論 文

객체 지향형 프로그래밍을 이용한 CAM 생성기 구현에 관한 연구

正會員 白 寅 天* 正會員 朴 魯 京** 正會員 車 均 鉉***

A Study on the Implementation of CAM Generator Using Objected-Oriented Programming

In Cheon PAIK*, Nho Kyung PARK*,** Kyun Hyon TCHAH*** *Regular Members*

要 約 본 논문에서는 객체 지향 프로그래밍을 이용한CAM(Content Addressable Memory)의 자동생성의 run-plot 과정을 위한 생성기, 그래픽 디스플레이 툴(tool)을 제시하였다. 레이아웃 생성이나 그래픽 메뉴, 마우스 드라이버, 데이타 구조들을 기본 클래스들로 구성하여 기존의 절차적 언어보다 명료하고도 수정이 용이한 프로그램을 구성하였음을 보였다. CAM의 구조와 기본 회로 구성을 보였고 이들을 생성하기위한 기본 클래스에 대해 설명하였다. 설계 규칙이나 공정에 독립적인 생성기 설계를 위해 기본 셀들을 사용자의 입력에 따라 변하도록 구성할 수 있는 매개변수화된 셀의 기법을 보였고 피치 매칭 기법을 이용한 배치·배선으로 레이아웃을 수행하였다. 끝으로 생성된 CIF의 디스플레이와 전체 run plot 과정을 위한 그래픽 메뉴의 구성을 설명하였다.

ABSTRACT In this thesis, CAM(Content Addressable Memory) generator and graphic display tool for run-plot sequence in automatic generation of CAM are presented. We show that implementing the layout generation, graphic menu, mouse driver, and data structure by using the basic classes is clear and easy in modification than the conventional procedural language. For the implementation of generator which is independent of design rule or process, we use the parameterized cell so that basic cell can be changed according to user's inputs, and perform the layout by means of placement and routing using pitch matching. Finally, the display of CIF which generated and constitution of graphic menu for total run-plot sequence are explained.

I. 서 론

반도체 공학의 지속적인 발달로 집적 회로의 집적도가 매우 커지고, 다양한 연산을 위한 프로세서들의 IC 칩화에 많은 발전을 가져왔다. 이들 프로세서들은 메모리, ALU(Aithmatic Logic Unit), PLA(Programmable Logic Array), 승산기와 같은 모듈들을 가지며, 모듈들을 서로 연결하기 위한 랜덤 로직을 갖는다. 각 모듈의 모든 가능한 레이아웃을 제공하는 것은 설계 시간이 많이 소요되고, 이와 같은 많은 경우에 대해 라이브러리화하게 되면 메모리 오버 플로우가 발생한다.

이 문제를 해결하는 하나의 방법으로 매개 변수화된 모듈을 사용하는 것이다[1,2]. 이 매개 변수화된 모듈은 레이아웃 생성기 또는 셀 컴파일러라고 불리는 소프트웨어에 의해서 구현할 수

*高麗大學校 電子工學科 大學院
Dept. of Electronic Eng., Korea University
**湖西大學校 情報通信工學科
Dept. of Information Telecommunication Eng., Hoseo Univ.
***高麗大學校 電子電算工學科
Dept. of Electronic Eng., Korea Univ.
論文番號 : 91-124 (接受1991. 8. 5)

있다[3]. 이러한 소프트웨어에 의한 집적회로 설계는 시간이 적게 들고 설계 검증 및 수정이 용이하다는 장점을 갖고 있어 최근에 널리 이용되고 있다.

본 연구에서는 캐쉬 메모리내의 고속 탐색 테이블이나 인식을 위해 입력 데이타와 메모리 내용을 비교하여 가장 근사한 데이타를 찾는데 사용되는 CMOS CAM(Content Addressable Memory)의 자동 생성기를 객체 지향형 프로그래밍(Object Oriented Programming) 기법을 사용하여 구현하였다. 객체 지향형 프로그래밍 기법은 개발된 프로그램의 보수, 유지 및 갱신이 용이하고 개발이 용이한 기법이다[7].

본 자동 생성기는 사용자의 입력에 따라 설계 결과를 CIF로 출력한다. CAM 레이아웃은 다른 메모리 모듈처럼 규칙적이고 반복적인 구조를 가졌기 때문에 레이아웃 제너레이터로 프로그램화 하기가 용이하다. 또한 생성된 결과의 검증 및 제너레이터를 개발할 때 필요한 run-plot sequence를 독자적인 충족을 위해 CIF 데이타를 그래픽 화면에 출력할 수 있도록 프로그램을 개발하였다.

Ⅱ. CMOS CAM의 구조와 동작

본 연구에서 구현한 CAM은 메모리 셀 어레이, 어드레스 인코더, 어드레스 디코더, 멀티플렉서, 감지증폭기의 5개의 기능별 블럭으로 구성되어 있다. CAM 생성기의 출력은 사용자의 입력 파라미터에 따라서 그림 1과 같은 구조의 레이아웃을 생성하게 된다. 그림 1은 생성할 CAM의 블록도이다. 구성된 CAM의 기능구조는 보는 메모리 위치에 대해 탐색하려는 데이타와 비교, 일치 여부를 고속으로 알리는 매치 기능과 SRAM과 같은 판독과 정보 저장 동작을 수행하도록 구성되었다. 즉 CAM중에서 어드레스 디코더, 메모리 셀 어레이, 비트 선, 워드 선, 감지 증폭기는 SRAM과 같은 방법으로 판독과 정보 저장 동작을 수행한다. 어드레스 버스와 데이타 버스가 내부적 어드레스와 데이타 전송에 요구된다.

CAM의 매치 동작은 다음과 같이 수행된다.

1) 탐색할 데이타가 칩의 외부핀으로 부터 데이타 버스를 통하여 멀티플렉서로 입력된다.

2) 제어신호인 MUXS가 매치, 판독, 정보 저장에 따라서 멀티플렉서를 제어하여 비트 선과 그 보수 선을, 판독이나 정보 저장을 하는 경우는

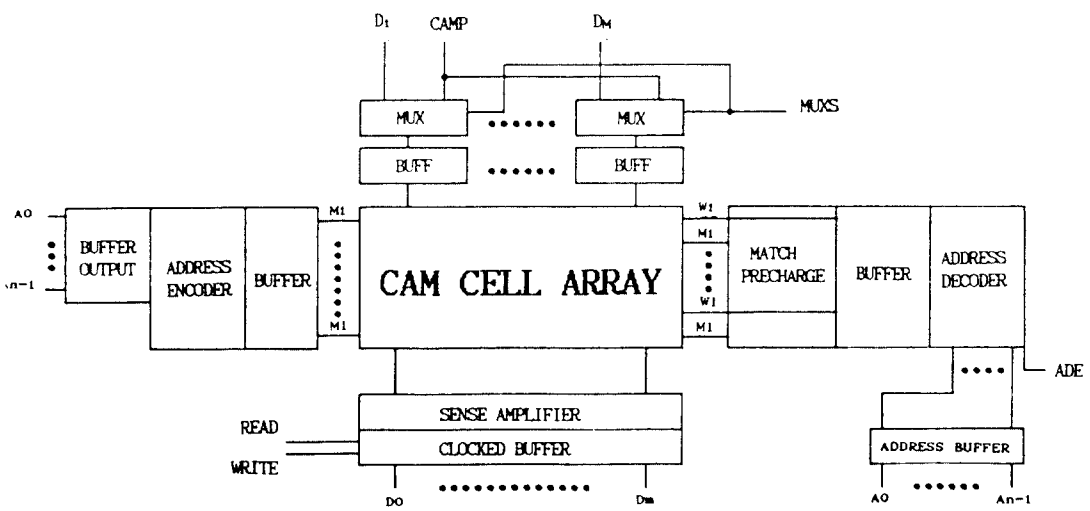


그림 1. 구성할 CAM의 구조

precharge에 연결시키고 매치 기능을 하는 경우에는 탐색할 데이타에 연결시킨다.

3) 메모리 셀에 저장된 내용중 탐색하려는 워드와 매치되는 워드가 있는가를 알기 위해서 매치 신호를 "1"로 하고 만약 비트 선에 입력된 탐색할 데이타가 메모리 워드에 저장되어 있는 데이타와 일치하게 되면 매치가 발생되어 그 워드에 있는 매치 감지선의 값이 "1"이 되고 어드레스 인코더로 입력되어 그 워드에 대한 어드레스를 출력하게 된다[5]. 각 블록을 형성하는 셀들의 지연 시간과 메모리 용량에 따르는 구동 버퍼의 최적 geometry 값을 SPICE 시뮬레이션을 통해 구하였고 구현된 CAM 생성기에서는 가상적 비트와 워드를 구동시킬 수 있는 버퍼 크기를 기본으로 정하였다. 프로그램 사용자가 원하는 억세스 시간에 따라 버퍼 크기를 바꾸고자 할 때 참고 자료로서 기본 값으로 정한 버퍼 크기에 대하여 비트수와 워드수가 바뀔 때 팬아웃(fanout)을 고려한 지연시간을 결정하였다.

Ⅲ. CAM 생성기의 구성

3.1 기본 객체의 구성

본 연구에서는 CAM 생성에서 기본적이면서 빈번히 사용될 변수와 프로시듀어에 대한 객체(Object)를 C++ 언어를 사용하여 구성하였다. 사용자의 입력에 관한 설계 공정 규칙, 공정상의 레이어, CIF 출력에 대한 것을 기본 클래스(base class)로 설정하고 사각형이나 다각형을 그리는 클래스(PenHolder class), 여러 종류의 콘택, 트랜지스터 들을 그리는 클래스(Macro-Pen class)들을 기본 클래스로 부터 상속시켜 제작하였다.

설계 공정 규칙 클래스는 제작 규칙(construction rule), 간격 규칙(spacing rule), 확장 규칙(extension rule)에 대한 변수, 차후 레이아웃의 스케일링(scaling)에 대처하기 위해 필요한 lambda 값을 위한 변수들을 protected 영역에 포함하고 이들의 설정, 규칙 제공에 대한 메쏘드(method)들을 public 영역에 위치시켰다.

레이어에 대한 클래스는 각 레이어를 인식할 수 있는 변수가 있고 현재 레이어의 설정, 현재 레이어의 제공에 대한 메쏘드들을 제작하였다.

CIF 출력에 대한 클래스는 셀의 번호, 비율, 변환 정보들을 저장하는 변수와 셀의 시작, 끝, 호출을 행하는 메쏘드들로 구성된다. 이들로 부터 PenHolder 객체는 자신의 도형 정보를 받고 내부의 기본 설계 규칙을 검증하며 현재 레이어에 따라 도형 정보를 출력한다. 이것은 플롯터에서 각 레이어에 해당되는 펜을 잡아 도형을 그리는 펜 홀더를 객체로 구현한 개념이다. 그림 2에 PenHolder 클래스의 예를 보였다. MacroPen 클래스는 PenHolder와 Cell 클래스로 부터 메쏘드들을 상속받아 여러 크기 및 종류의 콘택, 트랜지스터들을 그릴 수 있다. MacroPen을 이용하여 제작한 기본 셀은 수평·수직방향의 NMOS, PMOS, 일반 콘택, VDD와 VSS를 위한 콘택, POLY, NPLUS, PPLUS 콘택 및 이들의 수평·수직 배열 형태의 기본 셀및 블럭들을 한번에 그릴 수 있다. 예를 들면 VDD 콘택은 설계 규칙에서 최소 콘택선, 금속선 폭을 받아 NPLUS 확산, 금속, 콘택을 겹쳐 그린 것이다[6].

3.2 매개 변수화된 셀

매개 변수화된 셀을 이용하여 사용자 입력 파라미터로부터 생성기의 계층구조를 통해여 파라미터가 전파되도록 한다. 본 연구에서 사용된 매개 변수화된 셀은 다음의 두가지로 나눌 수 있다.

첫째는 셀의 배열이다. 셀의 배열의 매개 변수는 사용자 입력 파라미터 중에서 워드 수나 워드당 비트의 수가 된다. 셀의 배열에 있어서 주변의 다른 셀과의 인접지역에서는 접촉, 즉 블럭과 블럭이 연결되는 경우 입·출력의 피치가 맞아야 하기 때문에 내부적으로 같은 셀들끼리 연결된 것과는 셀의 모양이 틀려진다. 그러므로 셀 레이아웃을 할 경우 이점을 유의하여야 한다. CAM 생성기에서는 같은 셀들끼리의 내부적 연결은 상하·좌우 배치시 입력과 출력 그리고 파워선의 연결이 직접적으로 행해지도록 설계하였고, 메

```
class PenHolder : public Layer, public Rule {
POINT CurrentP ;
LayerType CurrentL ;
public :
PenHolder() { Layer::LayerSet() ;
              Rule ::RuleSet() ;
            }
void DrawBox(LayerType layer, POINT low, POINT high, int dirx, int diry);
void DrawRBox(LayerType layer, POINT RelPoi) ;
 // 현재 위치로 부터 상대적인 위치의 BOX //
void DrawPolyGon(LayerType layer, POINTS polygon) ;

.....
} // end of PenHolder Object //
```

그림 2(a). PenHolder 클래스의 예

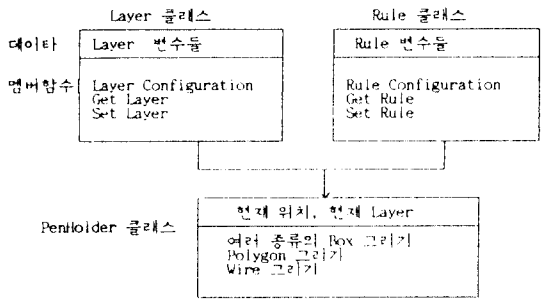


그림 2(b). PenHolder 클래스의 분석도

그림 2. PenHolder 클래스의 예와 분석도

모리 블럭과 다른 주변 블럭들과의 인터페이스는 주변 블럭에서 모두 처리하도록 하였다. 셀의 배열은 레이아웃이 2차원 형태의 정보를 가지기 때문에 셀의 bounding box를 이용하여 x축이나 y축으로 배열을 할 수 있다.

둘째는 매개 변수를 가진 셀인데 셀 자체의 매개 변수는 셀의 레이아웃 구조중 어떤 것일 수도 있다고, 예를 들어 셀들의 피치를 맞추기 위해 셀의 bounding box를 확장하거나 버퍼의 구동 능력을 조절하기 위해 W/L을 변화시키는 것등이다. 본 연구에서는 특히 억세스 시간에 가장 큰 영향을 미치는 버퍼 크기를 사용자의 입력 파라미터에 따라 모두 변화시킬 수 있게 하므로써 CMOS 제조 공정에서 SPICE 파라미터의 변화나 워드 수나 워드당 비트 수에 따른 부하 커패시턴스의 변화에 쉽게 대처할 수 있게 구성하였다. 또한 이것은 CAM이 어떤 시스템의 모듈로 들어갈 때 원하는 매치 시간, 판독 시간, 정보 저장 시간에 대한 요구와 칩의 크기에 대한 요구사이에서 CAM 생성기로 실현할 수 있는 레이아웃 중 가장 최적의 해를 만들어내기에 용이하다. 그림 3에 버퍼 크기를 가변시키기 위한 예를 나타내었다. 트랜지스터를 생성하는 메쏘드에서 폭을 바꿀 수 있는 매개 변수를 사용해 그림 2에서 hi와 lo 사이의 트랜지스터 폭 tr_w를 변화시키고, 콘택 배열을 정의한 함수를 사용해 tr_w에 맞게 콘택을 한다. 이에 따라 점선으로 표시된 bounding box의 높이 bb_w도 자동적으로 변하게 되어 셀 폭의 크기가 전체 셀 전체의 스케일링 효과를 가져오게 하였다. 또, 트랜지스터 폭의 설계 규칙에 따른 변화가 bounding box의 변화에 포함되도록 프로그래밍하였다. 그리므로 버퍼 크기의 가변성

으로 인한 셀의 생성이나 배열에 있어서의 문제는 전혀 발생하지 않는다.

3.3 피치 매칭(Pitch Matching)

배선이 전체 레이아웃의 면적에 크게 영향을 미칠 때와 레이아웃의 생성 속도의 증가, 완전 주문형 방식으로 설계한 경우 워드의 수나 워드당 비트의 수가 늘어날 경우 피치가 가장 큰 서브셀을 가진 블럭이 상하나 좌우로 길게 늘어나게되어 칩이 구형인 모양을 가질 수 없게 되므로 본 연구에서는 각 블럭의 최상위의 서브셀들끼리 상하 또는 좌우 셀들의 피치를 맞추도록 다시 레이아웃을 하였다[5]. 레이아웃한 셀의 피치는 가로의 피치를 맞추기 위하여 감지 증폭기 서브셀의 피치가 가장 컸으므로 메모리와 멀티플렉서 서브셀의 피치를 감지 증폭기에 맞추었고 세로의 피치를 맞추기 위하여 디코더의 서브셀의 피치가 가장 크므로 메모리와 인코더의 서브셀 피치를 디코더에 맞추었다.

3.4 레이아웃 프로그램

3.4.1 기본 셀(Leaf Cell)

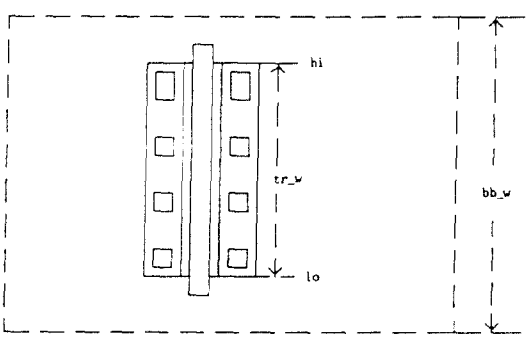


그림 3. 매개 변수화된 셀

(1)메모리

CAM의 메모리 셀은 트랜지스터 10개로 구성되어있다. 그림 4에 회로도와 레이아웃의 한 예를 보였다. 메모리 셀의 상측 트랜지스터 6개의 SRAM의 구조와 같고 동작도 같다. 그리고 아래쪽의 트랜지스터 4개로 구성된 EX-NOR에 의해 매치 동작을 수행한다. 메모리 블럭 구성은 주변 블럭에서 블럭간 인터페이스를 처리하기 때문에 그림 4(b)의 메모리 셀 레이아웃의 배열만으로 이루어 진다. 워드 선과 감지 선은 POLY로 수평 방향으로 연결되고 VDD선과 VSS선 그리고 비트 선은 메탈로 수직 방향으로 연결되어 메모리 블럭의 내부 배열에서 셀들이 상하·좌우 직접적으로 배치 가능하도록 레이아웃한다.

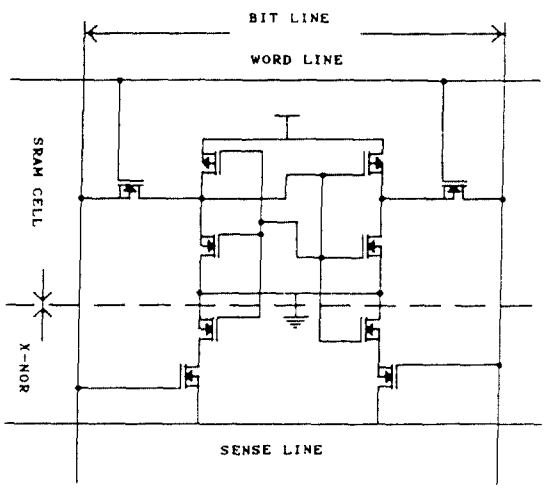


그림 4(a). 회로도

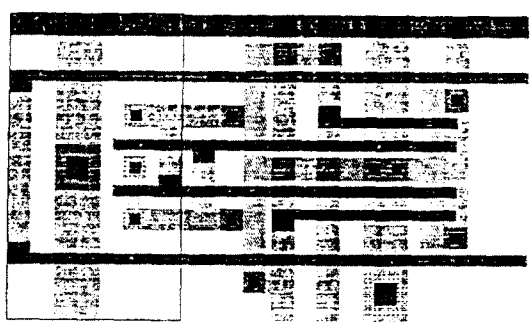


그림 4(b). 메모리 셀 레이아웃

그림 4. 메모리 셀 회로도 및 레이아웃

(2)멀티플렉서와 입력 버퍼

멀티 플렉서에 들어가는 제어 신호 MUXS에 의해 판독이나 정보 저장 동작시에는 precharge하고 매치 동작시에는 탐색할 데이타가 들어오게 된다. 사용자에 의하여 트랜지스터 폭이 변화될 수 있는 셀은 MUXS, $\overline{MUXS}$ 신호를 만들어 낼 제어 신호 버퍼와 데이타 입력 버퍼이다. 그림 5는 멀티 플렉서와 입력 버퍼의 회로도와 감지 증폭기의 서브셀과 피치 매칭을 한 레이아웃이다.

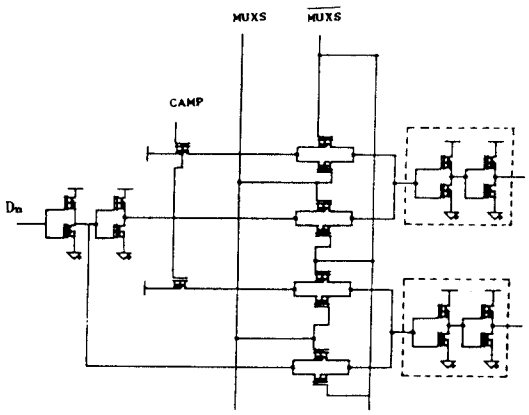


그림 5(a). 멀티 플렉서와 입력버퍼 회로도

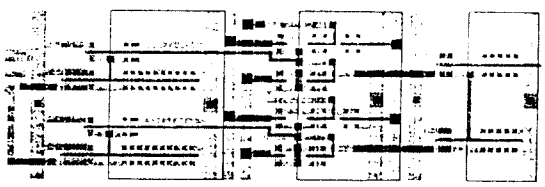


그림 5(b). 멀티 플렉서와 입력 버퍼 레이아웃

그림 5. 멀티 플렉서와 입력 버퍼 회로도와 레이아웃

(3)감지 증폭기와 clocked 버퍼

그림 6은 감지 증폭기와 clocked 버퍼의 회로도이다. 판독은 READ에 의해, 정보 저장은 WRITE에 의해 동작하고 두 제어 신호는 non-overlap 되어야 한다. 사용자에 의해 트랜지스터 폭이 변화될 수 있는 셀은 READ, $\overline{READ}$와 WRITE, $\overline{WRITE}$ 신호를 만들어낼 제어 신호 버퍼와 clocked 버퍼이다. 그림 6에서 보여진 셀은 메모리, 멀티 플렉서, 감지 증폭기 블럭의 최상위의 서브셀에 해당한다. 그러므로 이 블럭들의 구성은 위의 서브셀들의 배열에 의해서 이루어진다. 그러나 어드레스 인코더와 어드레스 디코더의 경우는 어드레스 코딩을 위한 것이 입력되므로 단순히 서브셀을 배열하는 것만으로는 블럭이 구성되지 않는다.

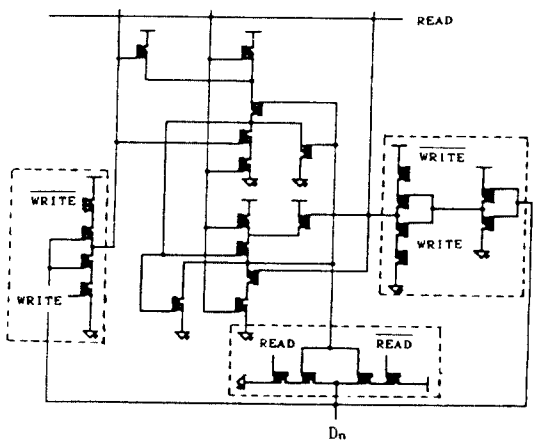


그림 6. 감지 증폭기와 clocked 버퍼 회로도

(4)어드레스 디코더

그림 7은 4워드를 생성할 때 디코더의 회로도와 레이아웃이다. 좌우의 서브셀 피치중 가장 큰 디코더의 피치를 줄이기 위하여 폴딩(folding)을 이용해 VDD선과 WELL을 공유하도록 디코더의 레이아웃을 설계하였다. 즉 디코더의 서브셀을 하나 만들고 이를 X축으로 대칭 이동시키고 Y축을 따라 위로 이동시켜 위아래의 서브셀과 합쳐서 디코더 블럭의 최상위 서브셀로 정의한다. 여기에서 디코더의 서브셀에는 디코딩하는 부분이 제외된다. 위와 같이 2개의 워드에 해당하는 서브셀을 최상위의 서브셀로 정의할 수 있는 것은 워드의 수가 2의 제곱 형태로 늘어나기 때문이다.

모든 어드레스의 각각의 비트에 대한 "1"과 "0"은 POLY선으로 디코더의 서브셀 배열시에 미리 생성되어 있고 프로그램에 의해 확산과 콘

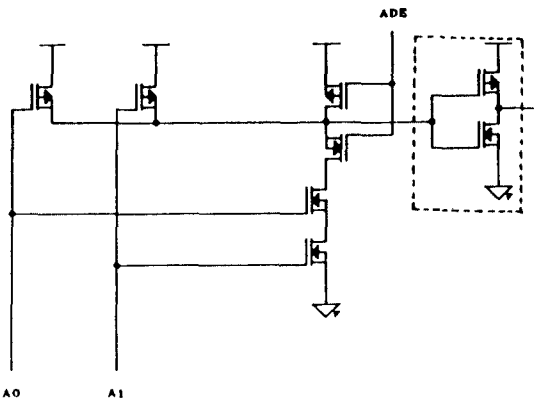


그림 7. 어드레스 디코더 회로도

택을 원하는 어드레스 생성을 위해 정해진 곳에 위치시키므로써 디코딩을 한다. 디코더의 최상위 서브셀 생성에도 대칭 이동을 이용했으므로 어드레스를 생성할 때도 대칭 이동을 이용해야 한다. 구현한 CAM 생성기의 디코더 어드레스는 MSB가 가장 왼쪽에 있고 LSB는 가장 오른쪽에 위치한다. 디코딩 부분의 바로 오른쪽의 POLY 선은 ADE 제어 신호로서 CAM이 판독이나 정보저장을 할 때만 디코더가 동작되어 워드선을 구동하도록 한다. 그림 1의 CAM 블럭도에서 매치 precharge는 원래 디코더 블럭에 포함되지는 않지만 여기서는 레이아웃 구조의 편의상 포함시켰다. 매치 precharge는 매치 동작시에 각 워드의 매치선을 precharge하는 역할을 하게된다.

사용자에 의해 트랜지스터 폭이 변화될 수 있는 셀은 매치 precharge 셀과 모든 어드레스 각각의 비트에 대한 "1"과 "0"을 구동할 어드레스 버퍼 그리고 그림 7에서 점선으로 표시한 워드선 버퍼이다.

(5)어드레스 인코더

인코더의 경우도 디코더와 거의 비슷하다. 그림 8은 4워드를 생성할 때 인코더의 회로도이다. 인코더 블럭의 서브셀은 매치선의 버퍼이다. 나머지 인코더의 precharge와 어드레스의 출력 버퍼는 각각의 기본셀을 정의하고 디코더 블럭의 위와 왼쪽에 따라 배열하여 구성된다. 매치되는 어드레스를 출력하기 위한 인코딩 부분은 상위 2워드의 레이아웃 부분이다. 여기서는 트랜지스터가 있는 경우와 없는 경우에 대한 셀을 미리 정의해 두었다가 어드레스 형성에 따라 정해진 곳으로 위치시키게 된다. 인코딩 부분의 어드레스 중 MSB는 가장 오른쪽에 있는 것이고 LSB는 가장 왼쪽에 있는 것이다. 사용자에 의하여 트랜지스터 폭이 변화될 수 있는 셀은 그림 8에서 점선으로 표시한 precharge 셀과 매치 신호 버퍼이다.

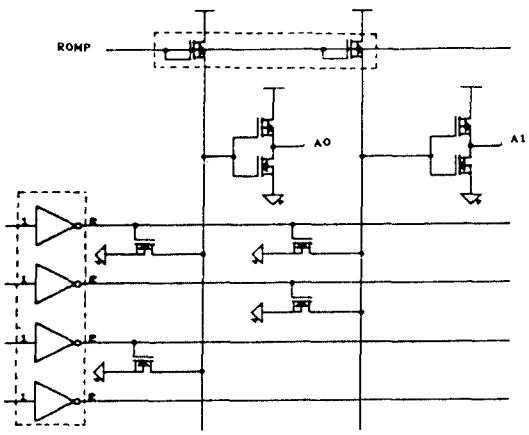


그림 8. 어드레스 인코더 회로도

3.4.2 레이아웃 알고리즘

CAM 생성기의 입력으로는 설계 규칙과 사용자 입력 파라미터이다. 생성기가 수행되면 설계 규칙을 관장하는 객체는 화일로 부터 데이타를 읽어 자신의 변수값을 정한다. 후에 사용자 입력 파라미터를 읽어 변수를 정한 후 이것을 이용하여 워드선 수, 비트선 수, 어드레스 수, 각 블럭에 들어가는 버퍼 크기등의 CAM을 구성하는데 필요한 정보를 계산한다. 앞에서 구한 정보를 이용하여 메모리, 멀티 플랙서, 감지 증폭기, 어드레스 인코더, 어드레스 디코더의 순으로 각 셀을 생성한다.

Ⅳ. CAM 레이아웃의 디스플레이

앞 절의 CAM 생성기의 확인을 위해 전용 디스플레이 프로그램을 개발하였다. 이것은 생성기의 출력을 곧바로 확인하고 다시 생성기를 디버깅할 수 있는 run-plot sequence를 다른 레이아웃용 소프트웨어가 없더라도 가능하도록 한다. 본 연구에서는 C++와 같은 객체 지향형 언어를 사용함으로 그래픽 화면상의 메뉴, 마우스 기능등을 간단 명료하게 구현하였다[8].

디스플레이 프로그램은 크게 CIF 데이타를 처리하는 기능과 그래픽 화면을 처리하는 기능으로 나눌 수 있다.

CIF 데이타를 처리하기 위해서는 CIF의 인식기, 정보 저장을 위한 데이타 구조가 필요하다. CIF를 인식하기 위해 LALR(LookAhead Left Right) 파서 생성기인 UNIX의 YACC를 사용하여 CIF 문법을 용이하게 작성하고 이를 DOS로 옮겨 컴파일하여 파서를 제작하였다[10]. 파서가 CIF 문법을 인식하면서 레이아웃 정보를 이진 트리(binary tree)에 저장한다. 따라서 이진 트리구조와 이의 운용을 위한 메쏘드들을 갖는 클래스를 제작하였고 이를 그림 9에 보였다[8]. CIF에서 하나의 셀(DS에서 DF까지)은 하나의 이진 트리에 저장된다. 따라서 각각의 셀에 대하여 하나의 트리 객체의 instance를 만들게 된다. 여기에 저장될 수 있는 데이타 형태는 그림 9의 DataItem type의 레이어, 명령어 종류(Box, Call, Polygon)들이다. 트리 클래스에서는 셀 정의의 시작시 instance가 생기고 CIF의 도형 정보에 따라 각 노드에 도형의 위치를 좌하단에서 우상단의 순으로 저장하게 된다(MakeCIFTree 메쏘드). 각 셀에 대한 트리의 루트노드는 트리 정보의 출력시 빈번히 사용되므로 빠른 탐색을 위해 각 트리의 루트노드의 포인터를 해쉬 테이블에 저장하였다. 제작한 해쉬 테이블 클래스는 사용자가 핸스트럭터 함수로 테이블의 크기를 정할 수 있는 것으로, 저장정보는 셀 번호, 스케일, 트리의 루트포인터이고 동일한 해쉬 값에 대해서는 연결 리스트 구조를 저장하게 하였다.

CIF화일의 모든 데이타가 각각의 트리 객체에 저장되면 CIF 데이타의 플롯을 위해서는 사용자가 원하는 셀 번호를 받아 해쉬 데이블에서 트리 포인터를 찾고 그 셀 트리에서 전체를 inorder 방식으로 운행하며 정보를 출력하게 하였다. 이때 트리 내의 노드 정보가 다른 셀을 호출하는 것이라면 다시 그셀의 트리 포인터를 해쉬 테이블에서 찾아 그 트리를 운행하고 돌아오는 반복적인(recursive)형태로 진행되게 된다. 여기서, 현재 셀을 처리하는 중에 다른 셀을 처리하고 다시 돌아올 때, 이전의 변환상태를 유지해야 하므로 이를 위해 스택(stack) 클래스를 구성하였다. 실제 트리를 운행할 때 CIF의 계층구조가 primitive CIF 형태로 바뀌므로(flattening) 셀을 호출하는 경우나 도형을 그리는 경우 현재 자기의 변환 정보를 사용하게 되므로 본 구현에서는 변환 매트릭스[11]를 사용하였다. 다른 셀이 호출되어 그 트리로 옮겨가고자 할때 스택에 입력되는 정보는 CallType 형태로서 이것은 Transport, Mirror, Rotate 정보를 갖고 있어 이로부터 변환 매트릭스를 구성한 후 이 매트릭스로 부터 도형을 변환한다. 호출이 끝나고 원래의 트리 노드로 복귀하면 자기자신의 변환정보를 스택으로 부터 CallType으로 돌려받아(pop 동작) 자신의 변환 매트릭스를 구성한다. 따라서 이 스택은 사용자에 의해 깊이가 정해지고 push, pop의 동작을 한다[11].

트리, 해쉬 테이블, 스택 형태를 갖는 객체의 간단한 구조를 그림 9에 보였다

그래픽 화면을 처리하기 위해서 기본적인 객체로는 마우스 객체와 메뉴 버튼 객체를 들수가 있고 화면 처리를 위해 전체 프레임, 위의 기본 메뉴 버튼을 조합하여 각각의 1차(primary) 메뉴 버튼 객체와 각각의 1차 메뉴버튼에 서브 메뉴에 해당하는 객체들을 각각의 메뉴 버튼들을 조합하여 구성하였다. 전체 구조는 그래픽 모드상에서 좀 다른 메뉴 형태로 하였고 각 셀들에 대한 디스플레이, 마우스에 의한 zoom 기능을 첨가하였다. 서브 메뉴 객체의 예를 그림 10에 보였다.

```
union ComdType {
     BoxType boxdata ;
     CallType calling ;
     Points pts ;
} ;
```

```
struct  DataItem {
    LayerType layer ;
    UType     utype ;
    ComdType  command ;
} ;
```

```
struct TreeNode {
     TreeNode *Root ;
     TreeNode *Left ;
     TreeNode *Right ;
     DataItem infor ;
} ;
```

```
 struct Htlist { int cellnum,a,b; Tree *Treeptr ; Htlist *next; } ;

 class Htable { int hashsize ;       // Start of Hash Table Object //
                Htlist **hashtab ;
 public :
 Htable(int sz) { hashsize = sz ;  hashtab = new Htlist *[hashsize] ;
 int hash(int s) ;
 Htlist *lookup(int s) ;
 Htlist *install(int num, Tree *ptr);
 Tree *lookcellptr(s) ;
} ;  // End of Hash Table Object //

class Stack {   // Start of Stack Object //
int size; CallType *top; CallType init[1] ;
public :
stack(int sz);
void push(CallType tp) ;
CallType pop() ;
} ;  // End of Stack Object //


class Tree {  // Start of Tree Object //
private :
 TreeNode *NRoot ;
public :
 Tree() ;
 -Tree() ;
 void Add(DataItem *NewItem) ;
 int MakeCIFTree() ;
 void PrintTree() ;
 ..............................
} ; // end of class Tree //
```

그림 9(a). 해쉬 테이블, 스택, 트리를 위한 자료구조

그림 9. 해쉬 테이블, 스택, 트리를 위한 자료구조 및 클래스 예

```
class SubMenuButton {
  MenuButton Open, Close, Quit ;
public :
  SubMenuButton() { Open.Intialize() ; ····· }
  void Draw() ;
  void Erase() ;
  int OpenHit() ;
} ;
```

그림 10. 서브 메뉴 객체의 예

따라서 이 메뉴 안에서 CAM의 생성, 디스플레이의 전 과정이 이루어 지도록하였다.

V. 결 론

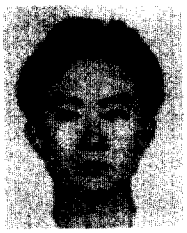
본 논문에서 사용자의 입력 파라미터에 따라 CMOS CAM의 레이아웃 정보를 생성하는 CAM자동 생성기 및 이를 디스플레이 할 수 있는 프로그램을 객체 지향형 언어를 사용하여 구현하였다. 사용자의 입력 파라미터는 워드 수와 비트 수 그리고 CAM을 구성하는데 필요한 버퍼들의 크기가 있다. 특히 모든 버퍼 크기의 변수화로 칩의 가장 큰 특징중 하나인 억세스 시간 조절이 매우 용이하게 하였다. 매개 변수화된 셀이 입력을 파라미터화하기 위하여 사용되었고 피치 매칭으로 각 블럭들을 연결하였다.

또한 생성된 레이아웃의 용이한 수정을 위해 레이아웃 디스플레이 프로그램을 객체 지향 프로그래밍을 통해 구현하였다. 따라서 이에 대한 보완이 용이하여, 보완 및 확장을 위한 개발 시간의 단축 및 효율적인 기능 구현의 효과가 예상된다. 추후 객체지향 프로그래밍을 통한 레이아웃 전용 언어의 구현, 이를 위한 전용 그래픽 에디터의 구현 연구가 기대된다.

참 고 문 헌

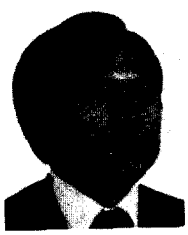
1. Chu, K.C. and R.Sharma, "A Technology Independent MOS Multiplier Generator", Proc. of 21st Design Automation Conf., pp.90-97, 1984
2. W.P.Swartz etc., "CMOS RAM, ROM and PLA Generators for ASIC Applications", Proceedings of 1986 CICC, pp.334-338, 1986
3. Stephen M.Trimberger, "An Introduction to CAD for VLSI", Chap.6-11, Kluwer Academic Publisher, 1987
4. Bamji C.S., C.E.Hauck, and J.Allen, "A Design by example regular structure generator", in Proc. of 22nd Design Automation Conf., pp.16-22, 1985
5. Bryan T.Preas, and Michael J.Loremetti, "Physical Design Automation of VLSI Systems", Chap. 7, The Benjamin/Cummings Publishing Co., 1988
6. 서정한 외, "Content Addressable Memroy Generator", International Conference on VLSI and CAD, ICVC'89, Seoul, pp.348-351, Oct., 1989
7. Gerald E.Peterson, "TUTORIAL : Object Oriented Computing", The Computer Society of IEEE, Chap 1., 1987
8. Ben Ezzell, "Turbo C++ Programming", Addison Wesley Publishing Company, Inc., chap 9-11, 1990
9. 서정한 외, "데이타 변화에 의한 PC 상에서의 IC 레이아웃 프로그램에 개발에 관한 연구", 대한 전자 공학회 논문지, 제25권 제9호, 1988
10. Mead, Conway, "INTRODUCTION TO VLSI SYSTEMS", Addison Wesley, pp.125, 1980
11. B.Stroustrup, "The C++ Programming Language", Addison Wesley, pp.165, 1986

※본 연구는 1990년 학술진흥재단 학술연구비 지원에 의하여 연구 되었음



白 寅 天(In Cheon PAIK) 正會員
1963年 1月 14日生
1985年：高麗大 電子科 卒業
1987年：高麗大 大學院 電子科(碩士)
1987年～現在：高麗大 大學院 電子科 博士過程

朴 魯 京(Nho Kyung PARK) 正會員
1958年 1月 8日
1984年 2月：高麗大學校 電子工學科 卒業
1986年 2月：高麗大學校 工學碩士學位 取得
1990年 2月：高麗大學校 工學博士學位 取得
1988年－現在：湖西大學校 情報通信工學科 助教授
※主關心分野는 VLSI/CAD, 通信 回路 및 시스템 自動設計 등



車 均 鉉(Kyun Hyon TCHAH) 正會員
1939年 3月 26日生
1965年：서울大學校 工學士
1967年：美國일리노이大學校 工學碩士學位 取得
1976年：서울大學校 工學博士學位 取得
1977年～現在：高麗大學校 電子電算工學科 教授