論文 89-26-4-12

Poly-jog을 사용한 그리디 스위치박스 배선기

(A Greedy Poly-jog Switch-Box Router (AGREE))

李 哲 東*, 鄭 正 和**

(Chul Dong Lee and Jong Wha Chong)

要 約

본 논문에서 제안하는 switch-box 배선기는 greedy poly-jog 배선기와 via 최소화기로 나누어진다. Greedy poly-jog 배선기는 Luk의 greedy swich-box 배선 알고리듬을 기본으로 하며, 수평track에 metal을 수직track에 poly-silicon을 배선하는 제한을 완화하여 필요한 경우에는 수평track에 poly-silicon을 배선함으로써 배선영역의 수평track을 증가시키지 않고 배선할 수 있다. Via 최소화기는 배선된 wire의 각 corner를 펴거나 wire 선분을 평행이동하거나 metal을 poly-silicon 및 poly-silicon을 metal로 바꿈으로써 via와 배선길이를 줄이는 과정을 수행한다.

본 배선기는 column 방향으로 배선영역을 scan함으로써 배선을 완료하며, 시간복잡도는 O(M(N+Nnet))이다. 여기서, M, N, Nnet은 각각 배선 column의 수, 배선 row의 수, net의 수이다.

Abstract

This paper proposes an efficient switch-box router which consists of two parts; greedy poly-jog router and via minimizer. The greedy poly-jog router, which is extended from the greedy switch-box router of Luk, routes not only metal wires at horizontal tracks and poly-silicon wires at vertical tracks but also poly-siliocon wires at horizontal tracks if necessary. The via minimizer reduces the number of vias and the wire length by flipping of each corner, parallel moving of wire segments, transformation metal into poly-silicon, and transformation poly-silicon into metal.

The result is generated through the column-wise scan across the routing region. The expected time complexity is O(M(N+Nnet)). Where M, N, and Nnet are respectively the number of columns, rows, and nets in the routing region.

I. 서 론

반도체 산업이 급속도로 발전함에 따라, 사용자들의 요구에 의한 주문형 IC가 널리 보급되고 있다. 이에 따라 주문형 IC를 설계할 수 있는 tool들이 많이 연구되고 있는데, 이들은 완전주문형 IC설계 tool, 반주문형 IC설계 tool 및 silicon compiler로 나눌 수 있다. 완전주문형 IC설계방식은 IC도면편집기를 이용하여 IC도면을 생성하고 검증하는 방식이다. 반주문형 IC설계방식은 미리 설계하여 보관하고 있는 cell을 이용하여 배치 및 배선을 수행함으로써 IC도면을 생성하는 방식이며, cell의 형태에 따라 gate array

*正會員, 韓國電子通信研究所
(Electronics and Telecommunications Research Institute)
**正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1988年 12月 22日

방식, standard cell방식 및 macro-cell 방식으로 나눌 수 있다. Silicon compiler는 high level language로 서술한 기능을 논리회로 생성 및 최적화를 수행하고 IC도면을 생성한다. 이때 IC도면 생성과정은 standard cell방식 및 macro-cell방식의 배치, 배선을 많이 이용한다.

반주문형 IC설계 tool 및 silicon compiler의 IC도면생성기에서 배치 및 배선은 중요한 문제이다. 서로 연결되어야 될 모듈들과 연결정보(net-list)를 입력으로 하여 모듈들을 배치하고[1], 모듈들 사이의 배선영역을 정의한 후 각 net에 대하여 대략적인 배선경로를 할당하고 각 배선영역에서 세부적인 배선을 수행한다.

세부적인 배선은 channel 배선기[2~4], switch-box 배선기[5~10] 그리고 river 배선기[11~12]들을 이용하여 수행할 수 있다. Channel 배선의 배선영역은 직사각형으로써 상하변의 net들은 고정된 위치에 존재하고, 좌우변의 net들은 각 변내의 임의의 위치에 존재할 수 있다. Switch-box 배선의 배선영역은 직사각형이고, 4변의 고정된 위치에 net들이 존재하며, 같은 net들을 배선영역내에서 배선하는 것이다. River 배선은 직각변으로 이루어진 영역에서 한 layer만 이용하여 배선하며 한 net의 wire가 다른 net의 wire와 교차하는 것을 허용하지 않는다. Channel 배선기와 switch-box 배선기가 주로 신호선을 배선하는데 비해 river 배선기는 power 및 ground선을 배선하는데 이용된다.

Switch-box 배선기는 여러 사람에 의해서 연구되어 왔는데[5~10], 수평track과 수직track을 서로 다른 layer로 구분하여 배선하는 방법들[5~8]과 수평track과 수직track에 부분적으로 서로 다른 layer를 허용하여 배선하는 방법들[9~10]이 있다. 후자의 경우가 전자보다 결과가 좋으나 복잡한 경험적 방법을 사용하기 때문에 더 많은 계산시간이 걸린다. 그 중 Weaver[9]는 배선전문가 시스템으로, 선분을 배선할 때마다 이미 저장한 많은 배선규칙을 조사하기 때문에 많은 계산시간이 소요된다. Mighty[10]는 한번 배선을 한 후에 weak modification과 strong modification 과정을 거치며, strong modification에서는 이미 배선된 선분들을 경험적으로 재배선시키게 되므로 많은 계산시간이 소요된다. 본 배선기도 수평track과 수직track에 부분적으로 다른 layer를 허용하지만, greedy switch-box 알고리듬[5]에 기본을 두고 있기 때문에 Weaver나 Mighty보다 계산시간이 빠르며, 많은 switch-box 배선을 필요로 하는 macro-cell 배선문제 등에서 더 실제적으로 이용되리라 생각된다.

본 논문에서 제안하는 switch-box 배선기는 greedy ploy-jog 배선기와 via 최소화기로 구성되어 있으며, 입력은 net-list이고 출력은 CIF로서 FULED (FUll custom Layout EDitor)[13]로 나타낼 수 있다. Greedy poly-jog 배선기는 poly-jog 개념을 도입하여 W. K. Luk이 제안한 greedy switch-box 배선 알고리듬[5]을 확장했다. 그리고, via 최소화기는 이미 배선된 wire를 평행 이동시키거나 corner펴기 및 metal을 poly-silicon 또는 poly-silicon을 metal로 바꾸는 과정을 수행함으로써, via의 수와 전체 배선길이를 줄인다.

Ⅱ. Greedy poly-jog 배선

배선문제는 NP-complete 문제로 판명되어[14] 그 최적해를 구할 수 없기 때문에, 최적해에 근접할 수 있는 해는 보통 휴리스틱 방법에 의하여 구하고 있다. 연결될 net들의 위치가 4변 모두 고정된 switch-box 배선문제는, 상. 하변 net들의 위치만 고정된 channel 배선문제보다 해를 구하기가 어렵다. 그러므로 switch-box 배선기들은 channel 배선기보다 그 해법이 복잡할 뿐만아니라, 계산시간 또한 많이 걸린다.

Greedy channel 배선기는[2] 왼쪽변 net들을 최좌측 column의 각 수평track에 할당한 후에 최우측 column까지 우측으로 한 column씩 진행하면서 각 column에서 join 및 jog를 수행한다. Join 및 jog는 서로 다른 수평track 또는 진행중인 column의 상. 하변 net와 수평tarck중에서 같은 net가 있으면 이를 연결하거나, 연결할 수 없을 경우에 서로 가까운 track으로 옮김으로써 다음 column 이후에 연결을 쉽도록 한다.

이 greedy channel 배선기는 알고리듬이 간단하고, 좌측으로부터 우측까지 한번 scan함으로써 종료되므로 시간복잡도가 (M (N+Nnet))인 매우 빠른 채널배선기로 알려져 있다. Luk의 greedy switch-box 배선기는[5] greedy channel 배선 알고리듬을 switch-box 문제에 적용되도록 확장한 것으로, 우측으로 진행하면서 각 column에서 join 및 jog를 할때 우측변에 연결될 net들이 고정된 위치에 있다는 점을 더 고려한다. 이는 greedy channel 배선기와 마찬가지로 다른 switch-box 배선기에 비하여 매우 빠르다.

본 greedy poly-jog 배선기는 switch-box 배선을 하기위한 것으로 poly-jog를 사용하여 Luk의 greedy switch-box 배선 알고리듬을 확장한 것이다. Greedy switch-box 배선기는 언제나 수평track에 metal을 수직track에 poly-silicon을 사용하여 배선하기 때문에 주어진 배선영역보다 수평track을 증가시켜야 배선을 완성하는 경우가 빈번하게 발생한다. 특히 Luk의 greedy switch-box 배선기의 경우, Ⅵ장에서 설

명하는 3가지 예제중에서 Burstein의 difficult switch-box와 Luk의 Dense switch-box를 수평track을 1개씩 증가시키면서 배선하였다. 그러나, greedy poly-jog 배선기는 수평track이 모두 metal로 점유되었고 또 다른 net가 수평track을 요구할때 이 net를 일시적으로 poly-silicon으로 배선함으로써 수평track을 증가시키지 않고도 배선할 수 있다.

1. 전제 및 용어

- 배선영역

배선영역은 직각사각형이며, 4변에 일정한 간격으로 terminal이 위치하고 있는데, 이곳에 net들이 연결된다. 4변의 각 terminal에 연결될 net들의 위치는 미리 정해지며, 각 terminal에 연결될 net가 있으면 그 net번호가 표시되고, 없으면 0으로 표시된다. Greedy poly-jog 배선 알고리듬을 전개하기 위하여 배선영역을 $R=\{0, 1, \cdots, M, M+1\} \times \{0, 1, \cdots, N, N+1\}$로 정의한다. 여기서, M 및 N은 각각 배선영역안의 column의 수 및 row의 수이다.

- TOP (i), BOTTOM (i)

i번째 column의 윗변 terminal 및 아랫변 terminal에 할당된 net이다.

- nearest possible track

TOP (i) (BOTTOM (i))와 같은 net가 할당된 수평track이나 비어있는 수평track중에서 윗변(아랫변)에 가장 가까운 수평track으로써, TOP (i) (BOTTOM (i))에서 그 수평track까지 poly-jog가 존재하지 않는 track을 말한다.

- split net

Column i에서 동일 net가 두개 이상의 수평track을 점유하고 있거나, 점유하려고 할때 이를 split net라 한다.

- poly-jog, poly-jog net

Poly-jog는 metal로 배선중인 수평track에 poly-silicon으로 겹쳐서 배선하는 것을 말한다. Column i에서 수평track에 poly-silicon으로 배선되기 시작하는 net이거나 배선이 진행중인 net를 poly-jog net라고 하며, column i에서 poly-silicon 수평track 배선이 종료되는 net는 poly-jog net가 아니다.

2. 배선 알고리듬

Greedy poly-jog 배선 알고리듬을 그림 1에 나타내었으며, 각 단계별로 상술한다.

(Step 1)

왼쪽변에 위치한 net들을 수평track을 통하여 column 1로 진행한다.

```
Bring nets of left terminals into column 1          (step 1)
for i=1 to M {
  Bring TOP(i), BOTTOM(i) into horizontal tracks    (step 2)
  if (step 2 failed) {
      N=N+1;
      go to step 1;
    }
  Join split nets                                   (step 3)
  if (step 2 left poly-jog net) {
      Poly-jog                                      (step 4)
      if (poly-jog failed) {
          N=N+1;
          go to step 1;
        }
  }
  for (net_with_no_right_terminals)                 (step 5)
      Jog
  for (net_with_right_terminals)                    (step 6)
      SW-jog
  if (i>=M-fanout parameter) {
      Fanout for multiterminal nets                 (step 7)
      Poly-jog for right terminal nets              (step 8)
    }
} /*for loop*/
while (split-net exist) {                           (step 9)
      M=M+1;
      Join split nets at column M;
  }
```

그림 1. Greedy poly-jog 배선 알고리듬
Fig. 1. Greedy poly-jog routing algorithm.

(Step 2)

Column i에서 TOP (i), BOTTOM (i)가 모두 nearest possible track이 존재하면 그 track까지 수직track을 통하여 poly-silicon으로 배선한다(그림2(a)). 그러나 TOP (i), BOTTOM (i)가 모두 nearest possible track이 존재하여도 nearest possible track까지 연결하는 수직 배선 선분이 서로 겹치게 되면 더 짧은 수직 배선 선분이 필요한 것을 배선하고 나머지 것은 Step 4에서 poly-jog를 하도록 한다. Column i에서 poly-jog net가 진행중이고 TOP (i) = 0 (BOTTOM (i) = 0)일때 그 poly-jog net를 TOP (i) (BOTTOM (i))처럼 취급한다(그림2(b)). TOP (i)와 BOTTOM (i)중에서 하나만 nearest possible track이 존재하거나 TOP (i)와 BOTTOM (i)의 nearest possible track이 같을때, 둘중에 하나를 Step 4에서 poly-jog 하도록 한다. TOP (i)와 BOTTOM (i)가 둘다 nearest possible track이 존재하지 않으면 둘다 poly-jog를 해야되는 상황이 되므로, 이는 Step 4의 column i에서 poly-jog net는 한개로 제한하는 법칙을 위반하므로 Step 2를 실패로 한다. 또한 column i에 이미 poly-jog net가 진행중이고 적어도 TOP (i)나 BOTTOM (i)중에서 하나가 nearest possible track이 존재하지 않으면 Step 2를 실패로 한다.

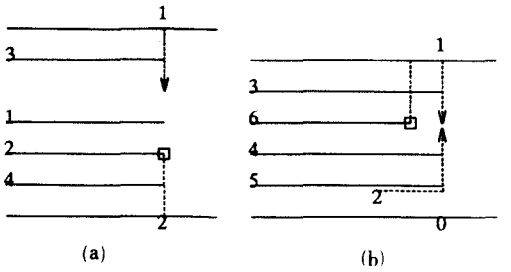


그림 2. Nearest possible track까지 연결
Fig. 2. Connect to the nearest possible track.

(Step 3)

Column i에서 빈 수평track을 늘리기 위하여 split net들을 연결하는 과정으로, 연결될 수 있는 split net가 2개이상 있을때 다음과 같은 우선순위로 수행한다.

1) Column i에서 split net인 poly-jog net가 존재할때 이를 우선하여 join한다(그림3(a)).
2) Split net가 3개이상 존재할때 더 많은 split net를 join할 수 있으면 이를 수행한다.
3) 더 많은 수평track을 점유하고 있는 split net를 join한다(그림3(b)).
4) Center track에 더 가까운 수평track을 점유하고 있는 split net를 join한다.
5) Split net중 가장 위의 수평track과 가장 아래 수평track의 거리가 더 큰 net를 join한다.

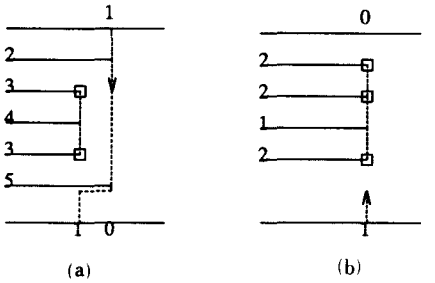


그림 3. Split net의 join
Fig. 3. Join split net.

(Step 4)

Column i에서 이미 모든 수평track이 metal로 점유되어서 TOP(i) 또는 BOTTOM(i)가 배선될 수 있는 빈 수평track이 없을때 이를 poly-jog한다. Column i에서 poly-jog net는 2개이상 발생할 수 없으며 2개이상 발생해야 할 경우 poly-jog를 실패로 하고 배선영역의 row의 수를 하나 증가시킨후 Step 1로 되돌아 간다. Column i에서 poly-jog net를 1개로 제한하는 것은 알고리듬에 첨가되는 경험적 고찰(heuristics)이 복잡해지는 것을 방지하기 위함이며, 또한 VI장의 예에서 보는바와 같이 poly-jog net를 1개로 제한하여도 주어진 배선영역안에서 해를 구할 수 있음을 알 수 있다. TOP(i) 또는 BOTTOM(i)가 poly-jog로 배선되어야 하는가는 Step 2에서 알려주며, 다음은 TOP(i) 또는 BOTTOM(i)가 poly-jog net일 경우에 poly-jog를 수행하는 것에 대하여 설명한다.

1) TOP(i)에 대한 poly-jog의 수행
Column i에서 join이 이루어진 가장 아래에 있는 수평track의 바로 아래 수평track으로TOP(i)를 poly-jog한다(그림4(a)).
2) BOTTOM(i)에 대한 poly-jog의 수행
Column i에서 join이 이루어진 가장 위에 있는 수평track의 바로 위 수평track으로 BOTTOM(i)를 poly-jog한다(그림4(b)).

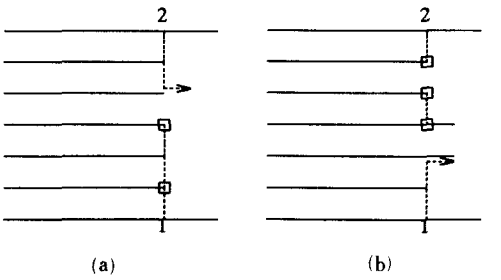


그림 4. Poly-jog
Fig. 4. Poly-jog.

(Step 5)

Column i에서 수평track을 통하여 배선될 net들에 대하여 column k(k>i)에서 join이 쉽도록 하기 위하여 미리 수평track를 바꾸어 진행(jog)하도록 하는 과정이다. 2개이상의 net가 jog할 수 있을때 다음과 같은 우선순위에 의하여 jog할 net를 결정한다. 단 switch-box 우측변의 net는 Step 6에서 고려하고, 여기서는 고려하지 않는다.

1) Poly-jog net
Poly-jog net를 jog의 가장 우선 대상으로 한다.

2) Split net
Column k(k>i)에서 split net의 join을 쉽게 하기 위하여 jog를 수행한다. Jog length가 큰 것을 우선하며, jog length가 minimum-jog-length보다 클때만 jog를 수행한다. Minimum-jog-length는 user에 의하여 주어지며, 보통 2-4의 값을 갖는다.

3) JOG_{TIB} : TOP(k) (BOTTOM(k)) (k>i)를 고려한 jog
TOP(k) (BOTTOM(k)) (k>i)와의 연결을 쉽게 하기 위하여 TOP(k) (BOTTOM(k))가 있는 net에 대하여 이를 고려하여 그림5(a), (b)와 같이 jog를 수행한다.

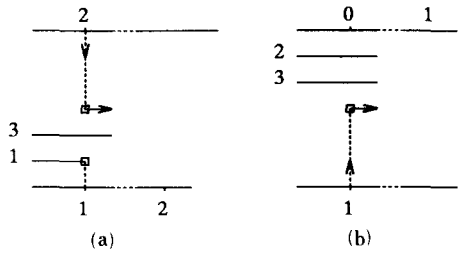


그림 5. TOP(k) (BOTTOM(k)) (k>i)를 고려한 Jog
Fig. 5. Jog considering TOP(k) (BOTTOM(k)) (k>i).

(Step 6)

SW-jog는 column i에서 수평track을 통하여 배선될 net들 중에서 switch-box 우측변에 연결될 net들에 대하여 column k(k>i)에서 join이 쉽도록 하거나 우측변에 연결을 쉽도록 하기 위하여 미리 수평track을 바꾸어 진행하도록 하는 것이다. 2개이상의 net가 SW-jog할 수 있을때 다음과 같은 우선순위에 의하여 jog할 net를 결정한다.

1) Poly-jog net
Step 5에서와 같이 poly-jog net를 SW-jog의 가장 우선 대상으로 한다.

2) Right terminal track으로 jog
어떤 net의 right terminal track이 column i에서 비어 있고, 그 track까지 jog를 할 수 있으면 jog를 한다. Jog length가 큰것을 우선jog하며, jog length가 minimum-jog-length보다 클때만 수행한다(그림6(a)).

3) Split net
Step 5에서와 같이 column k((k>i)에서 split net가 쉽게 join 되도록 jog를 수행한다. Jog length가 큰것을 우선하며, jog length가 minimum-jog-length보다 클때만 수행한다.

4) JOG_{SB} : TOP(k) (BOTTOM(k))와 우측변 terminal을 고려한 jog
JOG_{SB}의 대상이 되는 net의 최우측 상.하변 terminal이 위치한 column을 M+1로 나눈 값이 P값보다 크고 작음에 따라 다음과 같이 수행된다.

(a) 어떤 net의 최우측 상변 또는 하변의 terminal이 p-portion의 우측에 존재할때 : Right terminal track에 근접한 track까지 jog를 수행한다. Jog length가 큰 것을 우선 jog하며, jog length가 minimum-jog-length보다 클 때만 jog한다(그림6(b)). 여기서, p는 user에 의해서 주어진 값이며, 보통 0.4-0.6의 값을 갖는다.

(b) 어떤 net의 최우측 상변 또는 하변의 terminal이 p-portion의 좌측에 존재할때 : 최우측 상변 또는 하변 terminal 이전의 column에서는 Step 5의 $JOG_{T/B}$를 수행하고, 그 이후에서는 right terminal track에 근접한 track까지 jog한다. Jog length가 큰 것을 우선 jog하며, jog length가 minimum-jog-length보다 클 때만 jog한다(그림6(c)).

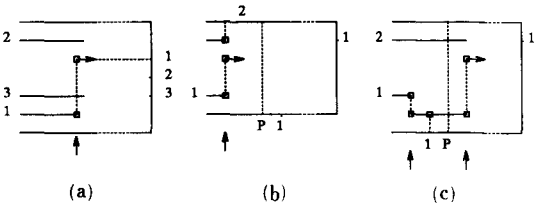


그림 6. SW-jog
Fig. 6. SW-jog.

(Step 7)

Switch-box의 우측변에 있는 multiterminal net들을 연결하기 위하여, 우측변으로부터 fanout parameter 떨어진 곳에서 부터 fanout을 수행한다. 여기서, panout parameter는 사용자에 의해서 주어지며, 보

통 3-5의 값을 갖는다. fanout은 우측변에 있는 multiterminal net들 중에서 그 net의 각 terminal과 일치하는 수평track이 모두 비어있는 net들에 대하여 수행하며, fanout이 가능한 net들 중에서 더 많은 terminal을 갖는 net를 우선 수행한다(그림 7).

(Step 8)

Switch-box 우측변에 있는 net들과 연결을 할 수 있도록 하기 위하여, 우측변으로 부터 fanout parameter 떨어진 곳에서 부터는 수평track에 metal로 진행되는 net가 column i에서 다른 수평track으로 옮겨서 poly-silicon으로 배선 가능하다. 이는 column i가 우측변에 가까워질수록 다른 수평track 위치에 있는 우측변 terminal과 동일 net를 갖는 net의 배선이 배선영역내에서 힘들어지므로 수평track에서 다른 수평track으로의 poly-jog를 허용한다(그림 8).

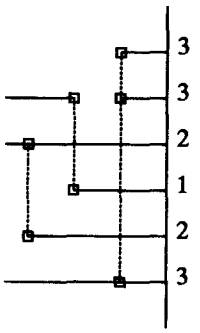


그림 7. Fanout 예
Fig. 7. Example of the fanout.

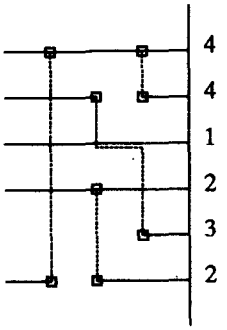


그림 8. 우측변 net의 poly-jog
Fig. 8. Poly-jog for the right side net.

(Step 9)

Column M 이후에 split net가 남았을때 column을 한개 증가시킨후 가능한 많은 split net들을 join하며, 그래도 split net가 남으면 split net가 존재하지 않을 때까지 column을 하나씩 증가시켜서 split net를 join한다(그림 9).

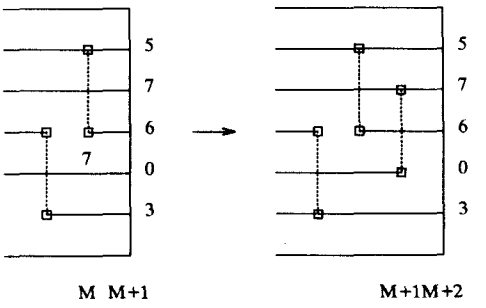


그림 9. Column의 증가
Fig. 9. Increase column.

Ⅲ. Via 최소화

Greedy poly-jog 배선 결과는 greedy 알고리듬을 기본으로 하고 있기 때문에 불필요한 구부림이 많이 생긴다[5]. Via 최소화기에서는 그런 구부림을 펴거나 wire의 선분을 평행 이동하거나 wire 선분의 layer를 metal에서 poly-silicon 또는 poly-silicon에서 metal로 바꿈으로써 via의 수를 줄이고, 경우에 따라서는 배선길이도 줄인다. Via 최소화 과정은 각 net에 대하여 corner펴기, wire이동, poly-silicon을 metal로 변환, metal을 poly-silicon으로 변환의 순서로 수행된다.

1. Corner펴기(corner flipping)

자유 corner는 어떤 net의 배선된 wire의 수직 선분과 수평선분의 양 끝이 만나서 이루는 점을 나타내고, 고정 corner는 어떤 net의 배선된 wire의 수직선분과 수평선분이 만나서 이루는 점중에서 자유 corner가 아닌 점을 말한다.

Corner펴기는 자유 corner에 한하여 실시한다(그림10). 그러나 자유 corner와 인접한 양쪽 선분과 연결된 두 corner가 모두 고정 corner일 때는 그 자유 corner를 펴도 효과가 없거나(그림11(a)), 그림11(b)와 같이 효과가 있어도 뒤에 설명하는 wire 이동으로 같은 효과를 얻을 수 있으므로 펴지 않는다. 또한,

자유 corner와 인접한 어느 한쪽 선분이라도 terminal과 직접 연결 되었으면 그 corner를 펴지 않는다. 그리고, 고정 corner의 펴기는 배선길이를 증가시키므로 펴지 않는다.

Corner펴기의 수행과정은 각 net에 대하여 최좌측 자유corner로 부터 우측으로 진행하면서 최우측의 자유 corner까지 corner펴기를 시도한다.

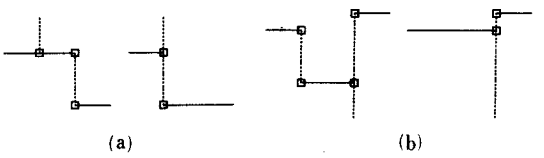


그림10. 자유 corner펴기의 예
Fig. 10. Example of the free corner flipping.

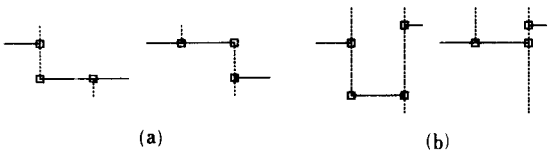


그림11. 인접한 corner가 고정 corner일 때의 자유 corner펴기의 예
Fig. 11. Example of the free corner flipping connected to fixed corners in both sides.

2. Wire 이동(wire move)

Corner펴기가 허용되지 않은 corner를 포함하는 ㄴㄱㄷㄱ형태 중에서 수평 또는 수직선분을 평행 이동할 때 이미 배선된 다른 net의 수평, 수직선분들과 설계규칙을 위반하지 않고, via나 배선길이를 줄일 경우에 그 선분을 평행 이동한다. 이동거리는 이동하는 선분이 처음 corner를 만날 때 까지이다. 이동할 wire 선분의 양끝에 있는 두 corner가 모두 고정 corner일 때는 그 wire 선분이 다른 wire 선분과 교차하지 않아야 된다(그림12(a)). 그리고, 이동할 wire 선분의 한 corner가 두 고정 corner와 인접한 자유 corner일 때는 이동할 wire 선분의 한쪽끝이 두 wire 선분의 교차점 일수도 있다(그림12(b)).

3. Poly-silicon을 metal로 변환(poly-to-metal)

Metal 선분과 교차하지 않는 수직 poly-silicon 선분중에서 그 선분의 어느 한쪽 끝이라도 다른 poly-silicon 선분과 연결되어 있지 않으면 그 수직 poly-silicon 선분을 metal로 바꾸고 다른 poly-silicon 선분과 연결되지 않은 끝에 있는 via를 제거한다(그림13).

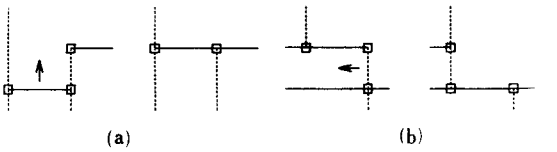


그림12. Wire 이동
Fig. 12. Wire move.

4. Metal을 poly-silicon으로 변환(metal-to-poly)

Poly-silicon 선분과 교차하지 않는 수평metal 선분중에서 그 선분의 어느 한쪽 끝이라도 다른 metal 선분과 연결되어 있지 않으면 그 수평 metal 선분을 poly-silicon으로 바꾸고 다른 metal 선분과 연결되어 있지 않은 끝에 있는 via를 제거한다(그림14). 단, poly-jog에 의하여 수평 poly-silicon 선분과 겹치는 수평 metal 선분에 대해서는 수행하지 않는다.

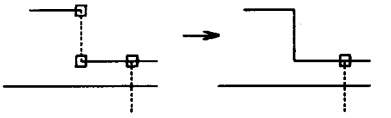


그림13. Poly-to-metal
Fig. 13. Poly-to-metal.

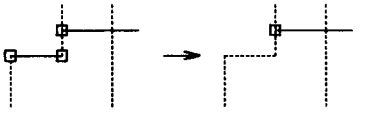


그림14. Metal-to-poly
Fig. 14. Metal-to-poly.

Ⅵ. 실험결과

본 switch-box 배선기는 UNIX OS하의 SUN workstation에서 C로 실현되었으며, 출력 데이타는 CIF 형태이며 ETRI에서 개발한 full custom layout editor인 FULED[13]와 연결하였다.

AGREE의 data는 배선영역의 크기와 배선영역의 가장자리에 위치한 net들의 번호에 관한 정보를 갖고 있다. 그림15(a)는 입력 file의 형태를 보여주는데, 처음 세줄은 각각 net의 수, 배선영역의 column 수 및 row수를 표시한다. 그리고, top-list 및 bot-list는 각각 배선영역 윗변 및 아랫변의terminal에 위치한 net 번호를 최좌측부터 최우측 column까지 나타낸다. 그리고, lef-list 및 rig-list는 각각 왼쪽변 및 오른쪽변의 terminal에 위치한 net 번호를 최하부터 최상의 row까지 나타낸다. 그림15(b)는 Burstein의 difficult switch-box(그림16참조)에 대한AGREE의 입력 file이다.

```
nnets      num1      <--number of nets
ncolumn    num2      <--number of columns
nrow       num3      <--number of rows
top-list             <--top net-list
#1..... #num2
bot-list             <--bottom net-list
#1..... #num2
lef-list             <--left net-list
#1..... #num3
rig-list             <--right net-list
#1..... #num4
```

(a)

```
nnet    24
ncolumn 23
nrow    15
top-list
15 0 2 4 12 7 6 9 5 8 13 15 14 15 0 21 20 1 2 19 1 18 0
bot-list
24 17 16 4 7 6 5 9 8 0 9 12 15 24 15 10 23 1 0 0 22 18 0
lef-list
3 10 4 16 12 17 2 9 1 24 11 13 14 0 0
rig-list
18 22 2 23 18 21 11 20 18 20 0 24 19 3 5
```

(b)

그림15. (a) 입력파일 형태
(b) Burstein의 difficult switch-box의 입력파일

Fig. 15. (a) Input file format.
(b) Input file for the Burstein's difficult switch-box.

그리고, 전술한 바와 같이 본 배선기는 greedy poly-jog 배선을 수행한 후에 via 최소화 과정을 수행하는데, 아래에 설명하는 3가지 switch-box 배선문제에 대한 via 최소화기의 효과를 표 1에 나타내고 있다. 여기서 15%이상의 via수의 감소와 Terminal-intensive의 경우 배선길이도 줄었음을 알 수 있다.

표 1. Via 최소화기의 결과

Table 1. Results of the via minimizer of AGREE.

Swith-box problem	Before via min.		After via min.	
	#vias	wire leng.	#vias	wire leng.
Burstein	57	555	47	555
Terminal-intensive	62	624	51	620
Dense	35	516	29	516

그리고, Burstein의 difficult switch-box[5]에 대하여 본 배선기(AGREE), Hamachi[6], Luk[5], Mod Detour[8], M-Sadowska[7], Weaver[9], Mighty[10]가 수행한 결과를 표 2에 비교하여 나타내었으며, 배선결과는 그림16에 나타내었다. 또한, Luk의 Terminal-intensive switch-box[5] 및 Dense switch-box[8]에 대하여 본 배선기(AGREE) 와 Luk이 배선한 결과를 각각 표 3, 표 4에 비교하여 나타내었으며, 배선 결과는 그림17 및 그림18에 나타내었다.

표 2. Burstein의 difficult switch-box의 배선결과 비교

Table 2. Routing results for Burstein's difficult switch-box.

Router Name	#rows	#vias	wire length
Hamachi (Magic)	15	67	564
Luk (Greedy)	16	59	577
Mod Detour	15	63	567
M-Sadowska	15	58	560
Weaver	15	41	531
Mighty	15	39	541
AGREE	15	47	555

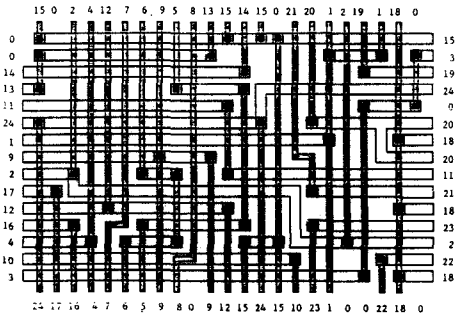


그림16. Burstein의 difficult switch-box 배선 결과

Fig. 16. Routing results for the Burstein's diffiult switch-box.

표 3. Luk의 Terminal-intensive switch-box 배선 결과 비교
Table 3. Routing results for the Luk's Terminal-intensive switch-box.

Router Name	#rows	#vias	wire length
Luk(Greedy)	16	70	629
AGREE	16	51	620

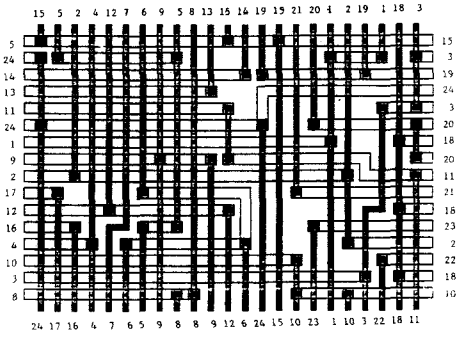


그림17. Luk의 Terminal-intensive switch-box 배선 결과
Fig. 17. Routing results for the Luk's Terminal-intensive switch-box.

표 4. Luk의 Dense switch-box 배선 결과 비교
Table 4. Routing results for the Luk's Dense switch-box.

Router Name	#rows	#vias	wire length
Luk(Greedy)	18	38	530
AGREE	17	29	516

전술한 결과로부터 본 배선기가 greedy switch-box 배선기보다 개선되었음을 알 수 있으며, greedy 알고리듬을 기본으로 하기 때문에 다른 경험적인 방법으로만 배선하는 경우보다 속도가 빠르고 배선된 결과도 우수함을 알 수 있다. 본 배선기중 greedy ploy-jog 배선기는 column 방향으로 한번 scan함으로써 종료되며, 각 row에 진행되는 net들을 고려하기 때문에 O(M(N+Nnet))의 시간복잡도가 걸리며, via 최소화기는 각 net에 대하여 모든 wire 선분을 수행하므로 시간복잡도가 O(NnetM)이다. 그러므로, 본 배선기 전체시간 복잡도는 O(M(N+Nnet))이다.

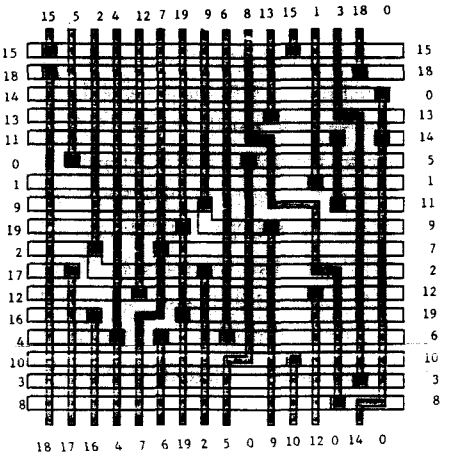


그림18. Luk의 Dense switch-box 배선 결과
Fig. 18. Routing results for the Luk's Dense switch-box.

V. 결 론

본 논문에서는 poly-jog를 도입하여 Luk의 greedy switch-box 배선 알고리듬을 확장한 greedy poly-jog 배선 알고리듬을 제안하고 이를 SUN workstation에 C로 실현하였다. 이는 수평track에 metal을 수직track에 poly-silicon을 배선하는 제한을 완화하여 수평track에 poly-silicon 선분을 metal 선분과 겹쳐 배선할 수 있게 한 것이다.

그리고, greedy poly-jog 배선기로 배선된 결과에서 via수와 배선길이를 줄이기 위하여 via 최소화 알고리듬을 제안하였는데, 배선된 wire의 corner를 펴거나 wire 선분을 평행이동하거나 metal을 poly-silicon 및 poly-silicon을 metal로 바꾸는 과정을 수행한다. 또한, 이를 C로 실현한 결과 15%이상의 via수를 감소시킬 수 있었다.

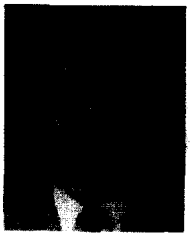
본 switch-box 배선기는 빠른 수행시간과 좋은 결과를 얻을 수 있었다. 실제로 switch-box 배선문제가 많이 발생하는 macro-cell방식의 배선문제에서 유용하게 사용될 수 있을 것이며, 본 배선기를 확장하면 channel 배선문제에서도 좋은 결과를 얻을 수 있으리라 기대된다.

參 考 文 獻

[1] B.T. Preas and P.G. Karger, "Automatic placement, a review of current techniques," *Proc. 23rd Design Automation Conf.*, pp. 622-629, Jun. 1986.

[2] R.L. Rivest and C.M. Fiduccia, "A 'greedy' channel router," *Proc. 19th Design Automation Conf.*, pp. 418-424, Jun. 1982.
[3] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Trans. on CAD*, vol. CAD-2, no. 4, Oct. 1984.
[4] D. Brawn, J. Burns, S. Devadas, H. Ma, K. Mayaram, F. Romeo, and A. Sangiovanni-Vincentelli, "Chameleon: a new multi-layer channel router," *Proc. 23rd Design Automation Conf.*, Jun. 1986.
[5] W.K. Luk, "A greedy switch-box router," INTEGRATION, the VLSI journal 3, pp. 129-149, 1985.
[6] G.T. Hamachi and J.K. Ousterhout, "A switch-box router with obstacle avoidance," *Proc. 21st Design Automation Conf.*, pp. 173-179, Jun. 1984.
[7] M. Marek-Sadowska, "Two-dimensional router for double layer layout," *Proc. 22nd Design Automation Conf.*, Ja pp. 117-123, Jan. 1985.
[8] Y. Hsieh and C. Chang, "A modified detour router," *Proc. Int. Conf. on CAD*, pp. 301-303, Nov. 1985.
[9] R. Joobbani and D. Siewiorek, "WEAVER: a knowledge-based routing expert," *IEEE Design & Test*, pp. 12-23, Feb. 1986.
[10] H. Shin and A. Sangiovanni-Vincentelli, "A detailed router based on incremental routing modifications: Mighty," *IEEE Trans. on CAD*, vol. CAD-6, no. 6, pp. 942-955, Nov. 1987.
[11] C.P. Hsu, "General river routing algorithm," *Proc. 20th Design Automation Conf.*, pp. 578-583, Jun. 1983.
[12] R.Y. Pinter, "River routing: methodology and analysis," the third Caltech Conference on VLSI, Mar. 1983.
[13] 김용주, 강길순, 이철동, 유영욱, "대화형 도면 편집기의 개발," 과학기술처 '87 특정연구과제 발표회 논문집, 7월 1988, pp. 91-94.
[14] A.S. LaPaugh, "Algorithms for integrated circuit layout: an analytic approach," Ph. D thesis, MIT, MIT VLSI memo 80-38, Dec. 1980. *

著 者 紹 介



李 哲 東(正會員)
1952年 2月 12日生. 1977年 2月 경북대학교 전자공학과학사 학위 취득. 1989年 2月 한양대학교 산업대학원 전자공학과 석사학위 취득. 1977年 2月～1984年 1月 한국전자기술연구소 설계개발실 선임연구원. 1984年 2月～1985年 5月 한국전자기술연구소 설계자동화 연구실 실장. 1985年 5月～현재 한국전자통신연구소 자동설계기기 연구실 실장. 주관심분야는 VLSI 설계 및 VLSI CAD 등임.

鄭 正 和 (正會員) 第26卷 第1號 參照
현재 한양대학교 전자공학과 부교수