Quad-tree구조를 이용한 계층적 회로 추출 알고리즘

林載允*, 金興洙*

A Hierarchical Circuit Extract Algorithm Using Quad-tree structure

*Lim Jae-yun,** *Kim Heung-soo**

Summary

A hierachical circuit extract algorithm, which efficiently extract circuits from VLSI mask pattern information, is programmed. Quad-tree is used as a datastructure which includes various CIF circuit elements and instances. This system is composed of CIF input routine, quad-tree making routine, transistor finding routine and connection list making routine.

This circuit extractor can extract circuit with hierarchical structure of circuit.

서 론

회로의 기술로 부터 실제의 마스크 패턴을 생성하는 각종 CAD프로그램이 개발되어 회로설계에 이용되고 있으며, 이렇게 설계된 회로는 기능적으로 정상동작하는지의 여부를 검사할 필요성이 대두되었다. 이러한 목적으로 개발된 도구가 회로 추출기로서 이는 마스크 도면으로부터 원회로를 추출하여 그 기능을 검사하는 방법으로서, 종래에는 각회로요소를 linked-list형태로 보관함으로서 회로탐색에 과다한 시간이 소모되었고, 모든 회로요소를 flatten시킴으로서 많은 기억용량이 필요하였다 (Anoop ,1983).

이러한 단점을 해결하기 위해 최근 회로의 기하학적 크기에 따라 일정 영역별로 회로요소를 분리 저장함으로서 회로 탐색시간을 대폭적으로 줄일 수 있는 quad-tree가 고안되어 VLSI설계에 널리 사용되고 있다(Kedem, 1982).

이에따라 본 논문에서는 VLSI회로의 마스크 데이타의 한 형태인 CIF로 부터 각 회로요소를 quad-tree형태로 보관한 후, 회로의 계층구조를 그대로 유지한채 원 회로도를 추출하는 알고리즘

* 공과대학 전임강사

을 제안한다. 우선 CIF형태의 데이타를 quad-tree형태로 보관한 후, 계층구조의 각 블럭별로 트랜지스터의 위치 및 종류별 상태를 발견하고 트랜지스터의 상태를 기초로 각 등전위점을 추출하여 신호선 리스트를 생성한 후, 원 논리회로를 추출하여 논리검증을 수행한다. 이렇게 추출된 결과는 그래픽 터미날 상에서 schematic 형태로 확인할 수 있다.

연구 방법

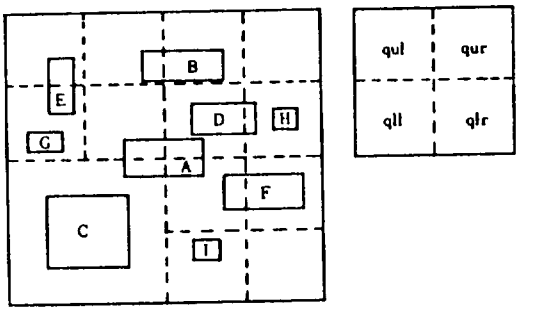
1. Quad-tree구조에 의한 회로요소 저장

본 회로추출기의 데이타 구조는 계층적 구조의 데이타를 효율적으로 처리하고, 회로의 각종 정보를 정확하고 규칙적으로 구성하기 용이하며 요소의 삽입, 삭제, 확인등에 따른 탐색 등이 효율적인 quad-tree구조를 채택하였다. 이는 요소의 크기에 따라 해당 영역에 데이타를 저장하는 형태로 초기의 사각영역을 설정하고 이를 4등분하여 적정 위치에 요소를 위치시키는 형태로서 Fig. 1은 요소 A, B, C, …등에 대한 quad-tree 구조를 나타낸 것이다.

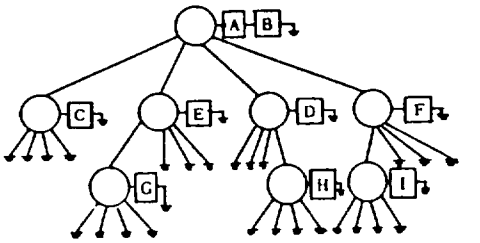
일반적으로 회로 구성요소는 CIF형태로 나타내며, 크게 box, 다각형, 블럭 등으로 구성되며 그 형태는 다음과 같다.

B width, height, x point , y point;
P xp1, yp1 xp2, yp2, …xpn, ypn;
C blocknum T xpoint, ypoint;

여기서 box 의 크기는 x pont-width/2, y point-height/2, x point+width/2, y point+width/2의 값으로 계산되어 보관되며, 다각형은 box 형태로 분할된 후 보관시킨다. 또 타 블럭의 복사형태를 C blocknum T C_x, C_y의 형태로 표시하며 blocknum은 복사할 블럭의 이름이며



a) Object 예



b) a)에 대한 Quad-tree 구조

Figure 1. Quad-tree data structure.

C_x, C_y는 현 블럭에 복사시킬 위치로서 블럭정의시 기준점이 된다. 만일 복사할 블럭의 원 기준점을 P_x, P_y라 하고, 복사할 블럭내의 임의의 한 점을 I_x, I_y라할 때 이 점이 현 블럭 내에서의 상대적 위치 N_x, N_y는 다음과 같이 계산된다.

$$N_x = I_x + C_x - P_x \quad \cdots\cdots(1)$$
$$N_y = I_y + C_y - P_y \quad \cdots\cdots(2)$$

이러한 요소들은 우선 해당 블럭내의 크기에 따른 초기 면적을 설정한 후, 이를 이 블럭의 초기 quad크기로 선정하고 각 회로의 요소들을 일정 영역에 recursive하게 위치시킨다.

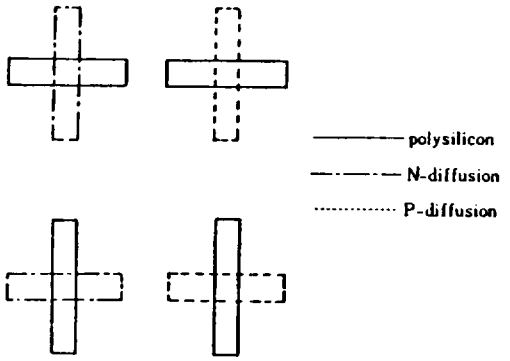
2. 계층적 회로 추출 알고리즘

본 시스템은 회로의 마스크 도면으로부터 계층적 구조를 그대로 유지한 채 논리도를 추출하는 시스템으로서 크게 트랜지스터 인식과정 및 등전

위 신호선리스트 생성과정으로 나눈다.

1) 트랜지스터 인식 알고리즘

트랜지스터는 polysilicon과 diffusion이 교차하는 곳에 생성되며, 이는 다시 P형 및 N형 트랜지스터로 구별되고 방향에 따라 다음(Fig. 2)과 같이 4가지 종류로 나눌수 있다.



a) N형 트랜지스터 b) P형 트랜지스터

Figure 2. N type and P type transistor.

이들이 각각 설계규칙을 만족하고, 이들 사이에 contact이 존재하지 않으면 트랜지스터로 인식하여, 발견된 트랜지스터는 종류 및 방향과 함께 binary-tree를 사용하여 X 및 Y값 순으로 저장한다. 계층적 트랜지스터 검출은 quad내, quad들간에서 box들 간의 트랜지스터 검출, box와 블럭들간 및 복사할 블럭들 간의 트랜지스터 검출등으로 이루어진다. 이렇게 발견된 트랜지스터는 등전위점을 추출하기 위해 diffusion 영역을 두 부분으로 나누어서 Fig. 3과 같이 보관한다.

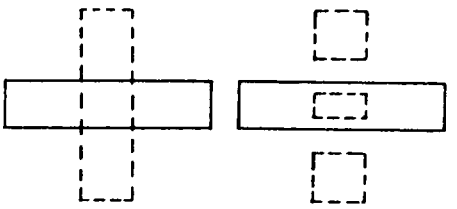


Figure 3. Partition of selected transistor.

우선 box 간의 트랜지스터 검출은 두 box가 만나는 중심점을 트랜지스터의 위치로 하여 블럭내에 보관시키며 다음과 같이 수행된다.

```
BB-TR-check(Q)
QUAD *Q;
{ if( ! Q)
    return;
  for(Q내의 서로다른 box 쌍에 대해)
    만일 두 box가 트랜지스터를 형성하면
      트랜지스터의 종류, 방향별 저장;
  for(Q 내의 한 box인 B에 대해)
    INTRA-BB-TR-check(Q, B);

  BB-TR-check(Q->ql1);
  BB-TR-check(Q->qul);
  BB-TR-check(Q->qur);
  BB-TR-check(Q->qlr);
}
```

또, box와 블럭간 트랜지스터 검출 과정은 하나의 box와 복사된 block들 간의 트랜지스터를 검출하는 과정으로서 한 블럭내에서 recursive하게 수행되며, 이 과정을 나타낸 예가 Fig. 4이다.

```
DS  1  2 / 1;
    L  CP;
    B  6  2  3.3 ;
    L  CND;
    B  2  14 3.7 ;
DF;
DS  2  2 / 1;
    L  CP;
    B  14 2  11.11;
    C  1  T  &0 ;
DF;
```

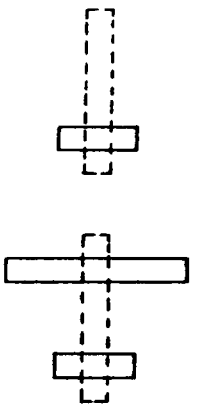


Figure 4. Transistor find between box and block

이에 대한 알고리즘을 기술하면 다음과 같다.

```
BC-TR-check(Q)
QUAD *Q;
```

```
{ if( ! Q)
    return;
  for(현 QUAD의 box인 B와 복사할 블럭인
  QB에 대해)
  {QB의 환산된 새 기준점 QX, QY 계산;
    INTRA-BC-TR-check(B, QB->Q,
    QX, QY);
  }
  BC-TR-check(Q->qll);
  BC-TR-check(Q->qul);
  BC-TR-check(Q->qur);
  BC-TR-check(Q->qlr);
}
```

한편, 블럭간 트랜지스터 검출은 현 블럭내의 복사된 두 블럭 사이에 교차 또는 포함 관계에 있을 경우 이들 블럭 간의 트랜지스터를 발견하는 과정으로서 다음은 하위 블럭으로 정의된 두 블럭사이에 형성된 트랜지스터의 예(Fig. 5)를 보인 것이다.

```
DS  10  2 / 1  20,2 ;
L   CP;
    B  40  4  20,2 ;
DF;
DS  20  2 / 1  20,10;
L   CND;
    B  4  20  8,10;
L   CPD;
    B  4  20  32,10;
DF;
DS  30  2 / 1  20,10;
    C  10  T  20,10;
    C  20  T  20,10;
DF;
E
```

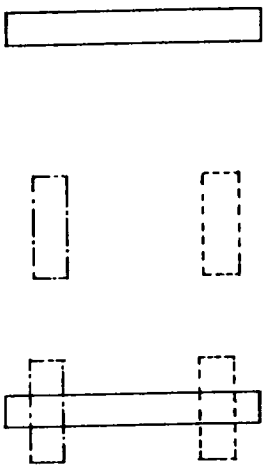


Figure 5. Transistors between bocks.

이런 블럭간 트랜지스터 검출 알고리즘을 나타내면 다음과 같다.

```
CC-TR-check(Q)
QUAD *Q;
{ if( ! Q)
    return;
  for(Q 내의 두 블럭 쌍에 대해)
  { 두 블럭의 상대 위치 계산;
      BOX-BOX, BOX-BLOCK 간 트랜지
      스터 발견 과정 반복;
  }
  for(Q 내의 모든 블럭 QA 에 대해)
    하부 QUAD 와 QA 와의 트랜지스터 발견
    과정 수행;

  CC-TR-check(Q->qll);
  CC-TR-check(Q->qul);
  CC-TR-check(Q->qur);
  CC-TR-check(Q->qlr);
}
```

2) 회로 연결도 검증

마스크 도면으로부터 트랜지스터의 위치 및 종류를 탐색한 후 트랜지스터를 중심으로 등 전위점을 찾아 원 회로도를 추출해 내는 과정으로, 신호선 리스트를 생성하여 원 회로와 비교, 회로를 검증하는 과정이다. 이중 하위 블럭 신호선 분할 및 병합과정을 나타내면 다음과 같다.

```
Block-Wire-insert(Block)
BLOCK *Block;
{
  for( 현 블럭 내의 모든 복사된 블럭들에 대
       해)
  { 만일 현 블럭내의 회로 요소에 대해 트랜
    지스터를 생성하면 --> 복사 블럭의 해당
    신호선을 분리하여 현 블럭의 신호선
    리스트로 병합하고;
  만일 단지 복사 기능만을 가지면
      --> 현 블럭의 신호선 리스트로 병합;
  }
}
```

또한 신호선은 크게 등전위점을 연결하는 방법과 서로 다른 전위점일 경우 분리하여 저장하는 방법이 있다. 신호선중 동일층 끼리 교차하거나 다른 층이지만 contact이나 via 등으로 연결되면 동일 신호선으로 간주하여 동일 그룹으로 둔다. 또한 동일 그룹중에서 신호선의 연결관계에 의해 두 신호선을 한 신호선으로 병합하거나, 다른 방향으로 신호선을 구별하여 저장한다. Fig. 6은 병합 또는 구별저장의 예를 보인 것이다.

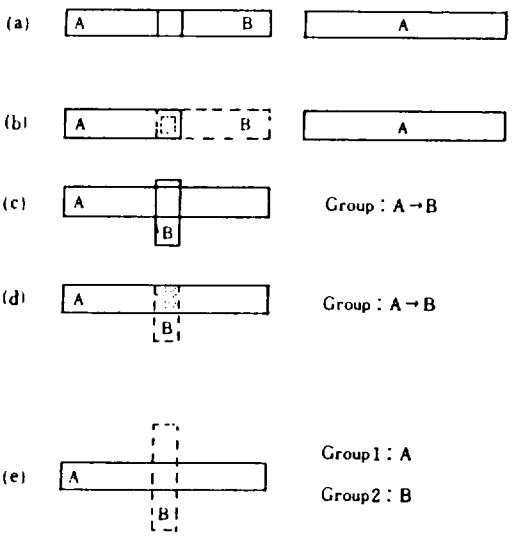


Figure 6. Merging and partition of signals.

한편 두 신호선이 서로 다른 층으로 교차하고 두 층사이에 contact, via가 존재하지 않으면 기존의 신호선과의 교차 여부를 검사하여, 동일 층으로서 교차관계에 있는 신호선 그룹을 찾으면 그 그룹내에 저장하고, 그렇지 못하면 새로운 그룹을 설정하여 그곳에 저장한다. 또 신호선 연결과정에서 한 신호선에 의해 다른 두 그룹이 병합가능하면, 두 그룹을 하나의 그룹으로 병합하면서, 그룹내에 병합 가능한 신호선은 병합시킨다. Fig. 7은 동일 신호선이 서로 다른 그룹에 속해 있을 때 새 신호선에 의해 두 그룹을 병합하는 과정을 보인 것이다.

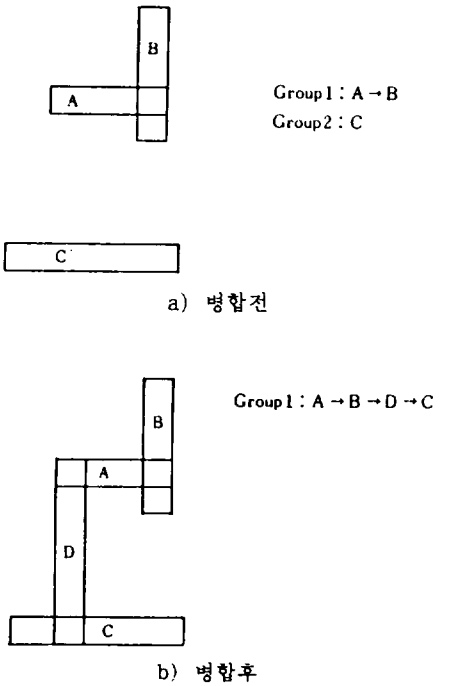


Figure 7. Group merging of signals.

이상의 회로 연결도 검증 알고리즘을 나타내면 다음과 같다.

```
Wire-search(Q)
QUAD *Q;
{ if( ! Q)
    return;
  for(Q 내의 서로 다른 두 box들에 대해)
  { 두 box 가 교차할 경우
    { 두신호선 선택여부 check;
      if( 두신호선이 동일층이면)
          동일 신호선으로 간주, 신호선리스
          트로 삽입;else
      { if(두 신호선 사이에 contact나 via가
        존재하면)
          동일 신호선으로 간주, 신호선 병합;
          else
          다른 신호선으로 간주 독립적으로
          신호선 삽입;
      }
```

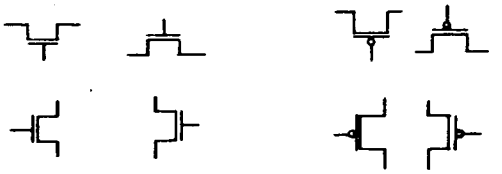
```
        신호선 병합가능시 병합;
      }
    }

    for(Q의 box와 하부 quad에 대해)
      recursive하게 동일한 방법으로 신호선 검
      사, 병합;

    Wire-search(Q->qll);
    Wire-search(Q->qul);
    Wire-search(Q->qur);
    Wire-search(Q->qlr);
  }
```

3. 회로 추출기의 출력형태

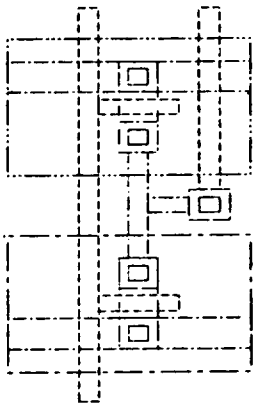
본 회로 추출기의 출력형태는 트랜지스터의 위치 및 종류별 리스트, 신호선에 대한 그룹별 리스트 형태로 나타낼 수 있고 그래픽 터미날 상에서 schematic형태로 표시할 수도있다. schematic 형태로 표시할 경우 트랜지스터는 P형 및 N형으로 나누고 그 위치에 따라 Fig. 8과 같이 각각 4종류로 출력된다.



a) N형 트랜지스터 b) P형 트랜지스터

Figure 8. Schematic output of transistor.

한편 회로 추출 결과를 신호선 리스트 형태로 출력시킬 수 있으며, 트랜지스터의 경우 트랜지스터의 종류 및 이와 연결된 신호선들의 상태로서 표시된다. 이에대한 예로서 그림 9-a는 inverter회로에 대한 마스크 도면을 나타낸 것이고 이에 대한 도면 정보로서 CIF형태를 사용하였으며 이를 Fig. 9-b에 나타내었다. 이를 입력으로 회로 추출 알고리즘을 수행한 후 그 결과를 신호선 리스트로 표시한 것이 Fig. 9-c이며, 이에 대한 schematic을 나타내면 Fig. 9-d와 같이 된다.



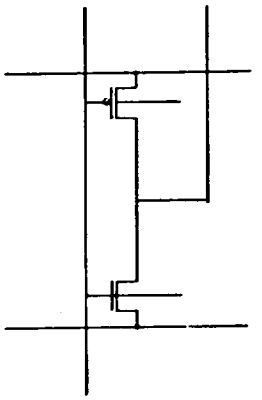
a) inverter 예

```
DS    5032   2  /   1   26   86;
L     CND;       B   8   24   26,  26;
L     CPD;       B   8   24   26,  78;
L     CP;        B  16    4   26,  26;
                 B  16    4   26,  78;
                 B   4  104   16,  52;
                 B   4   48   40,  80;
L     CM;        B  48    8   24,  18;
                 B  48    8   24,  86;
                 B   4   28   26,  52;
                 B   8    4   32,  52;
L     CNW;       B  48   36   24,  78;
L     CPW;       B  48   36   24,  26;
L     CC;        B   8    8   26,  34;
                 B   8    8   26,  18;
                 B   8    8   26,  86;
                 B   8    8   36,  70;
DF;
```

b) a)에 대한 CIF 도면 정보

```
PMOS   26   26   VDD   W1   W2;
NMOS   26   78   W2    W1   VSS;
W1     16    0   16   104;
W1     16   26   34    26;
W1     16   78   34    78;
W2     26   38   26    66;
W2     26   52   40    52;
W2     40   52   40   104;
VSS     0   18   48    18;
VDD     0   86   48    86;
```

c) 추출된 netlist



d) schematic 출력결과

Figure 9. Net list and schematic output from mask layout.

4. 회로 추출기의 구성도

본 시스템은 VAX 11/780 UNIX 4.3 BSD 상에서 자동구문 생성, 분석기인 yacc 및 LEX를 사용하여 구문을 구성하였으며, 유용한 각종 함수들은 C언어를 사용하여 프로그램 하였으며 전체 프로그램 길이는 약 8,000라인 정도이다. 본 회로 추출기의 입력형태로는 CIF 및 Auto CAD상의 DXF형태의 데이타를 이용할 수 있으며 DXF 형태는 자동적으로 CIF형태로 변환된 후 입력으로 사용된다. 내부 데이타 베이스로는 Quad-tree 구조를 갖게 하였으며 회로 추출기의 출력형태는 원 마스크 패턴에 대한 데이타는 CIF 또는 DXF형태로, 신호선은 CIF의 wire형태로 출력시키며 이를 그래픽 터미날에서 schematic형태로 나타낼 수 있게 하였다. 이를 간략히 표시하면 Fig. 10과 같다.

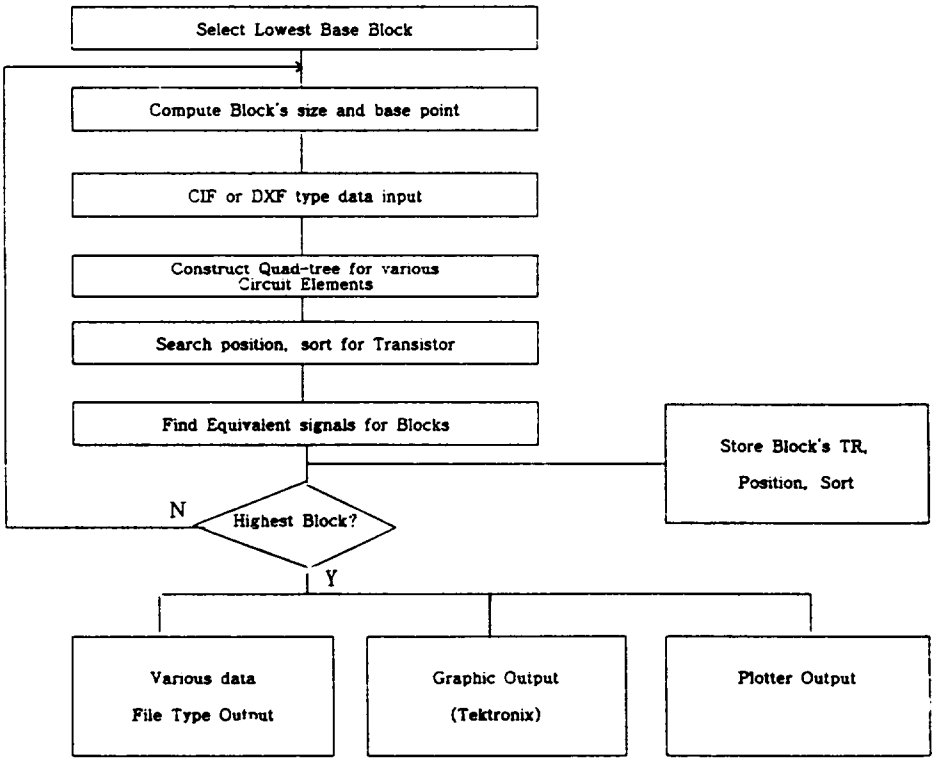


Figure 10. Configuration of circuit extractor.

결과 및 고찰

VLSI회로의 설계검증을 위한 계층적 회로 추출 알고리즘을 제안하였다. 원 회로가 갖는 계층적 구조를 그대로 이용함으로서 종래의 flatten화 시킨 방법에 따른 회로 기억용량 및 트랜지스터 탐색 시간 및 등전위 신호선 탐색시간을 감소시키며 보다 정확한 설계검증을 수행할 수 있게 하였다. 또한 각종 회로 요소들에 대한 데이타 베이스로서 quad-tree구조를 갖게 함으로서 탐색시간의 단축 및 불필요한 요소탐색을 배제할 수 있고, 회로요소의 변경에 따른 데이타 베이스 구축이 용이하게 구성하였다. 회로추출 결과는 그래픽 터미날 상에서 그래프로 나타낼 수 있게 하였고, 그 결과를 신호선 리스트 형태의 file로 나타낼 수 있게 하여 차후에 시뮬레이션 및 상위층의 회로설계에 대한 입력으로 줄 수 있게 하였다.

적 요

본 논문에서는 VLSI의 마스크 도면정보로부터 회로도를 효율적으로 추출하기 위한 계층적 회로 추출 알고리즘을 제안하였다. 마스크 레이아웃 데이타는 CIF형태의 요소로 구성하며 이들 각 요소를 quad-tree형태로 보관함으로서, 회로 탐색시간 및 기억공간을 감소시켰다. 본 알고리즘은 CIF입력부 quad-tree생성부, 트랜지스터 검증부 및 연결 리스트 생성부 등으로 구별되며 출력을 그래픽 형태로 표시할 수 있게 하였다.

본 회로 추출기는 회로의 계층적 구조를 그대로 유지하면서 트랜지스터의 위치 및 연결도를 중복없이 정확히 검증할 수 있으며, 이로 부터 회로도를 추출할 수 있다.

참 고 문 헌

Anoop G. 1983. ACE : A Circuit Extractor, *Proc. 20th DA Conf.*, pp. 721-725.

Bootehsz, A. and R. A. Cottrell, 1986. A Technology Independent to Hierarchical IC Layout Extraction. *Proc., 23th DA Conf.* pp. 425-431.

Chew, M. and A. J. Strojwas. 1987. Efficient Circuit Reextraction For Yield Simulation Applications. *ICCAD 87.* pp. 310-313.

Kedem, G. 1982. The Quad-CIF Tree : A Data Structure for Hierarchical On-line Algorithhms, *Proc, 19th DA Conf.*, pp. 352-357.

McCormik, S. P. 1984. EXCL : A Circuit Extractor for IC Design, *Proc. 21th DA Conf.* pp. 616-623.

Mukjerjee, A. 1986. Introduction to NMOS & CMOS VLSI Systems Design, *Prentice-Hall.*

YACC, LEX, UNIX Programmer's Supplementary Documents vol. 1 pls : 15-1~ pls : 16-13.