중인 프로그램의 실행 화일에 있는 심볼테이블로부터 f1의 시작 주소를 알아내어 이를 f1을 호출하는 사용자 프로그램의 코드에 넣어 줌으로써 해결할 수 있다. 이 과정은 UNIX의 'ld' 명령이 수행하는 작업과 동일하므로 이를 프로그램화 하지 않고 'ld' 명령을 그대로 이용하여 해결하였다. 그림 6은 이러한 과정을 그림으로 나타낸 것이며 그림 7은 솔거의 기본 명령어를 조합하여 새로운 명령을 정의하는 예이다.

그림 7에서는 솔거에서 제공하는 명령 'mkcell', 'cancel', 'box', 'addcell'을 이용하여 'subscell'이라는 새로운 명령을 정의하였다. 새로이 정의된 'subscell' 명령은 사용자가 선택한 영역을 하나의 셀로 만들어 하나의 셀 인스턴스로 대치시키는 역할을 하게 된다. 사용자가 정의한 새로운 함수가 명령으로 사용

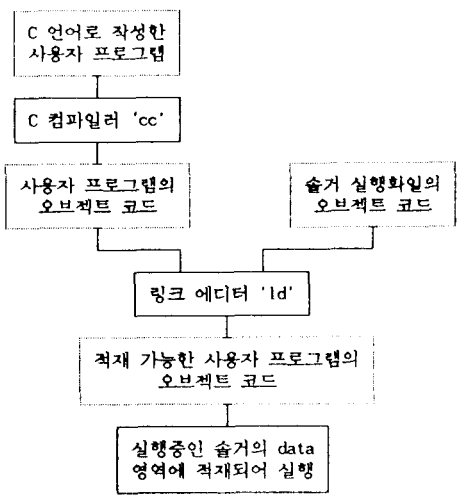


그림 6. C언어로 작성된 사용자 프로그램의 실행과정
Fig. 6. Execation process of a user program.

```
#include <solger/dyload.h>
SubsCell(argc, argv)
int argc;
char *argv[];
{
  SolgerRect box;
  if (argc != 2) {
    printf("must specify new cell name\n");
    return;
  }
  SelectBBox(&box);
  EvalCmd("mkcell %s", argv[1]);
  EValcmd("box %d %d %d %d",
    box.r_ll.p_x, box.r_ll.p_y, box.r_ur.p_x, box.r_ur.p_y);
  EvalCmd("addcell %s", argv[1]);
}
DyloadInit()
{
  DefCmd("subscell", SubsCell);
}
```

그림 7. 사용자 프로그램 작성 예
Fig. 7. An example of user program.

되기 위해서는 DefCmd 함수를 호출하는 DyloadInit이라는 함수를 작성해 주어야 하며 DyloadInit이라는 함수는 사용자 프로그램이 솔거와 링크된 후 자동적으로 실행되는 함수로서 사용자가 작성한 함수를 명령어 또는 메뉴의 형태로 사용할 수 있도록 초기화한다.

그림 7에서 볼 수 있듯이 사용자는 솔거에서 제공하는 함수들을 사용하여 원하는 기능을 수행시킬 수 있으며 기존의 C 프로그램과 전혀 차이없이 프로그램을 작성할 수 있다. 이때 사용자가 작성한 함수나 변수의 이름이 솔거 내부의 함수나 변수와 충돌할 수 있으므로 이를 방지하기 위하여 솔거가 사용자에게 제공하는 함수와 변수만을 선택하여 솔거 실행 화일의 심볼 테이블에 저장한다. 따라서 솔거의 실행 화일의 심볼 테이블은 솔거 내부의 모든 심볼들을 가지고 있지는 않으며 사용자에게 개방된 함수나 변수들에 대한 심볼들만을 가지고 있다.

4. 설계규칙 검사기

설계규칙 검사란 설계한 레이아웃이 제조 공정의 정밀도(mask misalignment, overetching 등)와 물리적 한계(metal migration 현상 등) 등의 제약으로 표현되는 설계규칙을 만족하는 지 여부를 검증하는 레이

아웃 설계의 한 단계이며, 설계규칙을 만족하는 회로는 레이아웃 단계에서의 동작성이 보장된다.

솔거의 설계규칙 검사기는 Magic 시스템의 점진적인 설계규칙 검사기[13]를 기반으로 개발되었으며, 솔거의 레이아웃 자료구조에서 허용되는 사다리꼴의 기하요소를 고려하여 45° 방향의 설계규칙 검사를 수행할 수 있다. 솔거의 설계규칙 검사기에서는 아래에 개념적으로 기술된 바와 같이 사용자의 설계 명령 사이의 idle time을 이용하는 이면 검사기(background checker)를 사용한다.

```
LayoutDesign()        /* 레이아웃 설계 */
{
  while (TRUE) {
    Edit();           /* 편집 명령 처리 */
    if (database changed) mark re-check area;
    if (there exists re-check area) DRC();
  }
}
DRC()  /* 이면 설계규칙 검사 */
{
    while (not INTERRUPT)
        Do DRC for each re-check area;
}
```

즉 회로 설계 시 내부 데이타베이스를 변화시키는 명령이 내려지면 솔거는 변화된 부분 주위의 영역(설계규칙에서 정의된 최대거리만큼 확장된 영역)을 검사를 필요로 하는 재검사 영역(re-check area)으로 표시하며, 이면 검사기는 검사를 필요로 하는 영역이 있으면 이 영역에 대해 설계규칙 검사를 수행한다. 각 재검사 영역을 검사하는 도중 INTERRUPT(다른 명령어, 키보드 입력, 마우스 입력 혹은 시스템 신호 등)가 발생하면 이면 검사기의 설계규칙 검사 작업을 중단하고 발생된 INTERRUPT를 처리한다. 이에 따라 이미 검사가 수행된 후 변화되지 않은 부분에 대해서는 검사를 하지 않고, 반드시 재검사를 필요로 하는 곳에 대해서만 검사를 수행함으로써 설계규칙 검사에 소요되는 시간적 효율을 증대시킬 수 있다. 또한 설계자의 회로 설계 작업과 설계자의 설계 고려 시간을 이용한 검사를 병행시킴으로써 오류를 즉시 검출할 수 있으며 검출된 오류를 즉시 수정할 수도 있다.

솔거의 설계규칙은 그림 8과 같이 마스크 사각형의 변을 중심으로 허용가능한 마스크 타입을 정의한다.

그림 8은 좌측의 마스크 타입이 T_i이고 우측의 마스크 타입이 T_j인 변 e에 대하여 정의된 거리 d 이내에

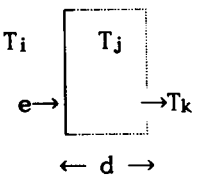
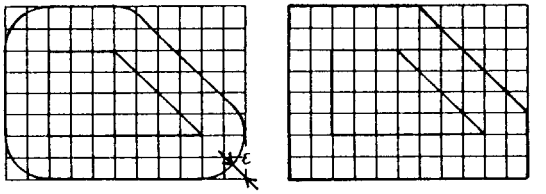


그림 8. 설계규칙의 기본 유형
Fig. 8. A typical example of design rules.

T_k 타입의 마스크 사각형만이 존재할 수 있음을 나타낸다. 따라서 솔거의 설계규칙 검사기는 각각의 검사 영역과 중첩되는 마스크 사각형의 변을 정의된 거리만큼 확장하여 허용되지 않은 타입의 마스크 사각형이 존재하는 지를 검사하는 방법을 이용한다.

그러나 솔거에서는 Magic 시스템과는 달리 마스크 레이아웃을 구성하는 기하요소가 45° 변을 가지는 사다리꼴이 허용되므로 그림 9와 같은 형태로 변을 확장하여 검사를 수행한다.



(a) Euclidean 확장 (b) 솔거의 확장

그림 9. 사다리꼴 확장
Fig. 9. Trapezoidal expansion.

그림 9(a)와 같은 이상적인 Euclidean 확장 방법을 사용할 경우 정확한 오류검출을 기할 수는 있으나 솔거에서 허용하는 기하형태를 벗어날 뿐아니라 모서리 부분의 ε 거리 내에는 다른 마스크 사각형의 일부분이 포함될 수 없으므로 이러한 Euclidean 확장법이 필수적이라고는 볼 수 없다. 따라서 솔거에서는 검사의 정확성과 효율성을 위하여 그림 9(b)와 같이 Euclidean 확장 방법에 근접하는 정확성을 가지는 확장 방법을 사용한다.

솔거의 점진적인 설계규칙 검사기는 Magic 시스템에서와 마찬가지로 계층적 설계 환경에도 적절히 사용될 수 있다. 즉 셀 내부의 마스크 정보뿐만 아니라, 셀과 셀 사이, 셀과 마스크 정보 사이 등의 모든 계층적 상황을 고려하여 설계규칙을 위반하는 오류를 검출한

다. 뿐만 아니라 그림 10과 같은 배열로 구성된 계층적 레이아웃의 경우, 점선으로 표시된 영역이 해당 배열 구조의 특이 패턴이므로 이들에 대해서만 설계규칙 검사를 수행한다.

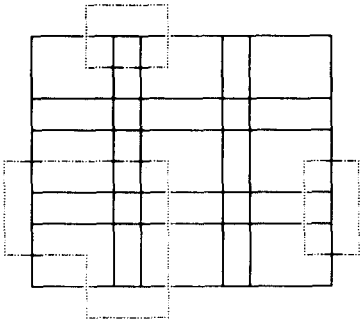


그림 10. 배열에 대한 검사 영역
Fig. 10. Design rule check area for an array.

한편 솔거에서는 설계규칙 검사를 활성화/비활성화 시키는 기능, 오류의 이유를 알려주는 기능, 그리고 오류의 위치를 탐색하는 기능, 오류 갯수 검색 기능 등을 제공하며, 일괄적 방식의 설계규칙 검사도 가능하다. 또한 솔거는 사용자의 편의를 위하여 공정 기술 편집기를 이용하여 설계규칙을 정의할 수 있도록 지원한다.

5. 노드 추출기 및 전기적 규칙 검사기

설계규칙 검사기는 레이아웃의 상대 위치를 검사함으로써 제조 공정상의 오류를 방지하지만 단순히 설계규칙 검사만으로는 회로 작동의 정확성을 보장할 수 없다. 보통 정확한 회로 작동을 확인하기 위하여 시뮬레이터등이 사용되나, 시뮬레이터를 사용하여 복잡한 회로 분석을 수행하기 이전에 간단한 트랜지스터의 연결 상태만을 검사함으로써 설계상 오류의 발견과 수정에 많은 도움이 될 수 있다. 전기적 규칙이란 회로의 동작을 이해하지 않고도 검사할 수 있는 단순한 기하학적 모양과 연결상태에 따른 회로의 특성이다. 예를 들면, 전체 회로의 전원 소모량, 트랜지스터의 비율 검사, 절연, 그리고 단락(short)된 회로 검사등이 있다.

1) 노드 추출

전기적 규칙 검사를 수행하기 위해서는 먼저 회로 연결 정보를 추출해야 한다. 솔거의 노드 추출기는 계층적으로 회로 연결 정보를 추출한다. 따라서, 레이아웃의 일부분이 변경된 경우, 변경된 부분을 포함하는 셀과 그 상위 셀들만을 새로이 추출하는 점진적 회로 추출 방식을 이용하여, 필요한 부분만을 효과적으로 재추출하게 되므로 회로 추출에 소요되는 시간이 단축된다. 한 셀의 회로 연결 정보 추출 과정은 다음과 같다.

i) 셀이 기본 마스크와 하위 셀로 구성되어 있는 경우, 기본 마스크 레이아웃에 대하여 II 장 3절에 기술된 노드 탐색 알고리듬을 이용하여 전기적으로 서로 연결된 타일들에 하나의 노드 이름을 부여한다.

ii) 기본 마스크 레이아웃에서 타일의 타입이 채널인, 즉 poly와 diffusion 타입의 타일이 겹쳐서 생성된 타일에 대해 이웃 타일을 탐색함으로써, 이 채널이 형성하는 트랜지스터의 종류, 채널에 연결된 소오스, 드레인, 게이트 단자와 노드 이름 등을 추출한다.

iii) 하위셀들에 대한 회로 정보는 이미 추출되어 있으므로 이들 정보를 기본 마스크 레이아웃에 대해 추출한 정보와 합병한다. 그러나 하위셀들이 서로 겹치는 경우, 새로운 트랜지스터가 생성되거나 노드가 합병될 수 있다. 따라서 겹치는 부분에 대해서는 그부분만 평면화시킨 다음 다시 연결 정보를 추출한다.

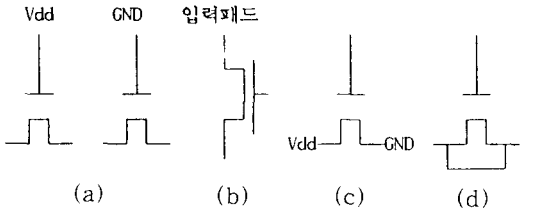
2) 전기적 규칙검사

전기적 규칙을 검사하기 위해서는 노드 추출기에 의해 추출된 회로 정보이외에 설계한 회로의 입출력 단자의 이름과 그 단자가 입력 단자인지, 출력 단자인지, 또는 입출력 단자인지를 입력해야 한다. 또한 "Vdd" 단자와 "GND" 단자를 반드시 레이아웃 상에서 지정해 주어야 한다. 전기적 규칙 검사 과정은 다음과 같다.

i) 추출된 회로의 연결 정보로부터 그래프 형태의 내부 자료구조를 구성한다. 솔거의 전기적 규칙 검사기는 계층적인 검사를 지원하지 못한다. 따라서 계층적인 회로 연결 정보를 읽어들여 이를 평면화시킨다.

ii) 깊이우선탐색(depth-first-search)방법을 이용하여 "Vdd" 단자나 입력 단자로부터 그래프를 탐색하면서 그림 11과 같은 모양의 회로가 존재하는지 검사한다. 오류가 발견되면 레이아웃에 오류를 나타내는 레이블을 붙인다.

위의 검사이외에도 트랜지스터의 비율, 트랜지스터의 단자가 입출력 단자도 아니면서 회로의 다른 부분과 전혀 연결되어 있지 않은 경우, 패스 트랜지스터의 출력이 다른 패스 트랜지스터의 입력 단자에 연결되어 있는 경우, 한 단자에 여러개의 풀업(pullup) 트랜지스터가 연결되어 있는 경우 등을 검사한다.



(a) Vdd 또는 GND가 입력에 연결된 경우
(b) 드레인이나 소오스에 입력 패드가 연결된 경우
(c) 단락된 트랜지스터
(d) 소오스와 드레인이 동일한 트랜지스터

그림11. 전기적 규칙 검사의 유형
Fig. 11. Types of electrical rule checks.

6. 공정기술 편집기

솔거는 집적회로에 관한 많은 일반적 지식을 프로그램에 직접 가지고 있지만, 여러 종류의 공정에 쉽게 적용될 수 있도록 일부 공정에 종속적인 정보들을 '공정기술 화일'을 통해서 받아들인다. 여기에는 사용될 레이어에 관한 제반 정보들이나 설계 규칙을 비롯하여 다른 설계 정보 형식과의 변환에 관한 정보들이 포함된다.

그러나 사용자가 공정기술 화일의 복잡한 양식을 이해하고 자신의 공정 특성에 맞는 공정기술 화일을 직접 만드는 것은 힘든 일이므로 솔거에서는 대화형 그래픽 공정기술 편집기를 제공한다. 이 패키지는 그래픽 사용자 인터페이스를 통해 공정기술 화일 양식을 모르더라도 쉽게 공정기술 화일을 만들거나 수정할 수 있도록 하며, 공정기술에 관한 정보들 사이의 전체적인 일관성을 자동적으로 유지시킴으로써 '공정기술 화일'의 시맨틱 오류 가능성을 줄인다.

Ⅳ. 결론 및 앞으로의 연구방향

본 논문에서는 종합적인 레이아웃 설계 시스템인 「솔거」에 대하여 설명하였다. 솔거 시스템은 45° 레이아웃의 효율적인 처리를 위하여 45° corner-stitching 자료구조를 사용하였으며 일반적인 레이아웃 편집기능 이외에 다양한 기능을 제공한다. 특히 설계 정보 관리 기능을 이용하여 방대한 양의 설계 정보를 일관성 있고 효율적으로 관리할 수 있으며, 설계규칙 검사와 노드 추출 및 전기적 규칙 검사 기능을 이용하여 설계 과정에서 발생할 수 있는 오류를 단 시간 내에 검사하여 이를 레이아웃 설계 과정에 신속히 반영할 수 있다. 또한 솔거는 시스템의 확장성과 사용자의 편의를 위하여 명령언어 처리 기능과 공정기술 편집 기능을 제공한다.

현재 솔거는 레이아웃 설계에 실제 활용되고 있으며, 향후 솔거 사용자와의 토의를 통하여 설계 과정에서 나타나는 문제점들을 보완하는 작업이 수반되어야 한다. 아울러 레이아웃 설계자가 요구하는 다양한 기능을 첨가하여 보다 종합적인 레이아웃 설계 시스템으로 발전시키기 위한 연구가 필요하다.

參 考 文 獻

[1] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor, "The Magic VLSI layout system", *IEEE Design and Test of Computers*, pp. 19-30, Feb., 1985.

[2] J. K. Ousterhout, "Caesar : an interactive editor for VLSI layouts", *VLSI Systems Design*, vol. 4, pp. 34-38, 1981.

[3] Calma, Calma GDS II reference Manual, 1984.

[4] Valid, Valid SCALD System Reference Manual, 1985.

[5] 한승재, 정성태, 김태형, 이재황, 전주식, "솔거 : VLSI 레이아웃 편집 시스템", 한국정보과학회 '89 봄 학술발표 논문집, pp. 419-422, 1989.

[6] 문의선, 박선주, 한승재, 정성태, 김재범, 이재황, 전주식, "레이아웃 편집기의 소개", 한국정보과학회 정보과학회지 제8권 제3호, pp. 43-54, 1990.

[7] J. K. Bentley, "Mutidimensional binary search trees for associative searching", Communications of ACM, vol. 18, no. 9, pp. 509-517, 1975.

[8] G. Kedem, "The quad-CIF tree : a data structure for hierarchical on-line algorithms", Proceedings of the 19th Design Automation Conference, pp. 352-357, 1982.

[9] J. K. Ousterhout, "Corner stitching : A data-structuring technique for VLSI layout tools", *IEEE Trans. on Computer Aided Design*, CAD-3 no. 1, pp. 87-100, 1984.

[10] 박채령, 전주식, "레이아웃 처리를 위한 45° Corner-stitching 자료 구조의 설계", 정보과학회 봄 학술발표논문집, vol. 16, no. 1, pp. 415-418, 1989.

[11] Ann R. Lanfri, "PHLED45 : An enhanced version of Caesar supporting 45° geometries",

Proceedings of the 21st Design Automation Conference, pp. 586-584, 1984.
[12] Sun Microsystems, SunOS Reference Manual, 1990.
[13] G. S. Taylor, J. K. Ousterhout, "Magic's incremental design rule checker", Proceedings of the 21st Design Automation Conference, pp. 160-165, 1984.

著 者 紹 介

金 宰 範(正會員)

1965年 4月 2日生. 대한전자공학회 정회원. 1987年 2月 서울대학교 컴퓨터공학과 졸업. 1989年 2月 서울대학교 컴퓨터공학과 석사학위 취득. 1991年 8月 서울대학교 컴퓨터공학과 박사과정 수료. 주관심분야 논리 합성, VLSI /CAD 등임.

●

鄭 成 太(正會員)

1965年 3月 23日生. 대한전자공학회 종신회원. 1987年 2月 서울대학교 컴퓨터공학과 졸업. 1989年 2月 서울대학교 컴퓨터공학과 석사학위 취득. 1991年 8月 서울대학교 컴퓨터공학과 박사과정 수료. 주관심분야: Self-timed system, 비동기 회로의 합성 및 검증, 병렬처리 시스템 등임.

李 在 晃(正會員)

1963年 2月 13日生. 대한전자공학회 종신회원. 1985年 2月 서울대학교 컴퓨터공학과 졸업. 1987年 2月 서울대학교 컴퓨터공학과 석사학위 취득. 1988年-1990年 서울대학교 반도체공동연구소 조교. 1989年 8月 서울대학교 컴퓨터공학과 박사과정 수료. 주관심분야: 신경계 자동화, 컴퓨터 그래픽스, VLSI /CAD 등임.

●

全 洲 植(正會員) 第28卷 A編 第12號 參照
현재 서울대학교 컴퓨터공학과 교수