論文93-30A-5-7

Gate Matrix 레이아웃 생성 시스템의 구현

(Implementation of a Layout Generation System for the Gate Matrix Style)

金 相 範*, 黃 善 泳**

(Sahng Beom Kim and Sun Young Hwang)

要 約

본 논문은 다단 논리 구현을 위한 gate matrix 레이아웃 생성 시스템에 대해 기술한다. 본 논문에서 제안된 시스템은 일반 네트 리스트 입력으로부터 보다 개선된 레이아웃을 생성하기 위해, 직렬 네트들에 대해 변경이 가능한 네트 바인딩을 수행한다. 또한 본 시스템은 네트의 길이가 짧을수록 트랙 최소화에 유리하다는 휴리스틱 정보를 통해 게이트의 재할당을 실시한다. 한편 본 시스템은 세분화시킨 레이아웃 열 정보를 통해 트랙 최소화를 실시함으로써 필요 트랙의 수를 감소시킨다. 실험 결과 네트 리스트 입력에 근거한 기존의 gate matrix 레이아웃 생성 시스템보다 본 시스템이 7.46% 정도 면적이 축소된 레이아웃을 생성함을 확인하였다.

Abstract

This paper describes the implementation of a layout generation system for the gate matrix style to implement multi-level logic. To achieve improved layouts from general net lists, the proposed system performs flexible net binding for series nets. Also the system reassigns gates by the heuristic information that shorter net lengths are better for the track minimization. By track minimizing with subdividing layout column information, the system decreases the number of necessary layout tracks. Experimental results show that the system generates more area-reduced (approximately 7.46%) layouts than those by previous gate matrix generation systems using net list inputs.

I. 서론

반도체 기술의 급격한 발전은 VLSI 칩 개발에 있어서 설계의 효율성 향상을 위한 설계 자동화 기법을 요구하였고, 이에 따라 상위 수준 언어로 VLSI 칩 설계가 가능한 실리콘 컴파일러[1]가 개발되었다. 모듈 생성 시스템은 실리콘 컴파일러에서 실제 레이아웃을 담당하는 중요 구성 요소로서 특정 입력 동작, 설계 사양으로부터 구조적, 기하학적 기술 형태를 생성하는 프로그램이다.[1] 디지탈 회로의 레이아웃에 있어서 랜덤 로직 모듈은 트랜지스터, 게이트와 같은 기본 소자로 이루어진 불규칙적인 구조의 모듈로 gate matrix[2], Weinberger array[3], metal metal matrix[4] 등의 형태로 레이아웃이 형성된다.

Gate matrix 레이아웃 형태는 동일 간격의 수직 poly 선들이 트랜지스터의 게이트와 회로 각 부분의

*準會員, **正會員, 西江大學校 電子工學科
(Dept. of Elec. Eng., Sogang Univ.)
接受日字: 1993年 1月 15日

연결을 담당하도록 하고, 수평 방향으로 diffusion을 배치하여 수직 poly 선과의 교차에 의해 트랜지스터가 형성되도록 한 후, 메탈에 의해 트랜지스터 간의 배선과 Vdd와 Vss 간의 배선을 실시하는 방식이며 1980 년에 A. D. Lopez와 H.-F. S. Law에 의해 제안되었다.[2] 랜덤 로직을 최적의 게이트 할당 순서를 통해 최소 면적의 gate matrix 형태의 레이아웃으로 구현하는 문제는 NP-complete이며[5] 이를 해결하기 위한 휴리스틱 알고리듬이 개발되어 왔다.[6-9] O. Wing등[7]은 간격 그래프 (interval graph)와 dominant clique[10] 개념을 통한 게이트 할당으로 레이아웃을 구현하였으나, 동일한 회로 기능을 수행하는 다양한 네트 바인딩에 대한 고려가 이루어지지 않음으로써 네트 리스트를 통해 결정된 네트 바인딩에 의해 게이트 할당이 제한된다. D. K. Hwang등[8]은 다이나믹 네트 리스트를 도입하여 게이트 할당이 이루어진 이후에 직렬 네트에 대한 네트 바인딩을 시도하여 O. Wing등의 방식을 개선했다. U. Singh과 C. Y. Roger Chen[9]은 positive logic boolean function[9] 형태의 입력을 통해 직렬 트랜지스터들만의 재배치를 고려하는 기존의 다이나믹 네트 리스트 방식을 확장하여 논리식에 근거한 입력을 통해 트랜지스터의 재배치를 시도하였다.

본 논문에서는 구현된 GAGEN (GAte matrix GENeration system) 시스템에 대해 기술한다. GAGEN은 네트 바인딩 지연을 위해 작성된 다이나믹 네트 리스트가 아닌 일반 네트 리스트를 입력으로 하여 게이트 할당을 한 후, 내부적으로 직렬 네트에 대한 네트 바인딩의 재시도가 가능한 flexible 네트 바인딩을 통해 직렬 네트의 바인딩 문제를 해결하고, 네트 병합과 세분화된 열 정보를 통한 트랙 최소화 과정을 거쳐 레이아웃 면적을 최소화한 후, 최종적으로 gate matrix 형태의 심볼릭 레이아웃을 출력한다. CMOS 회로 구현에 있어서 PMOS 부분과 NMOS 부분의 게이트 할당 순서가 같다는 duality 가정에 따라 본 논문에서는 NMOS 부분의 레이아웃만을 제시하며, Vss로의 연결은 두번째 메탈 층에 의해 이루어진다고 가정한다.[7]

Ⅱ. 시스템 개관

GAGEN 시스템의 전체적 구성은 그림 1과 같다. 트랜지스터 회로도로부터 추출된 네트 리스트를 입력으로 받아들여 간격 그래프와 v.d.c. 매트릭스 (vertex versus dominant clique matrix)[10]로 모델링하여 게이트 할당을 실시한다. 매트릭스의 행은 그래프의 vertex에 해당되고 열은 그래프의 dominant clique에 해당되는 v.d.c. 매트릭스를 간격 그래프에 대해 구성하면 구성된 v.d.c. 매트릭스는 매트릭스의 행 또는 열에서 연속적인 '1' 의 값을 나타내는 consecutive ones property를 갖는다.[10] 이 특성을 통해 각 네트가 매트릭스 상에서 구성되면 네트 간의 연결이 이루어지며 이때 직렬 네트와의 네트 바인딩은 최적 해를 얻기 위해 수정된다. 이후 수직 연결된 둘 이상의 네트를 하나로 병합하는 네트 병합과 필요 트랙의 수를 줄이는 트랙 최소화 과정을 통해 면적이 축소된 초기 레이아웃에서 얻어진 정보를 이용하여 게이트의 재할당을 실시하고 다시 네트 병합과 트랙 최소화 과정을 거쳐 면적 면에서 개선된 레이아웃을 얻는다. 구해진 매트릭스 형태의 레이아웃은 수직 방향의 연결을 위해 실제 레이아웃 상에서 diffusion run을 통한 배선이 이루어진다. 이때 diffusion run과 다른 diffusion 층 사이에 겹침이 없는 네트 배치가 이루어지기 위해 트랙 재배치 과정이 수행되고 이후 심볼릭 레이아웃 형태의 결과가 출력된다.

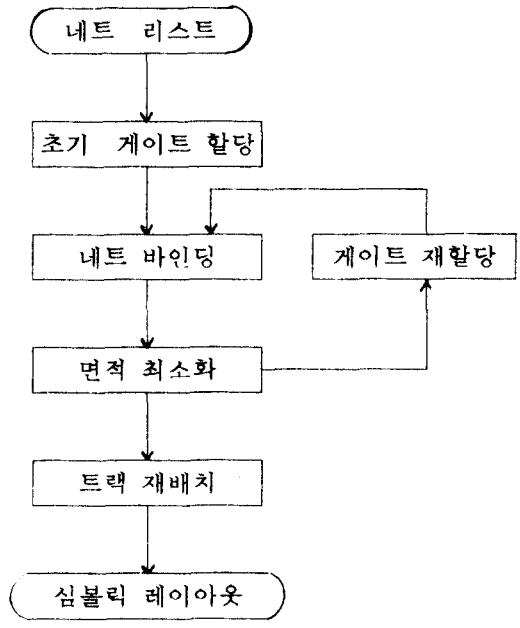


그림 1. 시스템 구성도
Fig. 1. System flow diagram.

Ⅲ. 게이트 할당 및 네트 바인딩

게이트 할당은 레이아웃 면적에 직접적인 영향을 준다. 그림 2 (a) 회로에 대해 그림 2 (b)의 게이트

할당은 4개의 트랙을 요구하나 게이트 할당 순서를 변경함에 따라 그림 2 (c)에서는 3개, 그림 2 (d)에서는 2개의 트랙만을 요구함을 알 수 있다. GAGEN에서 게이트 할당은 간격 그래프를 이용한 초기 게이트 할당과 생성된 초기 레이아웃에 대해 레이아웃 개선을 시도하는 게이트 재할당으로 구분된다.

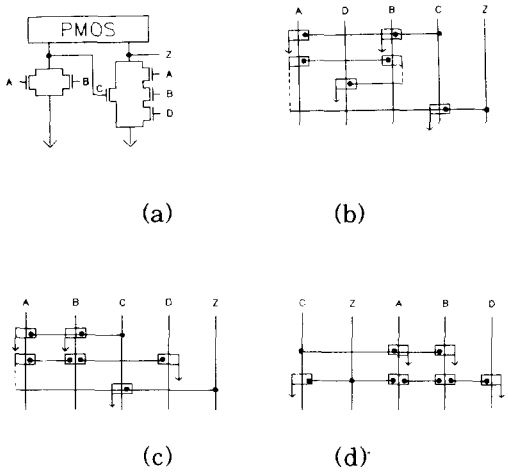


그림 2. 게이트 할당과 필요 트랙 수
(a) 예제 회로
(b) 4개의 트랙을 요구하는 게이트 할당
(c) 3개의 트랙을 요구하는 게이트 할당
(d) 2개의 트랙을 요구하는 게이트 할당

Fig. 2. Gate assignments and necessary tracks.
(a) example circuit,
(b) gate assignment requiring 4 tracks,
(c) gate assignment requiring 3 tracks,
(d) gate assignment requiring 2 tracks.

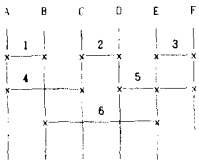
1. 초기 게이트 할당 [7]

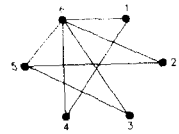
초기 게이트 할당을 위해 각 게이트에 대해 그 게이트를 포함하고 있는 네트를 나열한 게이트-네트 테이블을 구성한 후, 네트 ID를 vertex로 취하고 공통의 게이트를 취하는 네트들에 대해서 해당하는 vertex 사이를 edge로 연결한 연결 그래프[11] H = (V, E) 를 구성한다. 구성된 그래프 H는 레이아웃 상에서 네트가 차지하는 간격이 vertex로 대응되는 간격 그래프의 부그래프이다.[11] 연결 그래프 H로부터 구성된 v.d.c. 매트릭스에서 consecutive ones property를 만족시키기 위한 fill-in[10]에 해당하는 edge를 추가하면, 연결 그래프 H로부터 레이아웃을 표현하는 간격 그래프를 구성할 수 있다. 최소 트랙 수를 실현하는 간격 그래프를 찾으려면 그래프 H의 가장 큰 dominant clique의 크기 증가가 최소가 되도록 edge를 추가하여 간격 그래프를 구성한다. 이후 구성된 간격 그래프에 따라 게이트를 할당하는 것이 레이아웃의 면적을 줄일 수 있는 초기 게이트 할당이 된다.[7] 간격 그래프로부터 구성된 v.d.c. 매트릭스의 각 dominant clique를 적절히 배열하면 consecutive ones property를 만족시키는 v.d.c. 매트릭스 구성이 가능하다. 또한 lexicographic breadth-first search[15]를 도입하면 그래프의 vertex와 edge를 V와 E로 정의할 때 그래프 내의 모든 dominant clique를 O(|V |+ |E |)의 시간 내에 추출할 수 있다.[10]

그림 3 (a), (b)는 동일 회로에 대해 게이트 할당의 차이에 따른 서로 다른 레이아웃과 간격 그래프를 나타내며, 그림 3 (a)는 효과적인 게이트 할당을 통해 레이아웃의 트랙 수가 감소된 경우이다. 여기서 사용된 회로의 네트 리스트로부터 구성된 연결 그래프 H는 그림 3 (c)와 같다. H에서 추출한 모든 dominamt clique들은 (3 5 6), (1 6), (1 4), (2 5), (2 4)이며 이에 따른 v.d.c. 매트릭스는 그림 3 (d)이다.

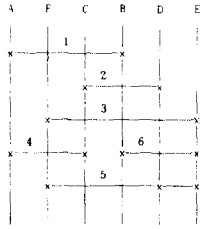
이후 consecutive ones property를 만족시키기 위한 fill-in을 도입했을 때 v.d.c. 매트릭스 내에서 '1'의 수가 가장 많은 칼럼의 '1'의 수가 최소로 되도록 칼럼의 재배치를 시도한 경우가 그림 3 (e)이며 여기서 f는 fill-in이다. 이후 게이트-네트 테이블을 통해 특정 게이트를 포함한 네트들이 차지하는 v.d.c. 매트릭스의 행의 공통 부분을 찾고 공통 부분에 해당하는 칼럼에 특정 게이트를 할당한 결과가 그림 3 (f)이며 이를 트랙 최소화시키면 그림 3 (a)의 레이아웃이 된다.

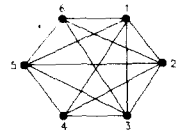
그림 4는 GAGEN에서 사용한 초기 게이트 할당 알고리듬이다. 먼저 연결 그래프 H에서 가장 큰 dominant clique와 이와 인접한 clique를 찾는다. Consecutive ones property 만족을 위한 fill-in이 추가되었을 때 가장 큰 dominant clique의 크기가 적게 증가하도록 앞서 찾은 clique에 해당하는 v.d.c. 매트릭스 상의 열을 재배치한다. 이후 게이트-네트 테이블을 참조하여, 각 게이트에 대해 그 게이트가 포함된 모든 네트의 공통 열 부분으로 정의되는 공통 간격을 찾아 이 간격에 포함된 열 중에 하나를 선택하여 해당 게이트를 할당한다.



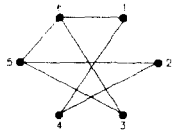


(a)





(b)



1	0	1	1	0	0
2	0	0	0	1	1
3	1	0	0	0	0
4	0	0	1	0	1
5	1	0	0	1	0
6	1	1	0	0	0

(c) (d)

1	1	1	0	0	0
2	0	0	1	1	0
3	0	0	0	0	1
4	1	f	1	0	0
5	0	0	0	1	1
6	0	1	f	f	1

(e)

	A	B	C	D	E	F
1	1	1	0	0	0	0
2	0	0	1	1	0	0
3	0	0	0	0	1	1
4	1	1	1	0	0	0
5	0	0	0	1	1	1
6	0	1	1	1	1	0

(f)

그림 3. 초기 게이트 할당 과정
(a) 3 트랙을 요구하는 레이아웃과 이에대한 간격 그래프
(b) 5 트랙을 요구하는 레이아웃과 이에대한 간격 그래프
(c) (a),(b)로부터 구성된 연결 그래프 H
(d) H의 dominant clique들로 구성된 v.d.c. 매트릭스
(e) 칼럼 재배치와 fill-in이 추가된 v.d.c. 매트릭스
(f) 각 칼럼에 게이트가 할당된 매트릭스

Fig. 3. Process of initial gate assignment.
(a) layout requiring 3 tracks and its interval graph,
(b) layout requiring 5 tracks and its interval graph,
(c) composed connection graph H from (a),(b),
(d) v.d.c. matrix composed by dominant cliques of H,
(e) v.d.c. matrix after column reassignment and fill-in addition,
(f) matrix after gate assignment to every column.

```
procedure Initial_gate_assignment (M: v.d.c. matrix)
begin
   CL = column of largest dominant clique in M;
   adj(CL) = columns of adjacent cliques of CL;
   LE = RE = CL; {left edge, right edge}
   while adj(CL) is not empty do begin
      Select a unmoved column in adj(CL) that makes the smallest increase of
        # 1's in CL when fill-ins are applied;
      Move the selected column to the better position between leftside of LE
        and rightside of RE;
      if the selected column is moved to leftside of LE then
        LE = selected column;
      else
        RE = selected column;
      if adj(CL) is empty then
        adj(CL) = unmoved columns of adjacent cliques of LE;
      if adj(CL) is empty then
        adj(CL) = unmoved columns of adjacent cliques of RE;
   end
   Find common intervals for all gates from the gate-net table;
      {A common interval is a set of common columns for all nets including
        specified gate}
   Assign all gates to columns of v.d.c. matrix by common intervals;
end;
```

그림 4. GAGEN의 초기 게이트 할당 알고리듬
Fig. 4. Initial gate assignment algorithm of GAGEN.

2. 게이트 재할당

간격 그래프에 의한 초기 게이트 할당은 부분 최적해일 수 있으며, 게이트 할당 이후 이루어지는 flexible 네트 바인딩 과정에서 네트의 구성이 부분적으로 변경된다. 따라서 초기 게이트 할당을 통해 면적 최소화를 이룬 초기 레이아웃은 개선될 여지가 있고 이를 위한 게이트 재할당 과정이 필요하다.

그림 5는 게이트 재할당 알고리듬이다. 먼저 트랙 최소화를 위해서는 네트의 길이가 짧을수록 유리하다[9] 는 휴리스틱 정보를 응용하기 위해 초기 게이트 할당을 통해 생성된 초기 레이아웃에서 가장 긴 네트를 선택한다. 선택된 네트가 포함하고 있는 게이트들을 상호 인접하게 재할당하여 네트 길이를 축소한 후 면적 최소화를 실현하여 그 결과가 초기 레이아웃보다 개선되면 이를 초기 레이아웃으로 설정하고 이 과정을 반복한다. 수직 diffusion run으로 연결된 2 개 이상의 네트가 상호 겹침 없이 한 트랙에 할당되어

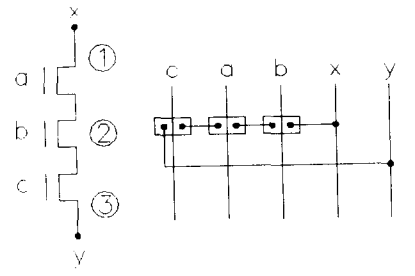
하나의 네트로 간주되는 것을 네트의 병합이라 하며, 초기 레이아웃에서 게이트를 가장 많이 포함한 병합된 네트를 선택하여 그 게이트의 순서대로 게이트 순서 리스트를 구성한다. 병합이 이루어진 네트의 게이트 순서 유지는 트랙 최소화에 유리한 네트 병합이 지속되도록 한다. 만일 가장 긴 네트 내에 포함된 게이트들이 앞서 구성된 게이트 순서 리스트의 한 게이트와 공통이면 네트 내에 포함된 게이트들은 게이트 순서 리스트에 추가된다. 게이트 순서 리스트에 포함되어 있지 않은 게이트들은 이미 구성된 네트-게이트 테이블을 참고하여, 동일 네트에 속한 게이트들이 상호 인접하게 배치되도록 한 후 게이트 순서 리스트에 삽입된다. 이후 게이트 순서 리스트에 따라 게이트가 재할당된다.

```
procedure Gate_reassignment (initial_layout)
begin
   repeat
      LN = longest net of the initial_layout;
      Reassign gates of LN such that gates are positioned closely;
      Get the new_layout after the net processing;
      if the number of tracks is decreased then
         initial_layout = new_layout;
   until the number of tracks is not decreased;
   MN = merged net of largest number of gates;
   Make the gate order list GLIST by the gate order of MN;
   LN = longest net of initial_layout;
   if LN has just one common gate of GLIST then
      Insert gates of LN to head or tail of GLIST;
   RLIST = gates not included in GLIST; {remaining gate list}
   for each gate g of RLIST do begin
      Find the net n included in g from net-gate table;
      if there is no net included in g then
         Insert g after tail of GLIST;
      else begin
         for all gates of n do begin
            if GLIST includes the gate of n then
               Keep the gate position POS of GLIST;
         end
         Select the latest POS;
         Insert g after the latest POS;
      end
   end
   Reassign gates by GLIST;
   Get the new_layout after the net processing;
   if the number of tracks is decreased then
      Choose new_layout;
end;
```

그림 5. 게이트 재할당 알고리듬
Fig. 5. Gate reassignment algorithm.

3. 네트 바인딩

직렬로 연결된 트랜지스터들은 그 연결 순서가 서로 바뀌어도 회로 기능상의 변화가 없으므로 GAGEN에서는 각 직렬 네트에 속한 트랜지스터들을 재배열하여 한 트랙에 할당한다.



(a) (b)

c a b x y
1 1 1 1 0
1 1 1 1 1

c a b x y
1 1 1 0 0
0 0 1 1 0
1 1 1 1 1

c a b x y
1 1 1 0 0
0 1 1 1 0
1 1 1 1 1

(c) (d) (e)

그림 6. 레이아웃과 매트릭스 표현
(a) 직렬 트랜지스터 회로
(b) 레이아웃 상에 배치된 (a) 회로
(c) v.d.c. 매트릭스로 표현된 (b) 레이아웃
(d) v.d.c. 매트릭스로 표현된 네트 병합 이전의 레이아웃
(e) 입력 네트 리스트로부터 구성된 v.d.c. 매트릭스

Fig. 6. Layout and matrix representations.
(a) series transistor circuit,
(b) (a) circuit placed on layout,
(c) v.d.c. matrix representation of (b) layout,
(d) v.d.c. matrix of the layout before net merging,
(e) v.d.c. matrix from the input net list.

게이트 할당이 c-a-b-x-y 의 순서로 요구될 때, 그림 6 (a) 회로의 직렬 트랜지스터는 재배열되어 그림 6 (b)와 같이 구현되는 것이 레이아웃 면에서 효과적이다.[8] 그림 6 (b)를 v.d.c. 매트릭스로 표현하면 그림 6 (c)와 같고 네트 병합이 이루어 지기 이전 상태는 그림 6 (d)와 같다. 한편 입력 네트 리스트 정보만으로 구성된 v.d.c. 매트릭스는 그림 6 (e)와 같으며 효과적인 레이아웃을 위해서 이를 그림 6 (d)로 변환하는 작업이 필요하다. 이는 입력 네트 리스트 상에서 구성된 직렬 네트에 대한 바인딩이 효과

적이지 못했기 때문에 일어나는 결과이며, GAGEN에서는 직렬 네트에 대한 네트 바인딩의 변경을 내부적으로 허용하는 flexible 네트 바인딩을 통해 이 문제를 해결한다. Flexible 네트 바인딩은 네트 리스트를 통해 이미 이루어진 직렬 네트에 대한 네트 바인딩을 수정하여 네트 병합 이후 면적 최소화시에 보다 최적화된 결과를 얻고자 하는 과정이다.

그림 7은 flexible 네트 바인딩의 알고리듬이다. 한 직렬 네트에 속한 트랜지스터들은 게이트 할당이 수행되고나면 게이트들의 순서가 변경될 수 있으므로 트랜지스터 상호 간의 연결 정보를 수정해야 한다. 따라서 직렬 네트에 속한 트랜지스터를 재배열하여 한 트랙에 할당한 후 트랜지스터 간의 연결 정보를 수정한다. 이후 구성된 v.d.c. 매트릭스 상에서 트랙에 할당된 각 네트의 시작 열과 끝 열을 리스트에 저장한다. 직렬 네트로 연결되어지는 네트에서 다른 네트와의 연결 정보만을 담당하는 연결 게이트를 제거하고 직렬 네트의 시작 열과 끝 열 중에서 거리 상으로 가까운 열을 선택하여 선택된 열에 연결 게이트를 생성하면 연결에 필요한 배선의 길이는 감소된다.

```
procedure Flexible_net_binding (M: v.d.c. matrix)
begin
   Rearrange TRs of series nets by the gate order;
   Assign tracks of M to all nets;
   Make the lists of first and last gate column IDs for all nets of M;
   for each net n of nets in M do begin
      if n is connected to series net s then begin
         mid_val := (last gate column ID of n + first gate column ID of n)/2;
         f_dist := |mid_val - first gate column ID of s|;
         l_dist := |mid_val - last gate column ID of s|;
         Remove the linkage gate of n;
         if f_dist < l_dist then
            Make new linkage gate of n to first gate column of s;
         else
            Make new linkage gate of n to last gate column of s;
      end
   end
end;
```

그림 7. Flexible 네트 바인딩 알고리듬
Fig. 7. Flexible net binding algorithm.

Ⅳ. 면적 최소화

Gate matrix 형식의 레이아웃 면적 최소화는 수직 배선을 통해 연결되어 있는 네트들을 하나의 트랙에 할당하는 네트 병합과, 각 트랙의 빈 영역을 활용하여 한 트랙 내에 서로 겹치지 않는 범위 내에서 여러 네트를 할당하는 트랙 최소화를 통해 이루어 진다. GAGEN에서는 퍼스낼리티 매트릭스 형태로 표현된 레이아웃을 통해 이 과정이 수행되며 매트릭스 데이타 구조의 한 아이템은 그림 8과 같다. 여기서 gate_kind 필드는 매트릭스의 한 아이템이 트랜지스터, contact, 또는 다른 네트로 연결을 담당하는 diffusion run인가를 구분하기 위한 필드이며, position 필드는 트랙 최소화 과정에서 보다 세분화된 left-egde를 위해 사용되는 필드이다. 매트릭스 아이템의 gate_kind 필드가 diffusion run이면 연결되는 네트의 row_ID는 connected_row_ID 필드에 저장된다. 각 네트의 시작점과 끝점 구분을 파악하기 위해서는 end_kind 필드가 사용된다.

```
struct personality_matrix_of_GAGEN {
        int   row_ID;
        int   column_ID;
        int   state;              /* 1 or 0 */
        char  *gate_kind;         /* from net list */
        int   position;           /* for precise left-edge */
        int   connected_row_ID;   /* linked row by diffusion run */
        int   end_kind;           /* net start or net end */
        struct layout_matrix *h_link;
        struct layout_matrix *v_link;
        };
```

그림 8. 퍼스낼리티 매트릭스의 데이타 구조
Fig. 8. Data structure for personality matrix.

1. 네트 병합

네트 병합은 레이아웃의 트랙 수를 줄이는 효과적 기법으로 이는 네트 간의 공통 트랜지스터 사용을 위해 diffusion run을 통한 수직 연결이 있을 경우에만 가능하다. 그림 9은 퍼스낼리티 매트릭스 내의 정보를 통해 상호 연결된 네트 간의 병합에 대한 알고리듬이다. Diffusion run을 통해 상호 연결된 두 네트를 찾아 두 네트가 상호 겹침 없이 한 트랙에 할당될 수 있으면 한 트랙에 할당하고 두 트랙을 한 트랙으로 병합시킨다.

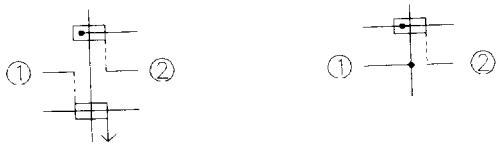
```
procedure Net_merge (P: personality_matrix)
begin
   for each row r1 of P do begin
      if r1 is linked to other net by diffusion run then begin
         Find connected row r2 by searching connected_row_ID field of r1;
         if no conflict between r1 and r2 when merging then
            Merge r1, r2;
      end
   end
end;
```

그림 9. 네트 병합 알고리듬
Fig. 9. Net merging algorithm.

2. 트랙 최소화

GAGEN에서는 각 네트들을 가능한 적은 트랙 내

에 모두 할당하기 위해 position 필드가 고려된 left-edge 알고리듬을 적용한다. 효과적인 트랙 최소화를 위해 그림 10 (a), (b)의 , 네트처럼 동일한 열을 차지하면서도 상호 중복을 일으키지 않는 두 네트를 한 트랙에 할당하는 경우도 고려해야 하나[12] 열 ID만으로 edge를 정한 기존 gate matrix 생성 시스템은 이를 처리할 수가 없다. 본 시스템은 이를 처리하기 위해서 퍼스낼리티 매트릭스 아이템의 gate_kind에 따라 네트의 edge 분류를 세분화하여 position 필드에 정보를 저장한 후, 동일한 열 ID에 대해서는 position 필드까지 고려하여 left-edge에 의한 정렬을 하고, 트랙 최소화 알고리듬 적용시 한 열에서 네트 간의 겹침이 일어날 경우 열의 position 필드까지 고려하여 실제로 겹침이 일어나는가를 확인하여 트랙 병합을 실시한다. 그림 11에 position 필드 할당의 예를 보였다.



(a) (b)

그림 10. 한 트랙에 할당된 두 네트
Fig. 10. Two nets assigned to a track.

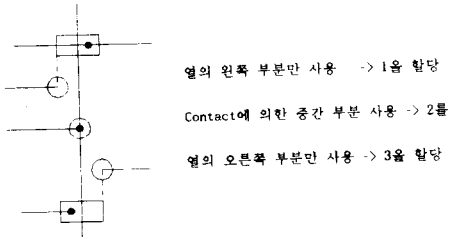
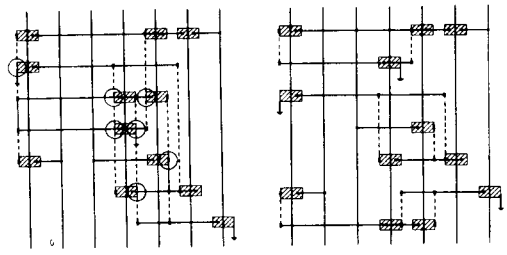


그림 11. Position 필드 할당의 예
Fig. 11. Example of position field assignment.

3. 트랙 재배치

트랙 최소화가 끝난 레이아웃은 네트 간의 수직 배선을 위한 diffusion run을 포함하고 있다. 그림 12 (a)와 같이 diffusion run이 트랜지스터나 다른 diffusion run과 만나면 회로의 오동작을 초래하므로, 그림 12 (b)와 같이 diffusion run이 다른 diffusion 레이어와 만나지 않도록 트랙들을 재배치할 필요가 있다. 트랙 재배치가 실현되지 못할 경우에는 열 사이의 간격을 넓혀서 처리한다.[7] GAGEN의 트랙 재배치 알고리듬은 RC 지연 시간 단축을 위해 트랙 내에 포함된 각 네트들을 이동하여 diffusion run의 길이를 감소시키는 과정도 수행한다.



(a) (b)

그림 12. 트랙 재배치 과정의 필요성
(a) 트랙 재배치 과정 적용 전
(b) 트랙 재배치 과정 적용 후
Fig. 12. Necessity of track realignment process.
(a) before track realignment process,
(b) after track realignment process.

```
procedure Track_realignment (personality_matrix)
begin
   loop_limit = reasonable large integer; {to avoid infinite loop}
   r_flag = NO;
   k = 0;
   DR = ϕ; {set of diffusion runs}
   if the layout includes layer confliction then begin
      repeat
         Move a conflicting row to outside of vertical interval by
            diffusion run;
         Increase k;
         if no layer confliction in the layout then
            r_flag = YES;
      until r_flag = YES or k ≥ loop_limit;
   end
   if r_flag = NO then begin
      Enlarge conflicting space between columns;
      Separate conflicting diffusion layers;
   end
   DR = all diffusion runs whose lengths are more than 2 track pitches;
   for all duffusion runs of DR then begin
      if one of two rows connected by diffusion run can be switched by other
            row to reduce diffusion run length then
         Switch rows;
      if there is empty track space to move a net of diffusion run to reduce
            diffusion run length then
         Move the net to the empty track space;
      Update DR;
   end
end;
```

그림 13. 트랙 재배치 알고리듬
Fig. 13. Track realignment algorithm.

그림 13는 GAGEN의 트랙 재배치 알고리듬이다. 수직 diffusion run과 diffusion 레이어 간의 겹침을 일으키는 행이 존재하면 이러한 행들을 수직 diffusion run이 차지하는 수직 영역 외부 트랙으로 이동시켜 겹침을 막는다. 이러한 과정을 수직 diffusion run으로 연결되는 모든 행에 대해서 실시한다. 레이어 간의 겹침이 해소되지 않으면 열 간격을 넓혀서 겹침을 일으키는 레이어를 분리시킨다. 이후 길이가 2 트랙 피치 이상인 diffusion run으로 연결된 네트에 대해 이러한 네트를 포함한 트랙 간의 상호 위치 교환, 또는 다른 트랙의 빈 공간에 diffusion run으로 연결된 네트의 이동을 통해 diffusion run의 길이를 감소시킨다.

V. 실험 결과

GAGEN은 SUN SPARC 워크스테이션의 UNIX 운영체제 하에서 C 언어로 구현되었다. 예제 회로에 대한 심볼릭 레이아웃을 그림 14에 나타내었다. 그림 14 (b)는 참고문헌 [7] 에 의한 결과로 5 개의 트랙과 3 개의 diffusion run을 필요로 하며, 그림 14 (c)는 참고문헌 [8] 에 의한 결과로 4 개의 트랙과 3 개의 diffusion run을 필요로 한다. GAGEN이 생성한 결과는 그림 14 (d)에 보였으며 4개의 트랙을 요구하면서 수직 diffusion ru· 은 오직 하나만 필요로 함을 알 수 있다.

표 1은 기존 gate matrix 레이아웃 생성 시스템과 GAGEN의 레이아웃 결과에 대한 비교로서 exp 1은 그림 14의 기본 입력 예제이며 exp 2, exp 3, exp 4, exp 5는 참고문헌 [9] 에서 사용한 회로들이며 CLA와 ALU는 참고문헌 [9] 의 논리식에 근거한 회로이다. 한편 exp 6와 full adder는 참고문헌 [13] 에서 사용한 회로이다. 참고문헌 [8] 의 레이아웃은 다이나믹 네트 리스트에 의한 네트 바인딩 지연과 min-cut 알고리듬 [14] 에 의한 게이트 할당을 통해 생성된 결과이다. 일반 네트 리스트를 사용한 GAGEN은 참고문헌 [8] 의 결과에 비해 평균 7.46% 정도의 트랙 최소화를 통한 레이아웃의 개선을 가져왔으며 참고문헌 [13] 의 결과에 대해서도 GAGEN은 평균 6.67% 정도 개선된 결과를 얻었다. 참고문헌 [9] 의 결과가 GAGEN에 비해 평균 6.22% 나은 결과를 보인 것은 참고문헌 [9] 는 논리식에 근거한 설계 양식을 입력을 받아들여 최적화를 수행한 후 레이아웃을 생성하므로 회로에 근거한 네트 리스트를 입력을 받아들이는 GAGEN에 비해 최적화를 수행하는 범위가 상대적으로 크기 때문이다.

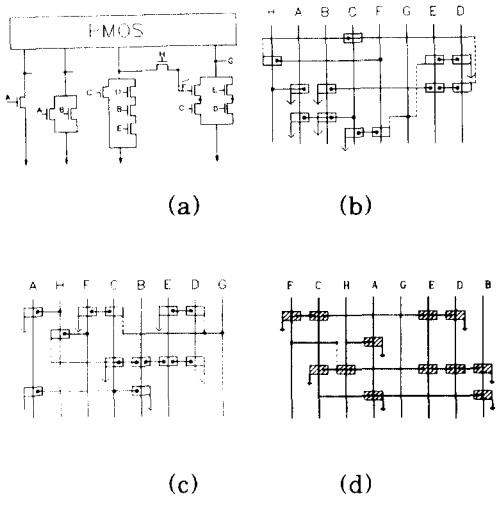


그림 14. 레이아웃 비교
(a) 예제 회로
(b) 참고문헌 [7] 의 시스템에 의한 결과
(c) 참고문헌 [8] 의 시스템에 의한 결과
(d) GAGEN의 결과

Fig. 14. Layout comparison.
(a) an example circuit,
(b) result by the system in [7] ,
(c) result by the system in [8] ,
(d) result of GAGEN.

표 1. GAGEN과 타 시스템의 결과 비교

Table 1. Result comparison between GAGEN and other systems.

입력 회로	트랜지스터 수	게이트 수	각 시스템이 생성한 레이아웃의 트랙 수 [8]	[9]	[13]	GAGEN
exp 1	12	8	4	-	4	4
exp 2	32	12	9.67(avg)	7.65(avg)	-	9
exp 3	19	8	8.31(avg)	7.23(avg)	-	7
exp 4	12	7	7	-	-	7
exp 5	12	7	-	4	-	4
exp 6	18	13	-	-	5	5
Full adder	14	7	-	-	5	4
CLA unit	46	23	14.63 (avg)	12.09 (avg)	-	11
4-bit ALU	144	53	34.90 (avg)	26.16 (avg)	-	37

참고문헌 [8] 과 [9] 의 시스템에 도입된 min-cut 알고리듬은 초기 분할에 따라 결과가 달라질 수 있으므로 신뢰할 만한 결과를 얻기위해 여러번의 초기분할 과정이 필요하다.[9] GAGEN은 간격 그래프에 의한 초기 게이트 할당과 초기 레이아웃으로 부터

얻은 정보를 통한 게이트 재할당을 수행하므로 항상 일정한 결과를 나타낸다.

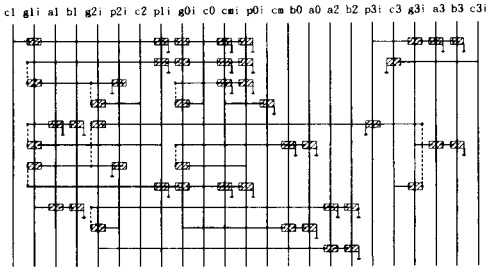


그림 15. GAGEN이 생성한 CLA 유니트 레이아웃
Fig. 15. CLA unit layout generated by GAGEN.

Ⅵ. 결론

본 논문에서는 다단 논리 구현을 위한 레이아웃 생성 시스템으로 gate matrix 형태의 레이아웃을 생성하는 GAGEN 시스템에 대하여 기술하였다. GAGEN에서는 네트 바인딩의 지연을 위한 다이나믹 네트 리스트가 아닌 일반 네트 리스트를 입력으로 받아들여 직렬 네트에 대한 바인딩 문제를 내부적으로 flexible 네트 바인딩에 의해 해결하고, 레이아웃 면적에 결정적인 영향을 주는 게이트 할당 문제를 초기 게이트 할당과 게이트 재할당으로 구분된 과정을 수행하여 보다 개선된 게이트 할당이 이루어지도록 하였다. 또한 트랙 최소화 과정 수행을 위한 left-edge 알고리듬 적용시 동일 열을 차지하면서 상호 충돌을 일으키지 않는 두 네트에 대해 position 필드 값을 주어 보다 세분화된 정보로 left-edge에 의한 트랙 최소화 과정을 수행하였다. Gate matrix 레이아웃에서 게이트 입력의 수를 g라하면 GAGEN 시스템의 초기 게이트 할당 알고리듬과 게이트 재할당 알고리듬의 time complexity는 $O(g^2)$이다. 네트의 수를 n이라하면 flexible 네트 바인딩 알고리듬과 네트 병합 알고리듬, 트랙 재배치 알고리듬의 time complexity는 각각 $O(n)$, $O(n^2)$, $O(n^2)$이다. 따라서 전체 time complexity는 $O(g^2 + n^2)$이다.

GAGEN은 다이나믹 네트 리스트로 표현된 회로에 근거한 입력을 받아들이는 참고문헌 [8] 의 결과보다 평균 7.46%의 면적상의 개선을 이루었다. 본 논문과 입력 형태가 다른 참고문헌 [9] 의 시스템은 최적화 범위가 큰 논리식 형태를 입력으로 하였으므로 입력의 크기가 증가되면 look ahead 측면에서 네트 리스트를 입력으로하는 GAGEN 보다 유리하여 보다 개선된 결과를 나타낼 수 있다. 앞으로 사용자

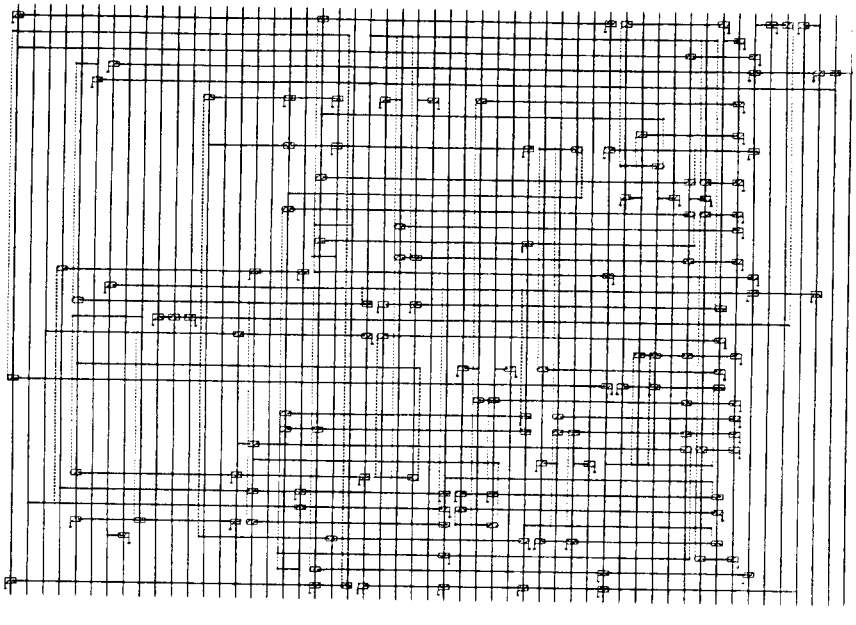


그림 16. GAGEN이 생성한 4-bit ALU 레이아웃
Fig. 16. 4-bit ALU layout generated by GAGEN.

가 요구하는 핀의 위치 고려, aspect ratio에 대한 지원등이 GAGEN 시스템의 과제로 남아있다.

參 考 文 獻

[1] D. D. Gajski, Silicon Compilation, Addison-Wesley Pub. : Reading, Mass., 1988.

[2] A. D. Lopez and H.-F. S. Law, "A Dense Gate Matrix Layout Method for MOS VLSI," *IEEE Trans.* Electron Devices, vol. ED-27, Aug. 1980, pp. 1671-1675.

[3] A. Weingerber, "Large Scale Integration of MOS Complex Logic: A Layout Method," *IEEE Journal of Solid-State Circuits*, vol. SC-2, no. 4, Dec. 1967, pp. 182-190.

[4] S. M. Kang, "Metal-Metal Matrix for High-Speed MOS VLSI Layout," *IEEE Trans. CAD*, vol. CAD-6, no. 5, Sept 1987, pp. 886-891.

[5] T. Kashiwabara and T. Fujisawa, "NP-Completeness of the Problem of Finding a Minimum-Clique-Number Interval Graph Containing a Given Graph as a Subgraph," *in Proc. Int. Symp. Circuits Syst.*, May 1979, pp. 657-659.

[6] O. Wing, "Interval-graph-based Circuit Layout," *in Proc.* ICCAD, Sept. 1983, pp. 84-85.

[7] O. Wing, S. Huang and R. Wang, "Gate Matrix Layout," *IEEE Trans. CAD*, vol. CAD-4, no. 3, July 1985, pp. 220-231.

[8] D. K. Hwang, W. K. Fuchs and S. M. Kang, "An Efficient Approach to Gate Matrix Layout," *IEEE Trans. CAD*, vol. CAD-6, no. 5, Sept. 1987, pp. 802-809.

[9] U. Singh and C. Y. Roger Chen, "From Logic to Symbolic Layout for Gate Matrix," *IEEE Trans. CAD*, vol. 11, no. 2, Feb. 1992, pp. 216-227.

[10] T. Ohtsuki, H. Mori, T. Kashiwabara and T. Fujisawa, "On Minimal Augmentation of a Graph to Obtain an Interval Graph," *J. Computer Syst. Sci.*, Feb. 1981, pp. 60-97.

[11] T. Ohtsuki, H. Mori, E. S. Kuh, T. Kashiwabara and T. Fujisawa, "One-Dimensional Logic Gate Assignment and Interval Graphs," *IEEE Trans. CAD*, vol. CAS-26, no. 9, Sep. 1979, pp. 675-684.

[12] W. Shu, M.-Y. Wu and S. M. Kang, "Improved Net Merging Method for Gate Matrix Layout," *IEEE Trans. CAD*, vol. 7, no. 9, Sept. 1988, pp. 379-395.

[13] "Gate Matrix Layout에 관한 연구," 최종 연구 보고서: CAD Software의 개발에 관한 연구, 과학기술처, 1987년 7월, pp. 225-246.

[14] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," Bell Syst. Tech. J., vol. 49, Feb. 1970, pp. 291-307.

[15] C-J. Tseng and D. P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," *IEEE Trans. CAD*, vol. CAD-5, no. 3, July 1986, pp. 379-395.

著 者 紹 介

金 相 範(準會員)
1990년 8월 서강대학교 전자공학과 졸업. 1992년 8월 서강대학교 전자공학과 공학석사 취득. (CAD & Computer Systems 전공). 1992년 8월 ~ 현재 삼성전자 반도체부문 DSP팀 근무. 주관심분야는 DSP Processor Architecture. VLSI 설계 등임.

黃 善 泳(正會員) 第 30 卷 A 編 第 2 號 參照
현재 서강대학교 전자공학과 교수.