論文93-30A-9-7

# 규칙에 기초한 마스크 레이아웃 변환 시스템의 설계 및 구현

# (Design and Implementation of Rule-based Mask Layout Transformation System)

李 在 晃*, 全 洲 植*

(Jae Hwang Lee and Chu Shik Jhon)

要 約

마스크 레이아웃이 내포하는 지역성의 특성 상 마스크 레이아웃 단계의 전역적인 문제는 여러개의 지역적인 문제로 분할하여, 주어진 마스크 레이아웃의 일부분을 좀더 나은 부분으로 계속적으로 변환시켜 최적에 가까운 전역적인 해에 근접시킴으로써 해결될 수 있다. 본 논문에서는 이러한 개념을 바탕으로, 확장성과 통합성이 결여되어 비효율적임이 지적되고 있는 기존의 마스크 레이아웃 처리 시스템의 문제점을 해결하고자, 마스크 레이아웃의 일부분에 대한 변환이라는 통합된 문제해결의 모형 하에서 대부분의 마스크 레이아웃 문제를 효율적으로 해결할 수 있는 규칙에 기초한 마스크 레이아웃 변환 시스템의 모델을 제안한다. 또한 본 논문에서는 제안된 모델을 구현하여 네트 추출, 마스크 레이아웃 컴팩션, 마스크 레이아웃 편집, 설계규칙 검사 등의 여러가지 문제에 대한 실험 결과를 제시한다. 실험 결과, 본 논문에서 제안한 규칙에 기초한 전문가 시스템을 이용한 방법은 마스크 레이아웃 문제해결에 효과적으로 사용될 수 있을 뿐아니라 기존의 마스크 레이아웃 처리 시스템의 문제점을 해결할 수 있음이 입증되었다.

**Abstract**

Owing to the nature of locality in mask layouts, it appears that most mask layout problems can be solved by transforming a part of the given mask layout into a better layout segment continuously toward a global suboptimal solution. This notion of local transformation addresses major weak points of existing mask layout processing systems, which lack both extensibility and unifiability. This paper attempts to elaborate upon developing a new rule-based mask layout transformation system wherein most of the mask layout problems can be solved under the unified framework of local mask layout transformation. The rule-based mask layout transformation system is applicable to various mask layout problems such as net extraction, mask layout compaction, mask layout editing, and design rule checking. The experimental results show that the rule-based expert system approach is an efficient means of solving those mask layout problems, and thus confronting major drawbacks of existing layout processing systems.

## I. 서론

기존의 마스크 레이아웃 처리 시스템에서는 주어진 마스크 레이아웃에 대한 처리 방법이 알고리듬 형태로 고정되어 있으므로 여러가지 레이아웃 처리 기법

* 正會員, 서울大學校 컴퓨터工學科
(Dept. of Computer Eng., Seoul Nat'l Univ.)

接受日字: 1992年 8月 6日

을 효과적으로 활용하기 어려울 뿐아니라 사용자의 레이아웃 설계 경험이나 공정기술(process technology)로부터 유도되는 레이아웃 특성 및 제약을 레이아웃 처리 과정에 충분히 반영하기 어렵다는 등의 단점이 노출되어 왔다. 이와 같은 기존의 마스크 레이아웃 처리 시스템의 제약점을 완화시키기 위해서는 마스크 레이아웃 처리 방법과 마스크 레이아웃 처리에 관련된 여러가지 인자를 사용자 의도대로 정확하고 손쉽게 정의하여 활용할 수 있는 환경이 제공되어야 한다. 이러한 환경은 사용자가 정의한 특정 분야의 지식을 바탕으로 주어진 문제를 단계적으로 해결하며 해당 분야의 지식이 확충됨에 따라 계속적인 시스템 확장이 가능한, 규칙에 기초한 시스템(rule-based system) 환경에 의해 제공될 수 있다.

규칙에 기초한 시스템을 이용한 접근방법에 대한 연구는 회로 설계에 관련된 여러 분야에서 활발히 진행되어 왔으며 실제로 회로 설계를 위한 전문가 시스템[1,2,3,4,5]이 많이 개발되었다. 그러나 규칙에 기초한 전문가 시스템 환경 하에서 여러가지 마스크 레이아웃 문제를 하나의 통합된 모형으로 해결할 수 있는 전문가 시스템에 대한 연구는 미진한 상태이다.

또한 기존의 마스크 레이아웃 전문가 시스템은 보통 기존의 규칙에 기초한 시스템[6,7]을 이용하여 많이 개발되었다. 그러나 기존의 규칙에 기초한 시스템들은 마스크 레이아웃에 관련된 특성을 표현하고 처리하기 위한 기능이 미흡하고 마스크 레이아웃 변환이 비효율적으로 수행되므로 마스크 레이아웃 문제에 직접적으로 활용되기 어렵다. 뿐만 아니라 기존의 규칙에 기초한 시스템에서는 각각의 규칙 적용시 데이타 메모리에 적재된 문제 정의역 전체를 한꺼번에 고려하므로 문제 정의역이 방대한 마스크 레이아웃 문제해결에 적용될 경우, 일반적으로 지적되고 있는 마스크 레이아웃 문제에 대한 공간 및 시간 복잡도의 문제점이 더욱 심화될 수 있다. 따라서 마스크 레이아웃 문제를 처리하기 위해서는 기존의 규칙에 기초한 시스템의 형태를 탈피한 새로운 형태의 규칙에 기초한 시스템이 요구된다.

그리고 최근에 발표된 CAD 소프트웨어들은 이전의 특정 설계 단계에서 특정 작업만을 수행시키는 형태를 탈피하여 여러 회로 설계 단계들을 유기적으로 통합시키는 보다 진보된 형태를 제공하고 있다. 즉, VLSI 회로 설계 단계에 있어서의 편집, 시뮬레이션, 검증 및 분석 등은 물론 각 단계 간의 유기적인 통합을 통하여 합성, 추출, 회로 비교 등의 기능뿐아니라 방대한 양의 설계 정보들을 효율적으로 관리하는 기능들도 제공하고 있다. 이러한 경향은 이전의 CAD 소프트웨어들이 안고 있던 다른 설계 도구들 간의 인터페이스 부재로 인한 설계의 비효율성과 같은 문제점들을 극복하기 위하여 필수적이라 볼 수 있다.

위와 같은 맥락에서, 본 논문에서는 마스크 레이아웃 단계에서의 여러가지 레이아웃 문제들을 효율적으로 처리할 수 있는 규칙에 기초한 마스크 레이아웃 변환 시스템을 구축하고자 한다. 이 시스템은 마스크 레이아웃 설계의 효율성을 제공하기 위하여 마스크 레이아웃 단계에서의 주요 문제들을 하나의 통합된 모형으로 해결할 수 있으며, 마스크 레이아웃 문제해결에서 공통적으로 지적되고 있는 공간 및 시간 복잡도의 문제점을 완화시킬 수 있다.

본 논문의 Ⅱ절에서는 기존의 규칙에 기초한 시스템과는 달리, 마스크 레이아웃 변환 지식의 표현 및 처리를 위하여 고안된 규칙에 기초한 시스템의 새로운 모델을 제안한다. Ⅲ절에서는 제안된 모델을 바탕으로 구현한 규칙에 기초한 마스크 레이아웃 변환 시스템에 대하여 개괄적으로 설명하며, 기존의 주요 마스크 레이아웃 문제들에 대한 실험 결과를 제시한다. 또한 시간 및 공간 복잡도, 마스크 레이아웃 변환 정의의 간결성 등의 관점에서, 구현된 규칙에 기초한 마스크 레이아웃 변환 시스템과 기존의 규칙에 기초한 시스템을 비교한다. 마지막 Ⅳ절에서는 본 논문의 결론을 내리고 앞으로의 연구방향에 대하여 기술한다.

## Ⅱ. 규칙에 기초한 마스크 레이아웃 변환 시스템 모델

### 1. 개요

본 모델의 기본 아이디어는 마스크 레이아웃의 일부분을 계속적으로 변환시킴으로써 주어진 마스크 레이아웃 문제를 해결할 수 있다는 부분변환[8,9]의 개념에 바탕을 두고 있으며, 이는 규칙에 기초한 시스템 하에서의 문제해결 방법과 유사하다. 그러나 기존의 주요 마스크 레이아웃 처리 알고리듬에 대한 고찰과 레이아웃 설계 전문가와의 토의 결과, 규칙에 기초한 시스템 하에서 이러한 본 논문의 기본 아이디어를 근거로 마스크 레이아웃 문제들을 해결하기 위해서는 마스크 레이아웃이라는 특정 분야의 지식 표현과 이에 대한 처리 기능이 제공되어야 함이 분석되었다. 즉, 마스크 레이아웃 지식을 효과적으로 표현 및 처리하기 위해서는 마스크 레이아웃 정보에 접근하기 위한 기능, 마스크 레이아웃 변환을 나타내기 위한 마스크 레이아웃 추상화 및 변환 기능, 규칙적용을 주어진 마스크 레이아웃의 일부분으로 국한시킬 수 있는 마스크 레이아웃 지역화 기능, 규칙적용에 관련

된 제어정보를 외연적으로 지정할 수 있도록 하는 기능 등이 제공되어야 한다. 그러나 이러한 요건들의 대부분은 범용으로 사용되는 기존의 규칙에 기초한 시스템에서는 제공되지 않으므로 기존의 규칙에 기초한 시스템을 마스크 레이아웃 문제해결에 이용하기에는 어려움이 따른다. 따라서 본 절에서는 마스크 레이아웃 변환 지식의 표현 및 처리를 위한 기능을 중심으로 마스크 레이아웃 문제해결을 위한 규칙에 기초한 시스템의 새로운 모델을 제안한다. 그림 1은 본 논문에서 제안하는 마스크 레이아웃 변환을 위한 규칙에 기초한 시스템의 개략적인 구성을 나타낸다.

그림 1의 규칙에 기초한 시스템 구성 하에서 마스크 레이아웃 변환 지식은 주어진 마스크 레이아웃의 일부분에 대한 변환을 나타내는 규칙(rule)과 정의된 규칙을 적용하는 순서를 지정하는 규칙적용 제어정보(rule application control information)로 정의된다. 또한 마스크 레이아웃 관리기는 주어진 마스크 레이아웃을 데이타베이스 기법을 이용하여 관리하며 추론기와 마스크 레이아웃 데이타베이스 사이의 인터페이스를 제공한다. 그리고 추론기는 변환시키고자 하는 마스크 레이아웃의 일부분을 마스크 레이아웃 데이타베이스로부터 추출하여 데이타 메모리에 적재하고, 사용자가 정의한 규칙적용 제어정보를 바탕으로 마스크 레이아웃 변환 규칙을 적용하여 데이타 메모리에 적재된 마스크 레이아웃의 일부분을 변환하는 과정을 계속적으로 수행함으로써 주어진 마스크 레이아웃 문제를 해결할 수 있다.

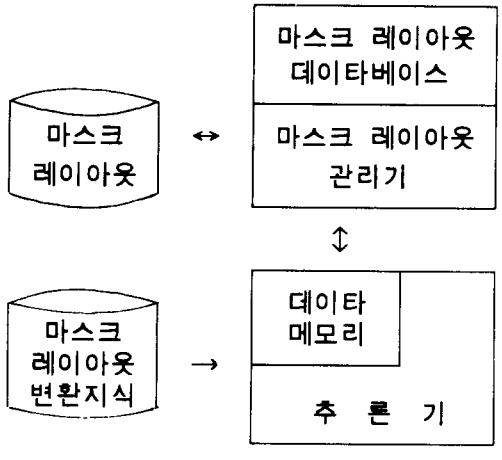



그림 1. 규칙에 기초한 마스크 레이아웃 변환 시스템 모델의 구성

Fig. 1. Configuration of Rule-based Mask Layout Transformation System Model.

본 논문에서 제안하는 규칙에 기초한 시스템 모델은 그림 1의 시스템 환경을 구성하는 추론기의 동작구조, 추론기가 제공해야 할 마스크 레이아웃 변환 지식의 표현 및 처리를 위한 여러가지 기능, 마스크 레이아웃 및 데이타 메모리 관리 등으로 총체적으로 정의된다. 본장의 각 절에서는 이들 각각에 대하여 자세히 설명한다.

## 2. 추론기의 동작구조

그림 2는 본 논문에서 제안하는 규칙에 기초한 시스템 모델 중 추론기의 동작구조를 나타낸다. 그림 2의 명령어 및 변환 지식 컴파일 단계에서는 사용자가 지정한 명령어와 마스크 레이아웃 변환 지식을 컴파일하여 내부 구조로 저장한다. 이때 사용자로부터 정의된 변환 지식을 적용하라는 명령어가 주어지면 저장된 규칙, 규칙적용 제어정보, 데이타 메모리 등을 바탕으로 규칙선택, 규칙매치, 규칙적용의 단계를 순환한다.

규칙선택 단계에서는 해당 규칙적용 제어정보에 정의된 순서에 입각하여 다음에 적용할 하나의 규칙을 선택한다.

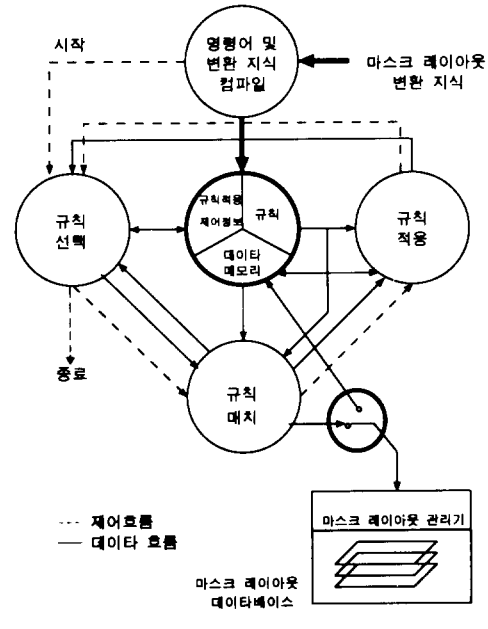



그림 2. 추론기의 동작구조

Fig. 2. Operational Architecture of the Inference Engine.

그리고 규칙매치 단계에서는 규칙선택 단계에서 선

택된 규칙이 현재 시점에서 데이타 메모리의 정보(주어진 마스크 레이아웃의 일부분과 기타 정보를 나타내는 마스크 레이아웃 오브젝트)와 부합되어 적용가능한지 여부를 판별한다. 이때 이전의 순환 단계와는 다른 부분(마스크 레이아웃의 일부분)에 대하여 선택된 규칙을 적용하고자 할 경우 데이타 메모리에 적재된 마스크 레이아웃 정보를 마스크 레이아웃 데이타베이스로 저장하고 변화된 마스크 레이아웃 데이타베이스로부터 새로운 부분을 추출하여 데이타 메모리에 적재한다. 규칙매치 단계와 마스크 레이아웃 관리기 사이의 스위치는 이러한 마스크 레이아웃 정보 교환이 선택적임을 나타낸다.

또한 규칙적용 단계에서는 규칙매치 단계에서 적용가능하다고 판별된 규칙을 적용한다. 이때 데이타 메모리에 저장된 정보는 규칙이 적용됨에 따라 갱신되며, 추론기의 다음 순환에서는 변화된 데이타 메모리 정보를 이용하게 된다. 이러한 규칙선택, 규칙매치, 규칙적용 등의 순환 단계는 종료 명령이 수행되거나 더이상 적용가능한 규칙이 존재하지 않을 경우에 종료된다.

### 3. 마스크 레이아웃 오브젝트에 대한 연산

마스크 레이아웃 오브젝트란 마스크 사각형, 마스크 레이아웃 경계사각형(bounding box), 설계규칙 정보, technology 관련 정보 등과 같이 마스크 레이아웃 단계에서 사용되는 단위정보를 의미한다. 따라서 마스크 레이아웃 변환 지식을 기술하기 위해서는 무엇보다도 먼저 마스크 레이아웃 단계에서의 여러가지 오브젝트에 대한 구조정의(define), 생성(make), 삭제(delete), 검색(get), 갱신(modify) 등의 연산이 필수적으로 지원되어야 한다. 아래에 이들 각각의 연산에 대하여 자세히 설명한다.

먼저, *define* 연산은 지정된 속성을 가지는 오브젝트 구조를 정의한다. 예를 들어 본 모델에서는 마스크 레이아웃을 수직과 수평 변으로 구성되는 마스크 사각형의 집합으로 정의하며, 이에 따라 각각의 마스크 사각형은 아래와 같이 좌측하단 꼭지점(xbot, ybot), 우측상단 꼭지점(xtop, ytop), 마스크 타입(type) 등의 속성(attribute)과, 여러가지 마스크 레이아웃 문제를 처리할 수 있도록 지원하기 위하여 해당 마스크 사각형의 네트 번호를 나타내는 net와 사용자가 정의하여 사용하는 user 등의 속성으로 정의할 수 있다.

```
define (box) {
    integer xbot; integer xtop;
    integer ybot; integer ytop;
    string type; integer user;
    integer net;
}
```

또한 *make* 연산은 해당 오브젝트를 생성하여 데이타 메모리에 저장한다. 예를 들어 속성값이 이미 지정된 임의의 오브젝트 x는 "*make*(x)"로 생성되어 데이타 메모리에 저장되며, 또한 polysilicon 타입의 마스크 사각형 사이의 최소이격거리가 2임을 규정하는 설계규칙을 나타내는 오브젝트는 다음과 같이 생성할 수 있다.

*make*(min_spacing type1 "polysilicon" type2 "polysilicon" spacing 2)

그리고 *delete* 연산은 해당 오브젝트를 데이타 메모리로부터 삭제한다. 예를 들어 어떤 규칙에 매치된 임의의 오브젝트 x는 "*delete*(x)"로 간단히 삭제할 수 있다.

뿐만 아니라, *get* 연산은 지정된 오브젝트의 해당 속성값을 검색한다. 예를 들어 오브젝트 x가 마스크 사각형 구조(box)로 지정되었다고 가정할 경우, x의 좌측하단 꼭지점의 X축 좌표는 "*get*(x xbot)"으로 검색할 수 있다.

또한 *modify* 연산은 지정된 오브젝트의 해당 속성값을 갱신한다. 예를 들어 오브젝트 x가 마스크 사각형 구조(box)로 지정되었다고 가정할 경우, x의 마스크 타입을 metal1으로 갱신하는 연산은 "*modify*(x type "metal1")"와 같이 표현된다.

### 4. 마스크 레이아웃 변환의 모형

기존의 마스크 레이아웃 처리 시스템에서와 같이 마스크 레이아웃 단계의 특정 문제만을 효율적으로 해결하기 위하여 특정 문제 전용의 구조나 연산을 지원하는 시스템은 이러한 특정 문제를 해결하는데만 사용될 수 있다. 또한 현재 쟁점이 되고 있는 문제 전용의 구조와 연산을 종합하여 지원한다 하더라도 해당 문제를 해결하는 방법이 이러한 구조와 연산에 많이 의존하므로, 기존의 마스크 레이아웃 처리 시스템과 마찬가지로 문제해결 방법이 고정된다는 문제점이 발생한다. 뿐만 아니라 이러한 구조와 연산은 소수의 특정 문제만을 지원하므로 새로운 문제를 해결하는데 효과적으로 대처할 수 없다.

한편 네트 추출, 마스크 레이아웃 컴팩션, 마스크 레이아웃 편집, 설계규칙 검사 등의 마스크 레이아웃

단계에서의 주요 문제들은 근본적으로는 마스크 레이아웃에 대한 기하학적인 특성 및 제약을 처리하는 문제이다. 그리고 이러한 문제들을 해결하는 기존의 알고리듬에서는 대부분 마스크 레이아웃 평면 상의 기하학적인 연산에 바탕을 두고 있다. 따라서 마스크 레이아웃의 기하학적인 의미와 마스크 레이아웃 단계에서의 여러가지 문제의 특성을 고려할 때, 마스크 레이아웃 문제에 대한 공통된 문제해결의 모형은 기하학적인 변환이다.

그러므로 본 논문에서 제안하는 규칙에 기초한 시스템에서는 마스크 레이아웃 단계의 여러가지 문제들을 통합된 모형 하에서 효과적으로 해결하기 위하여 특정 문제 전용의 구조나 연산이 아니라 아래와 같이 임의의 마스크 레이아웃 패턴 Pi를 새로운 패턴 Pj로의 기하학적인 변환을 위한 연산만을 지원한다. 따라서 마스크 레이아웃에 대한 기하학적인 특성 및 제약을 처리하는 모든 문제는 본 논문에서 제안한 규칙에 기초한 마스크 레이아웃 변환 시스템을 이용하여 해결할 수 있을 것이다.

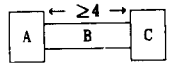
$$P_i \rightarrow P_j$$

아래에서는 마스크 레이아웃 패턴 Pi와 Pj의 표현 방법을 설명함으로써 본 모델에서 제안하는 마스크 레이아웃 변환에 바탕을 둔 기본적인 문제해결의 모형을 제시한다.

1) 마스크 레이아웃 추상화

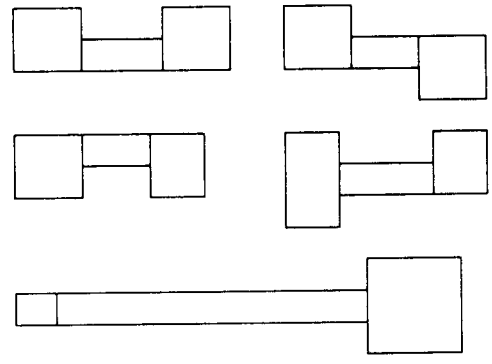
본 시스템 모델에서는 마스크 레이아웃 패턴을 추상적으로 표현하는 기능을 지원한다. 즉, 마스크 레이아웃 오브젝트의 속성값을 검색하는 get 연산과 eq(=) ne(≠) gt(〉) lt(〈) ge(≥) le(≤) 등의 관계연산, 산술연산자 -혹은 + 등을 이용하여 마스크 레이아웃 사각형 사이의 기하학적인 연관관계를 지정함으로써 마스크 레이아웃 패턴을 추상적으로 표현할 수 있다. 그림 3은 마스크 레이아웃 추상화의 예를 나타낸다.

그림 3-(a)에서는 그림 3-(b)와 같이 A와 C가 4 이상 이격되고 A와 B 그리고 B와 C가 서로 기하학적으로 인접한 마스크 레이아웃 패턴들을 추상적으로 나타낸다.




```
eq(get(A,xtop) get(B,xbot));    eq(get(B,xtop) get(C,xbot));
le(get(A,ybot) get(B,ybot));    lt(get(B,ybot) get(A,ytop));
le(get(C,ybot) get(B,ybot));    lt(get(B,ybot) get(C,ytop));
lt(get(A,ybot) get(B,ytop));    le(get(B,ytop) get(A,ytop));
lt(get(C,ybot) get(B,ytop));    le(get(B,ytop) get(C,ytop));
ge(get(C,xbot)-get(A,xtop) 4);
```

(a) 추상적인 레이아웃 표현



(b) 마스크 레이아웃 패턴

그림 3. 마스크 레이아웃 추상화의 예
Fig. 3. Example of Mask Layout Abstraction.

한편, 조건을 나타내는 관계연산의 좌우로의 이항을 고려할 경우 +와 -연산자 중 하나만을 이용하여 +와 항 모두를 표현할 수 있다. 또한 관계연산의 좌우 항의 위치를 서로 바꾸는 경우를 고려한다면 관계연산 lt(gt)는 gt(lt)로, 그리고 le(ge)는 ge(le)로 나타낼 수 있으므로 마스크 사각형 사이의 기하학적인 연관관계는 관계연산 gt와 ge(혹은 gt와 le, lt와 ge, lt와 le)를 이용하여 모두 표현할 수 있다. 따라서 레이아웃 추상화의 방편으로 사용된, get 연산과 관계연산 eq ne gt ge, 산술연산자 -등만을 이용하여 임의의 갯수 및 형태로 구성되는 마스크 레이아웃 패턴을 모두다 표현할 수 있다. [10]

2) 마스크 레이아웃 변환

개념적인 마스크 레이아웃 변환 「$P_i \rightarrow P_j$」에서, $P_i$와 $P_j$는 앞서 설명한 마스크 레이아웃 추상화 기능에 의거하여 동일한 방식으로 표현할 수 있다. 그러나 이 경우 $P_i$와 $P_j$는 마스크 사각형 사이의 기하학적인 연관관계로 표현되므로 $P_i$를 $P_j$로 변환하기 위해서는 배치, 크기 결정, 마스크 레이아웃 제약검사(예를 들어 설계규칙 검사) 등의 과정을 거쳐야 한다. 따라서 「$P_i \rightarrow P_j$」의 마스크 레이아웃 변환을 처리하는 과정에 사용자의 의도와는 상이한 연산이 포함될 수 있으므로 마스크 레이아웃 패턴 $P_i$와 $P_j$를 모두 추상화된 방식으로 표현하는 것은 바람직하지 못하다.

이에 따라 본 논문에서 제안하는 규칙에 기초한 시스템 모델에서 마스크 레이아웃 패턴 $P_i$는 아래와 같이 $P_i$에 대한 변환 $\boldsymbol{\theta}(P_i)$로 표현한다.

$$P_i \rightarrow \boldsymbol{\theta}(P_i)$$

여기에서 $\theta$는 추상화된 마스크 레이아웃 패턴 $P_i$에 대한 변환 함수이며, 앞서 설명한 마스크 레이아웃 오브젝트에 접근하기 위한 연산 중 make, delete, modify 등의 집합으로 구성된다.

$\theta$ = { make-연산, delete-연산, modify-연산 }

한편 개념적인 마스크 레이아웃 변환에 있어서 주어진 임의의 마스크 레이아웃 패턴 $P_i$를 $P_j$로 변환하는 과정에는 최소한 삽입, 삭제, 갱신 등이 필요하며, 역으로 이러한 삽입, 삭제, 갱신 등의 연산만이 제공된다면 주어진 패턴을 원하는 패턴으로 임의로 변환시킬 수 있다. 따라서 본 모델에서 모형화한 마스크 레이아웃 변환을 위한 변환 함수 $\theta$의 연산만을 이용하여 모든 마스크 레이아웃 패턴 변환을 정의할 수 있다. [10]

5. 규칙적용 제어언어

본 모델에서는 정의된 규칙을 적용하는 순서를 사용자가 지정할 수 있도록 지원하기 위하여 규칙적용 제어정보를 기술하는 규칙적용 제어언어를 제공한다.

규칙적용 제어정보란 정의된 규칙들중 현재 상황에서 적용가능한 하나의 규칙을 선택하기 위한 판단근거를 일컫는다. 규칙적용에 관련된 제어정보가 추론기 내부에 고정되어 있는 경우 특정 응용에서 필요로 하는 제어정보가 제공되지 못할 경우가 많다. 따라서 사용자가 의도하는 규칙적용 과정을 효과적으로 수행하기가 어려우며 사용자가 원하는 목표에 도달하지 못하는 경우도 발생할 수 있다. 특히 마스크 레이아웃 문제들의 경우, 주어진 문제의 결과와 문제해결에 소요되는 시간이 규칙적용 제어정보에 크게 좌우되므로 사용자에게 규칙적용 제어정보를 정의할 수 있도록 융통성을 부여하는 것이 필수적이다. 이에 따라 기존의 규칙에 기초한 시스템에서는 절차적 언어 혹은 정규수식(regular expression)을 이용하여 규칙적용에 관련된 제어정보를 기술하도록 지원하는 추세에 있다. [6,7,11]

본 모델에서는 사용자로 하여금 *RULE*, *IF-THEN-ELSE*, *REPEAT*, *DO* 블럭, 제어이전을 위한 프리미티브 등의 제어언어의 구조를 이용하여 문제해결에 필요한 규칙적용 제어정보를 외연적으로 정의할 수 있도록 허용한다.

*RULE* 구조에서는 지정된 규칙이 적용가능할 경우 이를 적용함을 나타낸다.

*DO* 블럭은 { } 내에 다른 제어구조를 포함하는 복합적인 구조로 정의되며 각각의 블럭에는 블럭 내에 지정된 제어구조를 적용하는 방법이 지정된다. 이러한 블럭 적용방법으로는 *ordered*와 *default*가 사용된다. 즉, *ordered* 적용방법은 블럭 내에 지정된 순서가 제어구조 사이의 우선순위를 나타냄을 의미하며 default는 별도의 적용방법을 지정하지 않고 추론기의 제어정보에 의거하여 지정된 제어구조를 수행함을 나타낸다. 또한 각 블럭에는 해당 블럭으로부터 선택되어 수행될 제어구조의 갯수가 하나(*one*), 가능한 많이(*many*), 모두(*all*) 등으로 지정될 수 있다.

*IF-THEN-ELSE* 구조에서는 *IF* 부분의 규칙이 적용가능할 경우 *THEN* 부분의 제어구조를 수행하고 그렇지 않을 경우에는 *ELSE* 부분의 제어구조를 수행함을 의미한다.

*REPEAT* 구조는 해당 제어구조를 가능한한 지정된 횟수만큼 반복 수행함을 나타낸다.

또한 정의된 규칙적용 제어정보를 처음부터 다시 수행함(*RESTART*), 정의된 제어정보의 임의의 위치로의 제어이전(*GOTO*), *REPEAT* 제어구조로부터의 이탈(*BREAK*), *REPEAT* 제어구조의 시작부분으로의 제어이전(*CONTINUE*), *DO* 블럭으로부터의 이탈(*EXIT*), 규칙적용 종료(*STOP*) 등을 나타내는 프리미티브 제어구조가 사용된다. 예를 들어 아래와 같은 규칙적용 제어정보를 살펴보자.

```
control (example)
{
 REPEAT forever DO all ordered ------(1)
  {
   IF RULE r1 THEN STOP; ELSE RULE r2; -- (2)
   DO one default {RULE r3; RULE r4;} ---- (3)
  }
}
```

위의 예에서는 example이라는 명칭을 가지는 규칙적용 제어정보가 정의되었다. (1)의 REPEAT 제어구조에서는 해당 DO 블럭을 반복 수행함을 나타낸다. 또한 이 DO 블럭은 지정된 내부의 제어구조를 지정된 순서에 의거하여 모두 수행함을 나타내므로 제어구조 를 수행한 다음 을 수행해야 한다. (2)에서는 규칙 r1이 적용가능할 경우 이를 적용한 다음 규칙적용을 종료하고, 규칙 r1이 적용가능하지 않을 경우 규칙 r2를 적용함을 나타낸다. 그리고 (3)은 추론기의 내부 제어정보에 의거하여 지정된 규칙 r3와 r4 중 하나를 선택하여 적용함을 의미한다.

6. 마스크 레이아웃 지역화

일반적으로 마스크 레이아웃 단계의 문제는 그 정의역(마스크 레이아웃)이 매우 방대하고 문제를 해결하는 알고리듬의 시간 복잡도가 상당히 크므로 주어진 문제 정의역 전체를 한꺼번에 고려할 경우 문제해결에 과도한 시간이 소요될 수 있다. 이에 따라 기존의 레이아웃 처리 알고리듬에서는 주어진 마스크 레이아웃 전체에 대한 문제를 레이아웃의 일부분에 대한 여러개의 작은 문제로 분할하여 해결한다. 이러한 문제 정의역 분할에는 윈도우(window)나 scan-line [12] 등이 많이 사용된다. 즉, 윈도우나 scan-line을 이용하여 마스크 레이아웃을 탐색하면서 이들과 중첩되는 일부분의 마스크 사각형들만을 처리함으로써 현 단계에서 고려해야할 문제의 정의역을 축소할 수 있다. 본 논문에서는 이와 같은 기법을 마스크 레이아웃 지역화(mask layout localization)라 한다.

본 논문에서 제안하는 규칙에 기초한 시스템 모델에서는 마스크 레이아웃 지역화를 위하여 윈도우의 개념을 제안한다(scan-line은 상하 혹은 좌우 변의 좌표가 동일한 윈도우로 정의할 수 있음). 윈도우는 수직 및 수평 변을 갖는 사각형의 형태이며, 주어진 방대한 마스크 레이아웃에 대한 규칙매치 과정은 이 윈도우와 중첩되는 마스크 사각형들로 국한된다. 또한 마스크 레이아웃 변환 규칙이 적용될 부분을 사용자 의도대로 선택할 수 있도록 지원하기 위하여 각 규칙에서 윈도우의 크기와 위치를 손쉽게 변화시킬 수 있어야 한다. 이에 따라 사용자는 윈도우를 이용하여 주어진 방대한 마스크 레이아웃을 탐색하면서 의도하는 부분(윈도우)에 원하는 규칙을 효과적으로 적용시킬 수 있다.

### 7. 마스크 레이아웃 데이타베이스

최근 방대한 양의 정보를 효율적으로 관리하기 위하여 규칙에 기초한 시스템과 데이타베이스 관리 시스템을 연계시키는 연구가 활발히 진행되고 있다. 따라서 마스크 레이아웃 변환을 위한 규칙에 기초한 시스템에서도 주어진 문제의 정의역(마스크 레이아웃)을 모두 데이타 메모리에 저장하는 방식을 탈피하여 이러한 데이타베이스와의 연계를 통하여 데이타 메모리를 효율적으로 관리하는 방법을 도입하는 것이 바람직하다.

본 논문에서 제안하는 규칙에 기초한 시스템 모델에서는 앞서 설명한 바와 같이 마스크 레이아웃 지역화 기능을 이용함으로써 규칙매치 및 적용 과정을 현재의 윈도우와 중첩되는 마스크 사각형들로 국한시킨다. 따라서 규칙에 기초한 시스템과 데이타베이스를 연계시키기 위해서, 추론기는 변환 규칙의 처리 과정에서 윈도우가 변경될 때마다 마스크 레이아웃의 이전 윈도우 부분을 제거하고 데이타 메모리에 적재된 마스크 사각형 오브젝트들을 마스크 레이아웃 데이타베이스로 저장하며, 다시 주어진 마스크 레이아웃으로부터 현재의 윈도우와 중첩되는 마스크 사각형들을 추출하여 데이타 메모리에 적재하는 등의 작업을 수행한다. 그러므로 마스크 레이아웃 데이타베이스에 대한 작업으로는 삭제 및 삽입 작업과 현재의 윈도우 내에 중첩되는 마스크 사각형을 추출하기 위한 탐색 작업이 주종을 이룬다.

따라서 본 모델의 마스크 레이아웃 데이타베이스 구조는 이러한 삭제, 삽입, 탐색 작업을 효율적으로 지원할 수 있어야 한다. 또한 마스크 레이아웃 데이타베이스를 관리하는 마스크 레이아웃 관리기는 마스크 레이아웃 데이타베이스에 대한 이러한 주요 작업을 효율적으로 수행해야 할 뿐아니라 이러한 작업에 대한 일관된 인터페이스를 제공해야 한다.

## Ⅲ. 마스크 레이아웃 설계 전문가 시스템

본 논문에서는 앞서 제안된 규칙에 기초한 마스크 레이아웃 변환 시스템 모델을 구현하였다. 이 시스템은 SUN 워크스테이션의 UNIX 환경 하에서 40,000 줄의 "C" 프로그래밍 언어로 구현되었으며 제안된 모델이외에 기존의 규칙에 기초한 시스템이 제공하는 여러가지 특성을 반영하였다. 본 절에서는 구현된 규칙에 기초한 시스템을 개략적으로 살펴보고, 이를 기존의 규칙에 기초한 전문가 시스템 개발 도구와 비교한다. 또한 구현된 시스템을 기존의 여러가지 마스크 레이아웃 문제해결에 적용한 경우의 실험 결과를 제시한다.

### 1. 시스템 구성

본 논문의 Ⅱ절에서 제안한 모델을 실제 구현한 규칙에 기초한 시스템의 구성은 그림 4와 같다.

그림 4의 규칙에 기초한 마스크 레이아웃 변환 시스템은 마스크 레이아웃 변환기와 마스크 레이아웃 관리기의 두부분으로 크게 나눌 수 있다.

마스크 레이아웃 변환기의 변환 지식 컴파일러는 사용자가 정의한 마스크 레이아웃 변환 지식(규칙 및 규칙적용 제어정보)을 컴파일하여 내부 구조를 생성한다. 또한 추론기는 규칙, 규칙적용 제어정보, 데이타 메모리 등에 의거하여 규칙선택-규칙매치-규칙적용 등으로 이루어지는 데이타 메모리의 상태 변환을 반복수행함으로써 주어진 마스크 레이아웃 문제를 해결한다. 그리고 Ⅱ절의 모델에서 제안한 마스크 레이

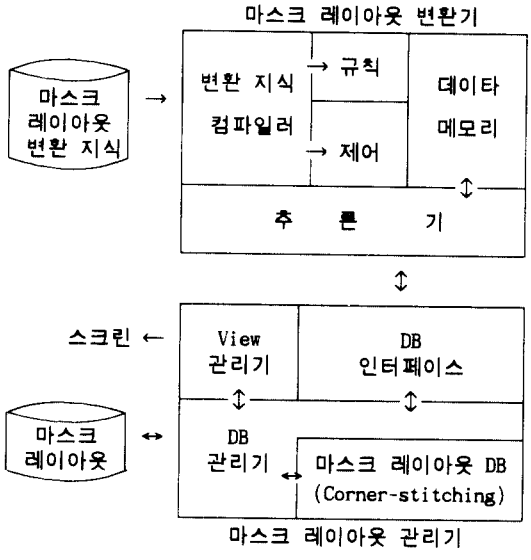



그림 4. 규칙에 기초한 마스크 레이아웃 변환 시스템의 구성

Fig. 4. Configuration of the Rule-based Mask Layout Transformation System.

아웃 오브젝트에 접근하기 위하여 필수적으로 제공되어야 할 연산이외에 사용자의 지식 표현을 효과적으로 지원하기 위하여 여러가지 다양한 연산이 제공된다.

또한 마스크 레이아웃 관리기의 DB 관리기는 주어진 마스크 레이아웃을 내부 레이아웃 데이타베이스 구조(corner-stitching [13] 구조)로 유지 및 관리한다. 이러한 corner-stitching 구조에서는 마스크 사각형들이 중첩되어 트랜지스터 혹은 contact과 같은 회로 요소가 생성될 경우 이들 회로 요소를 독립된 타일로 관리하므로 마스크 레이아웃으로부터 심볼릭 단계의 의미를 유추하기가 용이하다. 그리고 각각의 마스크 사각형이 최대 수평통합의 원칙에 의거하여 타일로 분할 및 통합되어 일관성있게 관리되므로 여러가지 마스크 레이아웃 변환을 정의하는데 효과적으로 사용될 수 있다. 또한 corner-stitching 구조에서는 삽입, 삭제, 탐색 작업이 효율적으로 수행되므로 마스크 레이아웃 지역화를 효과적으로 지원할 수 있다. 이에 따라 본 논문에서는 마스크 레이아웃 데이타베이스의 구조로 corner-stitching 레이아웃 구조를 채택하였다.

View 관리기는 현재의 마스크 레이아웃 상태를 화면에 그래픽하게 보여준다. 그리고 DB 인터페이스는 마스크 레이아웃 변환기와 마스크 레이아웃 관리기 사이의 정보교환을 지원한다. 이를 위하여 하나의 마스크 사각형을 corner-stitching 레이아웃 구조에 삽입하는 함수와 주어진 영역 내의 마스크 사각형들을 삭제하는 함수, 그리고 지정된 영역과 중첩되는 마스크 사각형들을 탐색하는 함수 등이 지원된다.

2. 구현된 규칙에 기초한 시스템 평가

본 절에서는 범용으로 사용되는 기존의 규칙에 기초한 시스템과 본 논문에서 구현한 시스템에 대하여 시간 및 공간 복잡도, 마스크 레이아웃 변환 정의의 간결성 등을 비교한다. 기존의 규칙에 기초한 시스템으로는 OPS5 [6] 를 선택하여 비교한다. 결론적으로, 본 논문에서 구현한 규칙에 기초한 시스템은 주어진 마스크 레이아웃 문제를 해결하기 위한 규칙 정의가 간결 및 용이하고 규칙 처리가 효율적이므로 OPS5와 같이 범용으로 사용되는 기존의 규칙에 기초한 시스템에 비해서 주어진 마스크 레이아웃 문제를 해결하는데 효과적으로 사용될 수 있다.

1) 시간 및 공간 복잡도의 비교

본 논문에서 구현한 시스템과 OPS5의 시간 및 공간 복잡도를 비교하기 위하여 아래와 같이 가정한다.

문제 정의역 : N개의 마스크 사각형으로 구성된 마스크 레이아웃
규칙 수 : R
규칙에 매치되는 오브젝트의 평균 수 : r
윈도우 내의 오브젝트의 평균 수 : W (W≤N)
규칙의 평균 조건 수 : C

앞서 설명한 바와 같이 구현된 시스템의 동작구조는 "규칙선택-규칙매치-규칙적용" 과정의 순환 구조이며, OPS5에서는 "규칙매치-규칙선택-규칙적용" 과정을 반복 수행한다. 이들 시스템에서, 규칙선택과 규칙적용 과정의 시간 복잡도는 O(1)이며 규칙매치 과정에 대부분의 시간이 소요된다. 따라서 구현된 시스템과 OPS5의 시간 복잡도를 결정하는 규칙매치 과정을 중심으로, 구현된 시스템과 OPS5의 시간 복잡도를 비교하면 아래와 같다.

OPS5의 시간 복잡도 : $O(R \times N^{2r-1})$ [14]
구현된 시스템의 시간 복잡도 : $O(W')$ [10]

위의 시간 복잡도 비교에서 보듯이, OPS5와는 달리 본 논문에서 구현한 시스템에서 매치하고자 하는 규칙의 수는 항상 1이므로 규칙의 수는 규칙매치의

시간에 영향을 미치지 않으며, 문제해결에 소요되는 시간이 문제 정의역의 크기 N에 비례해서 급격히 증가하지 않는다.

또한 주어진 문제를 해결하기 위하여 OPS5와 본 논문에서 구현한 시스템에서 사용되는 공간(기억장소)을 비교하면 아래와 같다.

OPS5의 공간 복잡도 : $O(R \times N^r)$ [14]
구현된 시스템의 공간 복잡도 : $O(R+N)$ [16]

위의 공간 복잡도 비교에서 보듯이, 본 논문에서 구현한 규칙에 기초한 시스템에서는 OPS5에 비해서 상당히 적은 양의 공간만을 필요로 함을 알 수 있다.

2) 마스크 레이아웃 변환 정의의 간결성 비교

범용으로 사용되는 기존의 규칙에 기초한 시스템에 비해서 본 논문에서 구현한 시스템을 이용할 경우, 논리적으로 동일한 마스크 레이아웃 변환 문제를 해결하는 규칙 정의가 상당히 간결해 질 수 있다. 예를 들어 기존의 대표적인 범용 시스템인 OPS5와 구현된 시스템에 대하여 아래와 같은 세 개의 마스크 레이아웃 변환 문제를 해결하는데 필요한 규칙의 수를 비교하면 표 1 [16] 과 같다.

문제 ① 중첩되지 않는 두 개의 마스크 사각형 사이의 거리가 4보다 작은 패턴을 표시
문제 ② 중첩되지 않는 두 개의 마스크 사각형 사이의 거리가 4보다 작은 패턴 사이의 공간을 표시
문제 ③ 수직 폭이 4보다 큰 마스크 사각형의 수직 폭을 4로 조정

표 1. 논리적인 변환 문제 ①②③에 대한 규칙의 수 비교
Table 1. Number of Rules for solving Logical Transformation Problems ①, ②, and ③.

| 마스크 레이아웃 변환 / 규칙에 기초한 시스템 | 문제 ① | 문제 ② | 문제 ③ |
|---|---|---|---|
| OPS5를 이용한 경우 | 33 | 33 | 6 |
| 구현된 시스템을 이용한 경우 | 1 | 8 | 1 |

이와 같이 구현된 규칙에 기초한 시스템을 이용할 경우, 마치 프로그래밍 언어를 이용하여 프로그램을 작성하는 것과 유사하게 어떤 기능을 제공하는 규칙과 이를 제어하는 제어정보를 분리하여 정의하므로 마스크 레이아웃 단계에서의 다양한 변환을 쉽고도 간결하게 정의할 수 있다. 더우기 앞서 비교한 규칙매치에 대한 시간 복잡도에서 규칙 수의 영향을 고려할 경우, 구현된 시스템은 OPS5보다 훨신 효율적임을 알 수 있다.

3. 실험 결과

본 절에서는 네트 추출, 마스크 레이아웃 컴팩션, 마스크 레이아웃 편집과, 설계규칙 검사 등의 주요 마스크 레이아웃 문제를 본 논문에서 구현한 규칙에 기초한 마스크 레이아웃 변환 시스템에서 실험한 결과를 보임으로써 Ⅱ절에서 제안한 모델의 타당성를 입증하고 이 모델을 실제 구현한 본 시스템의 유용성을 보이고자 한다.

1) 네트 추출

본 논문에서 실험한 네트 추출 알고리듬은 아래와 같다.

(알고리듬) 네트 추출
(1) 네트가 지정되지 않은 마스크 사각형을 선택하여 네트 번호를 할당
(2) 네트 번호가 할당된 마스크 사각형으로부터 전기적으로 연결 가능한마스크 사각형을 찾아 동일한 네트 번호를 할당
(3) 과정 (2)를 가능한한 계속 반복수행
(4) 네트가 지정되지 않은 마스크 사각형이 존재하면 과정 (1)로 분기

네트 추출 실험에서는 위의 알고리듬에 근거하여 네트 선택, 파급, 합병, 분할 등의 네트 추출 문제해결의 모형이 6개의 규칙으로 표현되었으며(OPS5를 이용할 경우 14개의 규칙이 필요), 50개의 연결가능성 오브젝트와 네트 번호를 저장하기 위한 1개의 오브젝트가 사용되었다.

또한 본 논문에서의 실험을 위하여 구성한 몇몇 샘플 마스크 레이아웃에 대한 네트 추출 실험 결과와 소요시간은 아래의 표 2, 3과 같다.

표 2. 네트 추출에 대한 실험 결과
Table 2. Experimental Results for Net Extraction.

| 마스크 레이아웃 | 마스크 사각형 수 | 추출된 네트 수 ① | ② | ③ | 비교 ①:②:③ |
|---|---|---|---|---|---|
| 1 | 54 | 6 | 6 | 6 | 1:1:1 |
| 2 | 94 | 11 | 11 | 11 | 1:1:1 |
| 3 | 40 | 6 | 6 | 6 | 1:1:1 |
| 4 | 16 | 3 | 3 | 3 | 1:1:1 |
| 5 | 45 | 8 | 8 | 8 | 1:1:1 |

① : 구현된 시스템을 이용하여 구축한 전문가 시스템
② : OPS5를 이용하여 구축한 전문가 시스템
③ : MAGIC 시스템 [15]

표 3. 네트 추출에 소요된 시간 비교
Table 3. Elapsed Time for Net Extraction.

| 마스크 레이아웃 | 마스크 사각형 수 | 추출된 네트수 | 소요시간(단위:초) ① | ② | 비교 (①/②)×100 |
|---|---|---|---|---|---|
| 1 | 54 | 6 | 72 | 330 | 21.8 |
| 2 | 94 | 11 | 141 | 870 | 16.2 |
| 3 | 40 | 6 | 32 | 300 | 10.6 |
| 4 | 16 | 3 | 31 | 58 | 53.4 |
| 5 | 45 | 8 | 52 | 275 | 18.9 |

① : 구현된 시스템을 이용하여 구축한 전문가 시스템
② : OPS5를 이용하여 구축한 전문가 시스템

2) 마스크 레이아웃 컴팩션

마스크 레이아웃 컴팩션 실험에서는 기존의 shear-line 컴팩션 기법[16,17]을 근거로 하는 문제해결의 모형이 38개의 규칙으로 정의되었으며(OPS5를 이용할 경우 185개의 규칙이 필요), MOSIS SCMOS 1.5 ㎛설계규칙에 근거하여 309개의 오브젝트가 사용되었다. 표 4는 구축된 전문가 시스템을 이용하여 설계한 여러가지 샘플 마스크 레이아웃에 대한 실험 결과를 나타낸다.

표 4. 마스크 레이아웃 컴팩션에 대한 실험 결과
Table 4. Experimental Results for Mask Layout Compaction.

| 마스크 레이아웃 | 마스크 사각형갯수 | 면적(㎛²) ①컴팩션 전 | ②컴팩션 후 | 비교 (②/①)×100 |
|---|---|---|---|---|
| 1 | 54 | 220.0×259.5 | 38.0×46.5 | 3.1 |
| 2 | 94 | 223.5×327.0 | 58.5×106.5 | 8.5 |
| 3 | 40 | 100.5×137.0 | 31.0×33.0 | 7.4 |
| 4 | 16 | 214.0×90.0 | 27.0×33.0 | 4.6 |
| 5 | 45 | 129.0×139.5 | 44.0×36.0 | 8.8 |

또한 구현된 시스템을 이용하여 구축한 마스크 레이아웃 컴팩션 전문가 시스템을 기존의 benchmark[18] 중 하나인 afa 셀에 대하여 실험한 결과는 아래의 표 5와 같다.

표 5에 나타난 바와 같이, 기존의 간단한 컴팩션 지식만을 사용하여 구축한 마스크 레이아웃 컴팩션 전문가 시스템을 이용하여 어느 정도 기존 시스템에 근접하는 결과를 얻을 수 있었다. 그러나 마스크 레이아웃 패턴을 변환하는 규칙과 기존의 마스크 레이아웃 컴팩션 알고리듬을 변환 규칙으로 정의함으로써 마스크 레이아웃 변환 지식을 확충할 경우 컴팩션 전후의 면적 감소율을 더욱 증대시킬 수 있을 것이다.

표 5. 마스크 레이아웃 컴팩션에 대한 Benchmark 실험결과
Table 5. Benchmark Results for Mask Layout Compaction.

| 시스템 | 면적(㎛²) | 비교(%) |
|---|---|---|
| MACS | 143×166=23738 | 100.0 |
| Zorro | 14.5×171=24025.5 | 101.2 |
| 구축된 시스템 | 161×173=27853 | 117.3 |
| SPARCS | 157×180=28260 | 119.0 |
| Symbolics | 160×189=30240 | 127.4 |

3) 마스크 레이아웃 편집

마스크 레이아웃 편집 실험에서는 기존의 레이아웃 설계 시스템인 MAGIC 시스템[15]의 주요 편집 기능인 paint, erase, delete, stretch, move 등의 주요 기능들이 21개의 규칙으로 표현되었다(OPS5를 이용할 경우 56개의 규칙이 필요). 또한 제안된 모델에 대한 실험과는 별도로, 이들 기능과 그래픽 인터페이스를 이용하여 앞서 언급한 샘플 마스크 레이아웃 1~5와 아래의 샘플 레이아웃 A~C를 실제로 설계하여 이용하였으며, 설계 과정에서 네트 추출 지식을 활용하여 설계와 네트 추출을 병행시킬 수 있었다.

4) 설계규칙 검사

설계규칙 검사의 실험에서는 마스크 사각형의 최소폭 검사, 마스크 사각형 사이의 최소이격거리 검사, 트랜지스터 확장거리 검사, 마스크 사각형의 특이 패턴 처리 등이 23개의 규칙으로 정의되었다. 설계규칙 검사 실험을 위하여 본 논문에서 별도로 구성한 샘플 레이아웃 A, B, C에 대한 실험 결과는 표 6과 같다.

표 6. 설계규칙 검사의 실험 결과
Table 6. Experimental Results for Design Rule Checking.

| 마스크 레이아웃 | 마스크 사각형갯수 | 실제의 설계규칙 오류 수 | 면적(㎛²) ① | ② |
|---|---|---|---|---|
| A | 64 | 13 | 13 (0) | 14 (+1) |
| B | 104 | 18 | 19 (+1) | 20 (+2) |
| C | 34 | 5 | 15(0) | 18 (+3) |

① : 구현된 시스템을 이용하여 구축한 전문가 시스템
② : MAGIC 시스템

표 6에서 보듯이 구축된 설계규칙 검사 전문가 시스템과 MAGIC 시스템에서는 실제 존재하는 설계규칙 오류의 수보다 많은 수의 오류가 검출되었다. 일반적으로 설계규칙 검사에서는 주어진 마스크 레이아웃에 존재하는 오류를 지나치는 것보다 모든 잠재적인 오류의 가능성을 검출하는 것이 주목적이므로 구축된 설계규칙 전문가 시스템과 MAGIC 시스템은 설계규칙 검사에 효과적으로 사용될 수 있을 것이다.

그러나 MAGIC 시스템에서 실제 존재하는 설계규칙 오류보다 많은 수의 오류가 검출된 주된 요인은 MAGIC 시스템이 설계규칙 정보만을 바탕으로 고정된 방법에 의거하여 설계규칙 오류를 검출하기 때문이다. 즉, MAGIC 시스템에서 실제 존재하는 오류 이외에 부가적으로 검출된 오류는 모두가 네트 정보를 이용하지 않기 때문에 발생된 것이다. 또한 구축된 전문가 시스템에서는 다수의 마스크 사각형으로 구성된 복잡한 패턴을 변환하는 규칙이 정의되지 않은 관계로 부가적인 설계규칙 오류가 검출되었다.

## Ⅳ. 결론 및 앞으로의 연구방향

본 논문에서는 기능이 알고리듬 형태로 고정됨으로 인해 시스템의 확장성, 새로운 문제로의 적응성, 사용자 지원 등의 관점에서 비효율적임이 지적되고 있는 기존의 마스크 레이아웃 처리 시스템의 문제점을 해결하기 위하여, 마스크 레이아웃의 일부분에 대한 변환이라는 통합된 문제해결의 모형 하에서 마스크 레이아웃 단계에서의 대부분의 문제를 효율적으로 해결할 수 있는 규칙에 기초한 마스크 레이아웃 변환 시스템을 제시하였다. 그리고 본 논문에서 제안한 시스템은 SUN 워크스테이션의 UNIX 환경 하에서 SUN View 그래픽 라이브러리를 이용하여 40,000 줄의 C 프로그래밍 언어로 구현되었으며, 구현된 시스템은 그래픽 윈도우와 마우스를 이용한 대화형 방식의 사용자 인터페이스를 제공한다.

또한 본 논문에서는 구현된 시스템을 네트 추출, 마스크 레이아웃 컴팩션, 마스크 레이아웃 편집, 설계규칙 검사 등의 여러가지 문제에 적용하였다. 실험결과, 본 논문에서 구현한 시스템은 범용으로 사용되는 기존의 규칙에 기초한 시스템에 비해서 주어진 마스크 레이아웃 문제를 해결하는 규칙 정의가 간결하고 마스크 레이아웃 변환 규칙의 처리 과정이 효율적이므로 마스크 레이아웃 문제를 해결하는데 효과적으로 사용될 수 있음이 판명되었다. 또한 여러가지 마스크 레이아웃 문제해결에 대한 변환 지식이 하나의 통합된 환경 하에서 저장 및 관리되므로 정의된 다양한 변환 지식 모듈을 연계하여 원하는 마스크 레이아웃 설계 작업을 효율적으로 수행할 수 있었다. 현재 구현된 시스템을 이용하여 본 논문에서 실험한 것 이외의 여러가지 마스크 레이아웃 문제를 해결하고자 하는 연구가 진행 중이다.

본 논문에서 구현한 규칙에 기초한 시스템은 기존의 마스크 레이아웃 처리 시스템에서 많이 사용되는 휴리스틱을 이용할 수 있도록 함으로써 문제해결에 소요되는 공간 및 시간 복잡도의 문제점을 완화시키고자 하였다. 그러나 구현된 규칙에 기초한 시스템을 네트 추출, 마스크 레이아웃 컴팩션, 마스크 레이아웃 편집, 설계규칙 검사 등의 여러가지 마스크 레이아웃 문제에 대하여 실험한 결과, 이들 기능이 알고리듬으로 구현된 기존의 마스크 레이아웃 처리 시스템에서 보다 문제해결에 많은 시간이 소요됨이 도출되었다. 일반적으로 규칙에 기초한 시스템에서는 문제해결(특히 규칙매치 단계)에 과도한 시간이 소요된다는 문제점이 지적되어 왔으며, 본 논문에서 제안한 마스크 레이아웃 변환을 위한 문제해결 환경도 규칙에 기초한 시스템에 바탕을 두고 있으므로 이러한 환경이 갖는 근본적인 문제점을 완전히 극복하지는 못하였음을 알 수 있었다. 이러한 문제점을 부분적으로 극복하기 위하여 기존의 규칙에 기초한 시스템의 매치 알고리듬을 혼합한 새로운 규칙매치 알고리듬을 고안함으로써 규칙매치 단계의 효율을 증대시키고자 하는 연구가 현재 진행 중이다.

향후 레이아웃 지식을 좀더 상위 단계로 추상화하여 표현할 수 있도록 함으로써 비단 마스크 레이아웃 뿐만 아니라 회로 설계의 여러 단계의 문제를 해결할 수 있는 전문가 시스템 개발 환경을 제공할 수 있을 것이다. 또한 문제해결에 많은 시간이 소요되는 문제점을 근본적으로 극복하기 위하여 추론 전용의 컴퓨터나 하드웨어 가속기에 대한 연구가 수반되어야 할 것으로 사료된다.

## 參 考 文 獻

[1] T. J. Kowalski and D. E. Thomas, "The VLSI Design Automation Assistant: An IBM System/370 Design", *IEEE Design and Test of Computers*, pp. 60 69, Feb. 1984.

[2] J. H. Kim, J. McDermott, and D. P. Siewiorek, "Exploiting Domain Knowledge in IC Cell Layout", *IEEE Design and Test of Computers, pp. 52* 64, Aug.

1984.

[3] E. Simoudis and S. Fickas, "The Application of Knowledge-Based Design Techniques to Circuit Design", *Proc. International Conference on CAD*, pp. 213-215, Jun. 1985.

[4] R. Brück and K. Herrmann, "KOCOS - An Expert System for VLSI Layout Compaction", International Conference on Computer Design, pp. 298-301, 1986.

[5] R. Joobbani and D. P. Siewiorek, "WEAVER: A Knowledge-Based Routing Expert", *IEEE Design and Test of Computers*, pp. 12-23, Feb. 1986.

[6] L. Brownston, R. Farrell, E. Kant, and N. Martin, Programming Expert Systems in OPS5, Addison Wesley, 1986.

[7] C. L. Forgy, OPS83 Report, Technical Report CMU-CS-84-133, Dept. of Computer Science, Carnegie-Mellon University, May 1984.

[8] J. A. Darringer, W. H. Joyner, C. L. Berman, and L. Trevillyan, "Logic Synthesis Through Local Transformations", IBM J. Research and Development, vol. 25, no. 4, pp. 272-280, 1981.

[9] R. Hojati, "Layout Optimization by Pattern Modification", *Proc.* 27th Design Automation Conference, pp. 632-637, Jun. 1990.

[10] 이재황, "규칙에 기초한 집적회로 레이아웃 변환 시스템에 관한 연구", 공학박사학위논문, 서울대학교, 1993년 2월.

[11] M. P. Georgeff, "Procedural Control in Production Systems", Artificial Intelligence, vol. 18, no. 2, pp. 175-201, Apr. 1982.

[12] F. Preparata and J. Nievergelt, "Plane-sweep Algorithms for Intersecting Geometric Figures", Communications of the ACM, vol. 25, no. 10, pp. 739-747, 1982.

[13] J. K. Ousterhout, "Corner Stitching: A Data Structuring Technique for VLSI Layout Tools", *IEEE Transactions on CAD*, vol. CAD-3, no. 1, pp. 87-99, Jan. 1984.

[14] C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence, vol. 19, no. 1, pp. 17-37, Sep. 1982.

[15] 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*, vol. 2, no. 1, pp. 19-30, 1985.

[16] S. B. Akers, J. M. Geyer, and D. L. Roberts, "IC Mask Layout with a Single Conducting Layer", *Proc.* 7th Design Automation Workshop, pp. 7-16, Jun. 1970.

[17] A. E. Dunlop, "SLIP: Symbolic Layout of Integrated Circuits with Compaction", *Computer-Aided Design*, pp. 387-391, Nov. 1978.

[18] D. G. Boyer, Symbolic Layout Compaction Benchmarks Session, Proc. International Conference on Computer Design, pp. 185-217, 1987.

## 著 者 紹 介

**李 在 晃(正會員)**

1963年 2月 13日生. 1985年 2月 서울대학교 컴퓨터공학과 졸업. 1987年 2月 서울대학교 컴퓨터공학과 석사학위 취득. 1988年~1990年 서울대학교 반도체공동연구소 조교. 1990年 서울대학교 컴퓨터공학과 시간강사. 1993年 2月 서울대학교 컴퓨터공학과 박사학위 취득. 1993年 3月 서울대학교 컴퓨터신기술공동연구소 특별연구원. 주관심분야는 디지탈 시스템 설계, 설계자동화, VLSI/CAD 등임.

**全 洲 植(正會員)**

1952年 3月 18日生. 1975年 2月 서울대학교 응용수학과 학사. 1977年 2月 한국과학기술원 석사학위 취득. 1983年 2月 Univ. of Utah 박사학위 취득. 1983年~1985年 Univ. of Iowa 조교수. 1985年~ 현재 서울대학교 컴퓨터공학과 부교수. 주관심분야는 컴퓨터 구조, VLSI/CAD, 병렬처리 시스템 등임.