論文 85-22-2-7

Bit-map 방식에 의한 설계규칙 검사

(A Design Rule Checker Based on Bit-Mapping)

魚 吉 秀*, 金 炅 台*, 慶 宗 旻*

(Kil Su Eo, Kyeong Tae Kim and Chong Min Kyung)

要 約

NMOS IC layout에서 직사각형 도형의 갯수에 비례하는 검사시간을 소모하는 설계규칙 검사의 알고리즘의 제안되고 그것에 의한 program이 개발 되었다. 일반적인 설계규칙 검사 algorithm의 시간소모는 0(nlogn) 혹은 0(n**1.2)에 비례하는데 반하여, (n은 직사각형 도형의 갯수) 이 논문에서는 pattern의 DF (direct format) data와 bit-map plane을 연관 지음으로써 0(n)에 비례하는 시간소모를 달성 할 수 있었다.

Abstract

This paper describes a DRC (Design Rule Check) algorithm and its program implementation which requires CPU time linearly proportional to the number of rectangular patterns in the NMOS IC layout. While the CPU time for conventional DRC algorithm is proportional to O(nlogn) or O(n**1.2), (n:number of rectangles) it was shown that the present algorithm only consumes CPU time linearly proportional to O(n).

I. 序 論

반도체 공정기술이 고도로 발전하여 집적회로 chip의 복잡도가 VLSI 수준에 이르게됨에 따라, 종전에는 사람의 시각관찰에 의존할 수 있었던 chip의 설계규칙 검사(design rule check ; 이하 DRC로 약칭함)가 컴퓨터의 도움 없이는 사실상 불가능하게 되었다. DRC는 집적회로설계의 오류를 지적하는데 그치고, 해결의 구체적 방법을 제공하지는 못한다 하여 일각에서는 설계규칙에 절대 어긋나지 않는 도형을 컴퓨터로 설계해 나가는 방법을 연구하고 있다. 그러나 이에 필요한 알고리즘이 아직 완전하지 못하며 향후 상당한 기간동안은 실제적으로 DRC작업은 필요성이 존재할 것으로 생각된다.

현재까지 발표된 설계규칙 검사의 방법에는 크게 두가지가 있는데, 첫째가 bit-map방법이다. bit-map방법은 각 mask를 단위길이(λ)의 정방형 mesh로 균일하게 나누고 투명한 부분과 불투명한 부분에 속하는 mesh를 각각 1과 0으로 표시 함으로써 mask data를 저장한 후에 (3×3=)9개 혹은 (4×4=)16개의 mesh로 이루어진 window로 bit-map plane(mask layer)을 scan하면서 검사하는 방식을 말한다. 이중 참고문헌[2]에 소개된 방식은 window 내부에 들어온 data의 처리를 hardware에 의해서 행함으로써 빠른 속도로 검사를 해내긴 하지만 design rule이 변할 경우에는 hard ware 논리회로 자체에 수정을 가해야하는 문제와 window를 chip전체 평면에 걸쳐서 scan하므로 pattern이 없는 부분에서는 불필요한 작업을 수

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Electrical Engineering, KAIST)

接受日字 : 1984年 10月 2日

(※ 본 논문은 한국과학재단과 과학기술처의 지원에 의하여 이루어진 것임).

행하게 되는 문제가 있다.[2]

두번째 방법은 polygon-algebra 방법으로서 layout의 수치적 data를 산술적 계산에 의해서 검사하는 방법으로 layout data를 polygon 화하는 과정(merging)을 필요로 하며 그것을 위해서는 0(nlogn)의 시간이 요구된다.[3]

이 논문에서는 상기한 두가지 방법의 장점을 살리기 위하여, mask data는 bit-map 방법으로 저장하되 mask bit pattern의 sparsity를 이용하여 bit-map plane 상에서 pattern이 있는 부분에서만 오류를 검사해나가는 방법을 택했다. 이렇게 함으로써 거의 0(n)의 시간을 소모하게 된다.

Ⅱ. 설계 규칙(Design Rule)

집적회로의 동작 조건 및 제조공정기술의 한계로 인하여, IC mask의 각 도형간에는 일련의 설계규칙이 지켜져야 한다. 이러한 설계규칙은 제조공정에 따라서 다르게 되지만 설계규칙의 기본 단위인 λ(lambda)를 도입 함으로써 상당히 융통성 있게 표현될 수 있다. 즉 λ는 공정기술에 따라서 10μm도 될 수 있고 0.5μm도 될 수 있는데 이에 의한 여러가지 설계규칙이 참고문헌[1]에 나와있다. NMOS(혹은 PMOS)의 경우, 종목별로 분류하면 표 1과 같다.

이 논문에서는 설계규칙을 기하학적 성질에 따라서

표 1. 설계 규칙 목록
Table 1. Indices of design rules.

규칙의종류	미 세 구 분	크기(단위 : λ)
자체 규칙	contact	2
	diffusion	2
	poly silicon	2
	implant	5
간격 규칙	metal	3
	contact-contact	2
	contact-poly	2
	diffusion-diffusion	3
	diffusion-poly	1
	diffusion-implant	1. 5
	poly-poly	2
	metal-metal	3
포괄 규칙	diffusion-contact	1
	poly-contact	1
	metal-contact	1
	butting contact	*
여유 규칙	diffusion on poly	2
	diffusion on implant	1. 5
	poly on diffusion	2
	poly on implant	1. 5

자체규칙, 간격규칙, 포괄규칙 및 여유규칙의 네가지로 분류하여, 각 경우에 대한 오류검사 알고리즘을 고안하였다. 첫째, 자체규칙은 어느 도형이 자체적으로 만족시켜야 하는 최소의 폭을 규정한다. 그림 1은 자체규칙을 설명하고 있다. 둘째, 간격규칙은 인접한 다

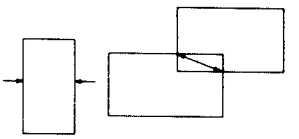


그림 1. 자체규칙
Fig. 1. Width rule.

른 도형 또는 같은 도형사이의 최소의 간격을 규정하며 그림 2에 표시하였다. 셋째, 포괄규칙은 주로 con-

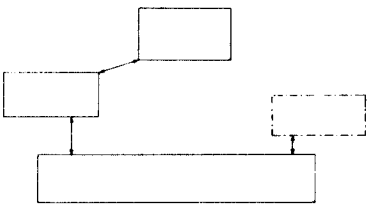


그림 2. 간격규칙
Fig. 2. Space rule.

tact 도형에 대한 것으로 diffusion, poly 혹은 metal 도형이 contact 주위를 둘러쌀 경우 contact 도형과의 최소한 거리를 나타낸다.(그림 3 참조) 그러나,

butting contact에 대해서는 일반적인 포괄규칙 알고리즘에 의거하지 않고 독립적으로 취급하였다. 그림 4는 정상적인 butting contact을 나타내고 있으며 a와 c는 그림 3의 경우와 비슷하고 b는 diffusion과 poly의 over lap을 의미한다. 여유규칙은 dffusion과 poly-

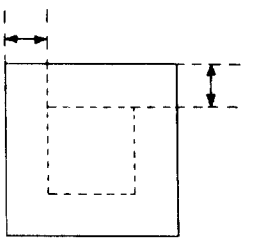


그림 3. 포괄규칙
Fig. 3. Enclosure rule.

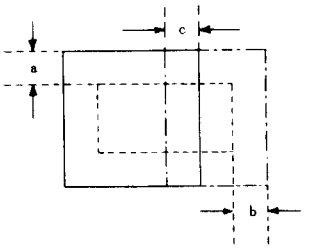


그림 4. Butting contact 규칙
Fig. 4. Rule for butting contact.

silicon이 교차하여 MOS transistor를 형성할때 공정중 misalign되거나 over-etch(과삭각) 되는 것을 고려하여 교차를 어느정도 충분히 여유있게 시키고자 하는 규정으로서 그림5에 보이고 있다.

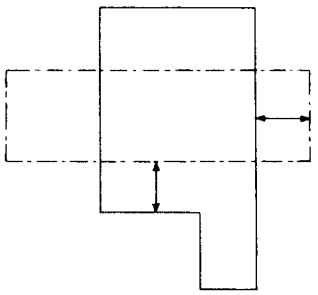


그림 5. 여유규칙
Fig. 5. Margin rule.

Ⅲ. 알고리즘

집적회로의 mask pattern을 표현하는 방법에는 여러가지가 있겠으나, 대표적인 것으로는 CIF(caltech intermediate form)에 의한 표현방법과 bit-map plane에 도형의 투명여부를 1과 0의 2차원 array로 표현하는 방법이 있다. 임의의 도형을 전자의 CIF로 표현할 수 있는 방법은 매우 여러가지가 있으므로 표현의 자유도가 높게되나, 후자의 방법을 쓰는 경우, 도형과 이를 나타내는 bit-map plane 사이에는 1 : 1 대응관계가 성립하게 된다. 물론 bit-map plane은 CIF data로 부터 만들어질 수 있다.

이 논문에서는 알고리즘의 복잡성을 피하기 위하여 모든 mask 입력 data를 일단 bit-map plane data로 바꾼후의 알고리즘을 설계하였다. 그림6은 bit-map plane의 한예를 보인 것이다. '0'은 비어있는 부분이고 '1'은 차 있는 부분이다.

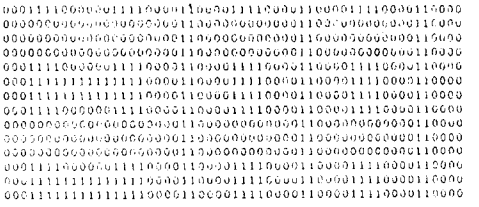
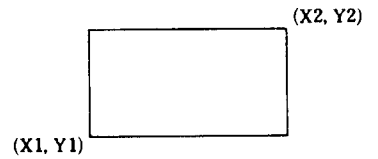


그림 6. Bit-map plane의 예
Fig. 6. An example of bit-map plane.

이 논문에서는 우선 직립사각형 즉 manhattan 도형만을 취급하기로 하는데, 직사각형에 대한 수치정보는 그림7(a)에서 보이는 바와같이 mask layer, lower left corner의 좌표와 upper right coner의 좌표로 이루어진다.



mask layer	X1	Y1	X2	Y2

그림 7. (a) 직립직사각형의 좌표
(b) Direct format의 data 구조
Fig. 7. (a) Coordinates of rectilinear box.
(b) Data structure of direct format.

그림7(b)는 한개의 직사각형에 대한 data structure로서 DF(directformat)이라고 하며, 이것을 bit-map plane data로 바꾼후에 다음에 설명하는 알고리즘에 의하여 설계규칙의 위반 여부를 검사하게 된다. 한편 초기의 간단한 규칙검사는 DF file에서 직접 이루어지게 된다. 이제 DF와 bit-map plane data를 이용하여 각각의 설계규칙 검사를 어떻게 수행하는지를 설명한다.

1. 자체 규칙(Width Rule)

자체규칙 검사는 도형자체가 가져야 할 최소폭 보다 큰지 여부를 검사하는 것으로서 다음의 두 가지를 단계를 밟는다. 첫째 단계는 그림8(a)의 경우로서 body check라고 한다. body check는 DF data로 부터 도형의 x축 방향과 y축 방향으로의 width를 계산하여 rule size(이하 RS로 약칭함)보다 크면 'pass' 시키고 만약 RS보다 작은 경우에는 그림8(a)의 오른쪽

그림과 같이 다른 도형과 만나서 결과적으로 자체규칙을 위반하지 않게 되는지의 여부를 bit-map plane data로 부터 검사한다. 자체규칙 검사의 두번째 단계는 그림8(b)에 나타나 있는대로 각각 독립적으로는 자체규칙을 만족시키는 도형일지라도 그림8(b) 그림처럼 모서리에서 같은 mask layer 상의 다른 직사각형과 충분히 겹치지 않으므로 해서 자체규칙을 위반하는 가의 여부를 검사하는 것으로서 corner check라 부른다. corner check는 각 직사각형 도형의 UR(up-upper right) corner와 LR(lower right) corner에 대해서만 하게 되는데 이 이유는 UL(upper left) corner와 LL(low left) corner에 대해서는 그 도형의 왼쪽, 아랫쪽, 혹은 좌하측에 있는 도형의 UR corner와 LR corner 검사에서 이루어지기 때문에 검사의 중복을 피하기 위한 것이다. UR corner 검사는 그림8(c), (d)에, LR corner 검사는 그림8(e), (f)에 그 과정을 각각 도시하였는데, 이중 그림8(c)의 경우만 설명하기로 한다. 그림8(g)는 bit-map plane 상에서 도형의 UR corner부분을 보이고 있다. 먼저 이 corner에서 주체도형이 다른 도형과 겹쳐 있는지를 알기 위해서 A부분의 bit-value를 읽어서 1이면 겹쳐지는 것으로 판정하여 점 B로 부터 x축 방향으로 bit 값을 읽어 더해감으로써 그림8(c)에서 보인 a값을 산출해 낸다. 만약 a값이 RS보다 작으면 그림8(c)의 P점에 정사각형의 window를 그것의 UL corner가 놓이도록 갖다 놓는다. 그리하여 이 window안에 들어오는 bit value를 읽어서 0이 발견되면 그점과 P점까지를 피타고라스 정리에 의해서 계산하여 RS와 비교한다. RS보다 작은 경우 그점 바로 위 bit value가 1이면 error message를 내보내고 0이면 pass한다. 이것은 그림의 바로 윗점의 bit 값이 '0'인 경우에는 바로 윗점에서 이미 error가 발견 되었을 것이므로 같은 error의 중복 check를 피하기 위한 것이다. 그림8(d), (e), (f)는 이 경우와 방향이 다를뿐 똑같은 과정이다.

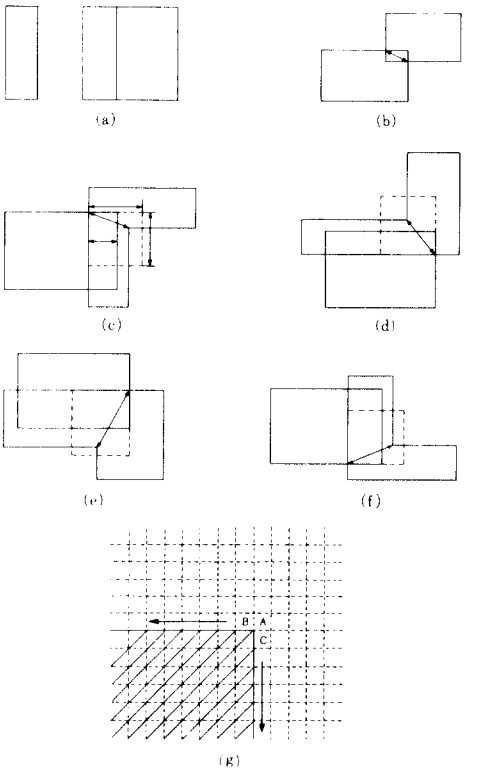


그림 8. 자체규칙 검사의 설명
(a) Body (b) Corner
(c) UR corner에서의 lower--right area scan
(d) UR corner에서의 upper-left area scan
(e) LR corner에서의 lower-left area scan
(f) LR corner에서의 upper-right area scan
(g) UR corner에서의 자체규칙 검사

Fig. 8. Descriptions of width rule check.
(a) Body. (b) Corner.
(c) Lower-right area scan at UR corner.
(d) Upper-left area scan at LR corner.
(e) Lower-left area scan at LR corner.
(f) Upper-right area scan at LR corner.
(g) Width rule check at UR corner.

2. 간격 규칙(Space Rule)

간격규칙 검사는 임의의 한도형(주체도형 이라함)과 그 주위에 있는 다른 도형(객체도형 이라하며, 주체도형과 같은, 혹은 다른 mask layer상에 있을 수 있음)과의 간격이 RS만큼 혹은 그 이상이 되는가의 여부를 검사하는 것으로서, 주체도형의 네개의 corner에서 검사가 이루어 진다. 그림9(a)는 네개의 corner에서 scan 하는 범위를 보여준다. UL corner와 LL corner에서는 diagonal 방향으로의 scan을 하지 않는데 그 이유는 UL cornor 방향과 LL corner 방향에 있는 도형의 UL corner 검사와 UR corner 검사시에 각각의 검사가 이루어지므로 중복을 피하기 위한 것이다.

이제 UL corner에서의 검사를 설명하고자 한다. 그림9(b)를 보자. 먼저, B점이나 C점에서의 주체 도형의 bit 값이 1이면 이 corner에서 pattern이 연속되므로 이 corner에서의 검사는 생략한다. B점과 C점에서의 주체도형의 bit 값이 모두 0이면 점 A로 부터 C방향으로의 horizontal scan과 점 A로 부터 B방향으로의 vertical scan을 행한다. 그런데 만약 점 A와 C에서의 객체 도형의 bit 값이 모두 1이면 객체도형과 주체 도형은 overlap 되어 있으므로 horizontal scan을

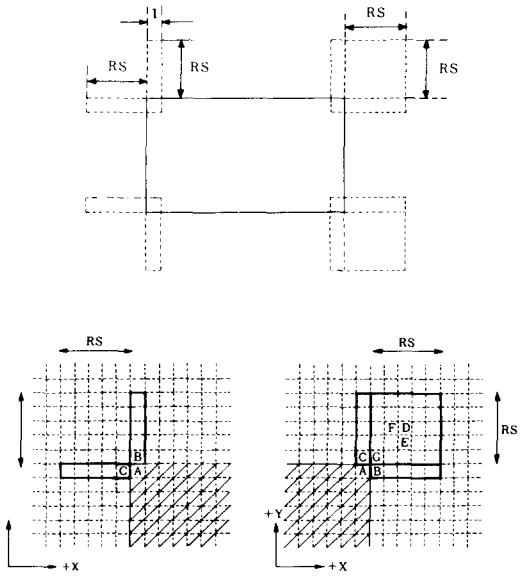


그림 9. (a) 간격규칙 검사의 scanning area
(b) UL corner에서의 간격규칙 검사
(c) UR corner에서의 간격규칙 검사
Fig. 9. (a) Scanning area of space rule check.
(b) Space rule check at UL corner.
(c) Space rule check at UR corner.

생략하고 마찬가지로 점 A와 B에서의 객체 도형의 bit 값이 모두 1일때는 vertical scan을 생략 한다.

Horizontal scan은 점 C로 부터 X 방향으로 RS 만큼 객체 도형이 있는 bit-map 평면의 bit 값을 읽어 나가다가 1이 발견되면 곧 바로 error message를 내보내고 vertical scan으로 넘어 간다. vertical scan도 horizontal scan과 유사하다. 이상과 같은 작업에 의해서 UL corner에서의 space rule을 검사하며 LL corner에서의 space rule 검사도 이와 유사하다. 이제 그림9(c)를 통하여 UR corner에서의 space rule 검사과정을 설명 하고자 한다. 검사 생략 조건을 찾아내는 과정은 UL corner check와 유사하다. 즉 그림 9(c)에서 점 B 혹은 C에서의 주체도형의 bit 값이 1이면 연속을 의미하므로 이 corner에서의 검사를 생략한다. UR corner의 검사는 horizontal scan과 vertical scan 그리고 diagonal area scan으로 이루어진다. 점 A와 B에서의 객체도형의 bit 값이 모두 1이면 horizontal scan을 생략하며, 마찬가지로 점 A와 점 C에서의 객체도형의 bit 값이 모두 1이면 vertical scan을 생략한다. horizontal scan과 vertical scan은 UL corner의 경우와 유사하다. diagonal area scan은 점 G로 부터 +X방향으로 RS 만큼의 horizontal scan을, +Y 방향으로 RS 만큼 하는 것 이다. 한 point에서의 bit 값이 0이면 다음 point도 넘어가며 1이면 그 점이 corner인지 아닌지를 검사한다. 이과정을 점 D의 경우를 예를 들면 점 D에서의 bit 값이 1이더라도 점 F 혹은 점 E에서의 bit 값이 1이면 점 D는 corner point가 아니므로 error message를 보내지 않는다. 점 F와 E에서의 bit 값이 모두 0이면 점 A로부터 점 D까지의 거리를 pythagorean theorem을 써서 계산하고 그 값이 RS 보다 작으면 error message를 내 보낸다. LR corner에서의 space rule 검사는 UR corner의 경우와 유사하다.

3. 포괄규칙(Enclosure Rule)

포괄규칙은 contact cut 주위의 각 mask layer의 도형들간에 만족 되어야 하는 규칙으로서, 일단 contact cut 내부 영역에서의 각 mask layer 상의 도형의 bit value로 부터 error여부를 식별하게 된다. 이때, contact cut 내부 영역에서의 metal, poly 및 diffusion 도형의 점유 상황의 경우를 표2에 나타내었다. 여기서 각 도형이 contact 내부영역에 꽉 차 있는 경우를 2, 부분적으로 차 있는 경우를 1, 완전히 비어있는 경우를 0로 표시하며, x는 don't care를 뜻한다.

표 2. Contact cut 내부의 상황표
Table 2. The estimate of contact cut.

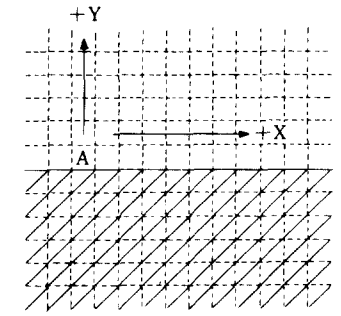
경우 / mask layer	1	2	3	4	5	6	7	8	9	10	11
metal	0	1	2	2	2	2	2	2	2	2	2
poly	X	X	0	0	0	1	1	1	2	2	2
diffusion	X	X	0	1	2	0	1	2	0	1	2

이상의 11가지 경우에 대한 진단은 다음과 같다.

1) 경우1, 2 : contact위를 metal이 완전히 덮고 있지 않으므로 위반으로 간주한다.
2) 경우3 : metal만 있고 poly나 diffusion이 없으므로 위반이다.
3) 경우4 : diffusion이 비어있는 부분이 존재하므로 bulk와 contact이 이루어져 위반이다.
4) 경우5 : diffusion과 metal의 contact이다.
5) 경우6 : poly가 비어있는 부분이 존재하므로 bulk와 contact이 이루어져 위반이다.
6) 경우7 : butting contact으로 간주한다.
7) 경우8 : butting contact으로 간주한다.
8) 경우9 : poly와 metal의 contact이다.
9) 경우10 : butting contact이 잘못된 것으로 판정한다.
10) 경우11 : butting contact이 잘못된 것으로 판정한다.

이중 경우5와 경우9에 대해서는 contact 외부에서의 diffusion, poly, metal의 포괄성을 검사하며 butting contact으로 판정된 7과 8의 경우는 butting contact의 관점에서 역시 contact 주위로 포괄성을 검사한다. 일반 contact에 해당하는 경우5와 경우9에 대한 포괄성 검사는 다음과 같다.

Contact의 상변에 대한 포괄성 검사 과정이 그림10(a)에 나타나 있다. A로 부터 +y방향으로 RS 만큼



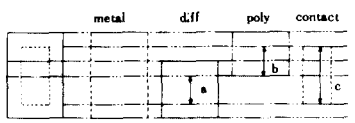


그림10. 포괄규칙의 검사
(a) Upper edge scanning
(b) Butting contact의 해부도

Fig. 10. Enclosure rule check.
(a) Upper edge scanning.
(b) Details of butting contact.

bit-map plane을 scan하면서 bit 값을 합하여 그 값이 RS보다 작으면 error로 간주한다. 이와같은 과정을 네변에 대하여 모두 행하여 준다. butting contact의 포괄성을 검사하려면 먼저 poly와 diffusion이 겹치고 있는 방향을 탐지해야 한다. 이것은 contact의 네 꼭지점에서의 polysilicon mask의 bit 값으로 부터 찾아낸다. 방향탐지가 끝나면 그림10(b)의 b로 표시된 polysilicon이 차지하는 width를 탐지한다. 또한 contact의 폭 c에서 b를 빼으로써 diffusion layer상의 실제 contact 폭 a를 계산할 수 있으며 이때 a와 b는 각각 contact의 자체규칙을 만족해야 한다. 그리고 butting contact에 대한 포괄성 검사는 일반적인 contact에서와 비슷하게 행해지거나 diffusion의 경우에는 polysilicon과의 교차 필요성 때문에 polysilicon 내부로 1λ단위만큼 들어 갈 때까지 검사한다.

4. 여유 규칙(Margin Rule)

여유규칙은 주체도형을 객체도형이 overlap 할때에 객체도형의 길이를 실제 겹치는 길이보다 어느정도 여유있게 정해 주느냐 하는 여유(margin)에 대한 규칙이다. NMOS layout에서의 여유규칙의 주체는 diffusion, polysilicon이고 객체는 diffusion, polysilicon 그리고 implant가 되며 그림11(a)는 주체도형의 상변에서의 여유규칙을 검사하는 과정을 보여준다. 점 A에서의 객체도형의 존재 여부를 탐지한다. 만약 점 A에서 객체도형이 없으면 다음 bit line으로 넘어가고, 만약 객체도형이 있으면 B에서 부터 RS만큼 +y 방향으로의 객체도형의 bit 값을 합한다. 그 값이 RS와 같으면 다음 +x 방향의 bit line으로 넘어가고 RS 보다 작으면 위반으로 간주한다. 그림11(b)의 경우는 설계자가 의도하는 바에따라 위반 여부가 정해진다. 즉 MOS transistor를 만들려고 하는것과 MOS capacitor를 만들려고 한 경우의 두가지로 해석될 수 있다. 설계자의 의도가 transistor였다면 위반이며 MOS

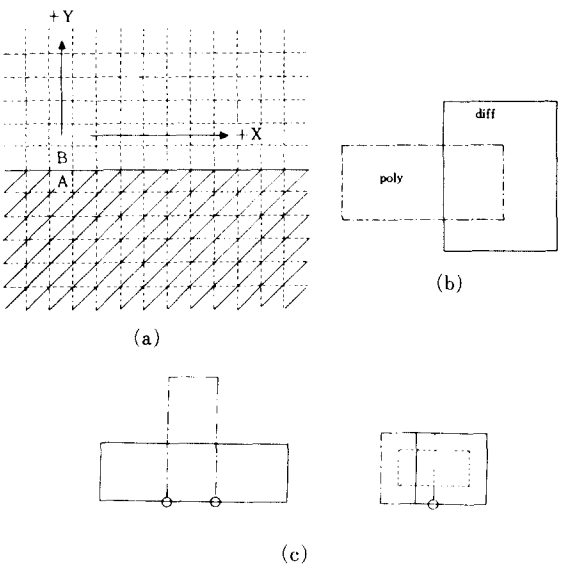


그림11. 여유규칙의 검사
(a) Upper edge scanning
(b) 모호성(capacitor or transistor?)
여기서는 capacitor로 간주됨
(c) 주체도형속에 contact이 발견되면 여유규칙 검사는 하지 않는다

Fig. 11. Margin rule check.
(a) Upper edge scanning.
(b) An ambiguity (capacitor or transistor?) (regarded as capacitor, here).
(c) When contact is found in subjective pattern, the margin rule check is passed.

capacitor였다면 위반이 아닌것으로 간주되어야 하는데 이 논문에서는 이 경우를 위반으로 간주하지 않기로 하였다. 또한 그림11(c)의 경우를 보면 diffusion과 poly가 butting contact에서 만난 왼쪽의 경우는 error가 아니지만 오른쪽의 경우처럼 contact이 없는 MOS transistor의 경우는 error로 간주되어야 할것이다. 여기서는 주체도형속에 contact pattern이 발견되면 여유규칙은 검사하지 않기로 했다.

Ⅳ. Programming 및 결과

Program 상에서 bit-map plane은 pattern의 layer 이름, X 좌표 Y 좌표의 세가지 정보를 갖는 integer array이다. NMOS layout에서 implant mask가 0.5λ의 resolution을 가지므로 bit-map plane도 0.5λ resolution을 가지게 했다. layout의 크기가 너무커서 하나의 bit-map plane에 나타낼 수 없을 때는 전체도면을 분할한다. 이때, 분할된 각 도면의 경계면은 최소한 6λ의 길이를 갖는 butting contact이 한 도면안에 완전히 들어가도록 서로 10λ만큼 교차시켰다. bit-map plane은 초기에 전체적으로 "0"으로 set 되며 DF data를 0.5λ 단위로 양의 정수값을 가지도록 scaling하여 저장하고 이것을 이용하여 bit-map plane을 만든다. DF data와 bit-map plane 상에서의 pattern data의 위치가 직접적으로 연관을 가지게 함으로써 DF data를 이용하여 쉽게 bit-map plane 상의 위치를 알 수 있게 하였다. 예를 들면 DF data가 X1=15, Y1=20, X2=25, Y2=40, IPATTERN(pattern의 종류)=2(:diffusion)이라고 하면

MAP(i, j, 2); i=16~25, j=21~40}=1

이 됨을 의미한다.

Program의 출력인 error message는 graphic terminal 상에 표현되는데 먼저 layout을 그리고 설계규칙에 위반되는 부분을 화살표와 원을 사용하여 알기 쉽게 지적한다. 그림12, 13은 output의 예이다.

Error index에 대한 설명은 표3에 보였다.

표 3. Error code의 설명
Table 3. Description of error code.

code	description
CS	contact size
DW	diffusion width
PW	poly width
IW	implant width
MW	metal width
DDG	diffusion-diffusion gap
DPG	diffusion-poly gap
PPG	poly-poly gap
CCG	contact-contact gap
IDG	implant-diffusion gap
MMG	metal-metal gap
PDM	poly-diffusion margin
DPM	diffusion-poly margin
IPM	implant-poly margin
IDM	implant-diffusion margin
DCE	diffusion-contact enclosure
PCE	poly-contact enclosu re
MCE	metal-contact enclosure
BTC	butting contact rule

그림12는 shift register의 일부분인데 고의로 layout 상의 error를 발생시킨 것으로서 정확하게 error가 지적되고 있음을 볼 수 있다. 그림13은 PLA의 input buffer 부분을 설계한 layout인데 poly 사이의 gap이 1λ밖에 되지 않아서 필요치 2λ에 미달 되기 때문에 error가 발생된 것을 보이고 있다. VAX 11/780

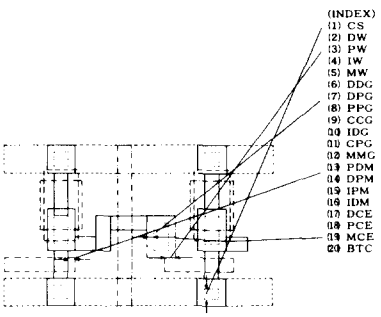


그림12. 출력 예(dynamic shift register)
Fig. 12. Output example(dynamic shift register).

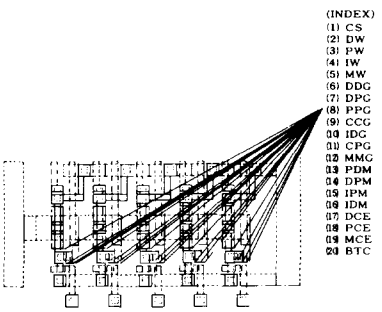


그림13. 출력 예(PLA의 input buffer 부분)
Poly-to-poly 간격규칙 위반이 발견됨
Fig. 13. Output example(input buffer parts of PLA)
Poly-to-poly space rule errors are found.

computer로 1 stage당 3개의 error message를 갖는 shift register를 반복시킨 layout data를 input으로 하여 pattern의 갯수와 check CPU time의 관계를 나타낸 것이 그림14의 graph로서 CPU time이 도형의 갯수에 비례하는 것을 알 수 있다.

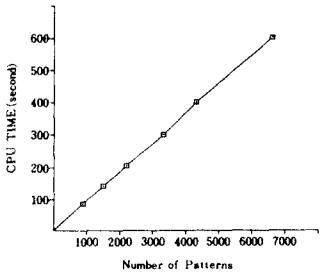


그림14. 도형의 갯수와 CPU time의 곡선
Fig. 14. Curve of number of pattern and CPU time.

V. 結 論

각 pattern의 근처만 scanning을 함으로써 도형의 갯수에 비례하는 CPU time을 소모하는 DRC algorithm을 제안하고, 이를 FORTRAN program으로 실현하였다. 일반적으로 sorting 과정을 통한 DRC algorithm의 소모시간은 0 (nlogn)에 비례하는 것으로 알려져 있는데 반하여, 각 pattern의 근처만 scanning을 하는 이 논문에서 제안된 bit-map 방법은 일단 bit map 평면이 완성되면 data의 sorting을 따로 하지 않으므로 pattern의 갯수에 비례하는 CPU time을 소모하게 된다.

이 논문의 DRC program은 NMOS에 대하여만 언급되었으나, CMOS IC에 대한 DRC program으로도 확장 개선될 수 있을 것으로 생각된다.

參 考 文 獻

[1] Carver Mead & Lynn Conway, *Introduction to VLSI Systems.* Addison Wesley, 1980.

[2] Larry Seller, *A Hardware Assisted Design Rule Check Architecture.* 19th Design Automation Conference, p. 232, 1982.

[3] Michael H. Arnold and John K, Ousterhourt, *Lyra; A New Approach to Geometric Layout Rule Checking.* 19th, Design Automation Conference, p. 530, 1982.