보다 콤펙트한 채널배선을 위한 전처리 및 후처리 알고리즘

Ⅸ. 보다 콤팩트한 채널배선을 위한 전처리 및 후처리 알고리듬

임 종 석 교수
(서강대학교 전자계산학과)

보다 콤팩트한 채널배선을 위한 전처리 및 후처리 알고리듬

서강대학교 전자계산학과

임 종 석

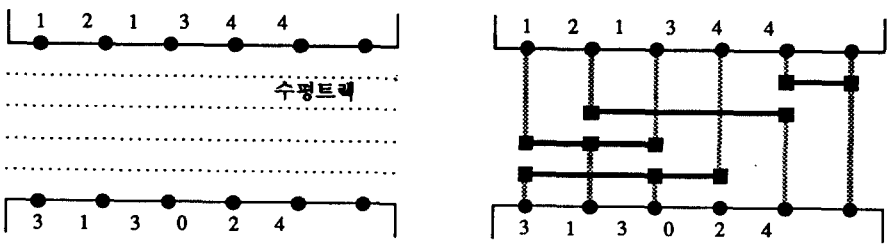
요 약

VLSI 레이아웃에서 채널배선의 중요성은 아주 잘 알려져 있다. 그런데 지금까지 발표된 채널배선을 위한 대부분의 알고리듬들은 배선에 필요한 수평트랙의 수를 최소화하는데 주력하고 있다. 그러나 한편으로는 채널배선의 전처리 또는 후처리 과정으로서 좀더 콤팩트한 배선결과를 얻기위한 기법들에 대한 연구도 아울러 수행되고 있다. 이들은 크게 둘로 나눌 수 있는데 그 한가지는 논리적으로 서로 동등한 터미널들을 서로 교체하여 주어진 채널배선문제의 최대밀도 (maximum density)를 감소 시키는 방법이며 다른 한가지는 배선에 필요한 via (contact hole)의 수를 줄이거나 또는 그들의 위치를 조정하여 트랙간의 간격을 더욱 좁게하는 방법이다. 본 장에서는 이들 두 접근방법 및 그 대표적인 해결방법들을 소개한다.

1. 서 론

두개의 기능모듈이 그림 1(a)와 같이 위 아래로 놓여 있고 이들의 외곽선에 터미날들이 놓여있다고 가정하자. 이들 터미날들은 서로 전기적으로 같은 상태를 유지해야 하는 것끼리 분류되어 있다(네트 리스트). 이때 채널배선문제는 두 모듈 사이의 영역에 사용 가능한 레이어의 선들을 이용하여 전기적으로 서로 같은 터미날끼리 연결하는 것이다. 여기서 요구되는 조건의 하나는 배선에 필요한 수평트랙(그림 1 참조)의 갯수를 최소화하는 것이며, 이는 전체 칩 면적을 가능한 적게하기 위함이다. 배선은 둘 또는 그 이상의 레이어를 사용하나, 여기서는 두개의 레이어를 사용한다고 가정한다. 그림 1(b)에 그림 1(a)의 경우에 대한 배선 결과를 보였다. 그림의 배선에서 수직선의 레이어와 수평선의 레이어가 서로 다른데, 이는 전기적으로 서로 다른 선과의 만남을 피하기 위함이며, 같은 네트의 배선을 위한 서로 다른 레이어의 선들은 두 레이어 사이에 구멍을 뚫어 이를 연결시키는데, 이 구멍을 "via" 또는 "contact hole"이라 한다.

일반적으로 두 레이어 이상에서의 채널배선문제는 NP-hard이며[13], 이에 대한 많은 휴리스틱 알고리즘이 제안되었다[12]. 이 알고리즘들은 채널에 터미날의 위치가 고정되어 있을 때 최소의 수평트랙을 사용한 배선을 얻어내는데 주력하고 있으며, 그 결과 또한 대부분의 벤치마크 예에 대해서 최적의 배선 결



(a) 채널배선문제의 한 예. (b) 배선 결과.

그림 1. 채널배선문제.

과를 얻고 있다. 따라서, 채널배선은 게이트어레이, 스텐다드셀 그리고 매크로셀 등의 배선에 유용하게 사용될 수 있다.

채널배선문제에서 터미날의 위치가 고정되어 있지 않을 경우, 또는 논리적으로 동등한 터미날들이 존재할 경우, 이들을 이용하여 채널배선문제의 최대 밀도를 더욱 감소시킬 수 있다. 이는 위의 채널배선 알고리즘의 전처리과정에 응용할 수 있으며 이에 대한 많은 연구가 이루어 졌다[1,6,8,10,11,14,17]. 또한, 일반적으로 via의 너비와 높이는 수직 또는 수평 배선의 폭 보다 커서 채널에서의 수평 또는 수직트랙 간의 간격을 크게하는 요인이 된다. 따라서, 주어진 채널배선 결과에 대해서 다른 채널배선으로 바꾸거나, via의 수를 줄이고 그들의 위치를 조정하여 더욱 컴팩트한 채널배선을 얻을 수 있다[2,3,4,15,16]. 이러한 방법은 일반 채널배선 알고리즘 결과의 후처리과정으로 유용하게 사용 할 수 있다.

본 장에서는 채널배선에서의 위의 두 기법에 대하여 소개하고 그에 대한 대표적인 해결방법을 기술한다. 서론에 이어 2장에서는 채널문제를 정식으로 정의하고 이에 관련된 기본 용어들을 소개하며, 3장 및 4장에서는 위의 전처리 및 후처리과정을 기술하며 5장에서 결론을 맺는다.

2. 채널배선문제

채널배선문제의 입력은 그림 2에 보인바와 같이 두개의 평행한 수평선과 그 선 위의 터미날 $\{t_1, t_2, \ldots . t_n, b_1, b_2, \ldots . . b_n\}$, 터미날 간의 연결조건인 네트리스트, 그리고 사용가능한 레이어의 갯수 등으로 주어진다. 여기서 터미날 t_i와 b_i, i=1,...,n, 는 수평으로 서로 같은 위치에 있고, 바로 옆 터미날과의 간격은 일정하다고 가정한다. 배선은 두개의 수평선 사이에서 이루어지며 이 영역을 채널이라 한다. 네트리스트는 전기적으로 같은 상태로 연결되어야 할 터미날 그룹들의 집합인데 그림 1(a)의 경우

$\{ n_1 = (t_1, t_3, b_2),\ n_2 = (t_2, b_5),\ n_3 = (t_4, b_1, b_3),\ n = (t_5, t_6, b_6) \}$

로 표시한다. 그러나, 전체 문제를 간단하게 나타내기 위하여 터미날에 그가 속한 네트의 인덱스를 기재하여 그림1(a)와 같이 표시한다.

임의의 터미날 t에서 수직 라인을 그렸을 때, 이 라인과 교차 또는 접하도록 배선할 수 밖에 없는 네트들의 갯수를 행 i 에서의 밀도(density) d_i라 하며, 최대 밀도 $d_{max} = \max\{d_i | 1 \le i \le n\}$라고 정의 한다. 그림 1(a) 의 경우 $d_{max} = 4$이다. 채널배선문제의 한 예가 주어졌을 때 그에 대한 dmax는 배선에 필요한 최소한의 수평트랙 수를 나타낸다.

그림 1(a)에 보인 채널배선문제에서 배선을 위한 수직선과 수평선들의 레이어를 서로 다르게 한다면 그림 1(b)와 같이 네트 1 을 위한 수평선은 네트 3을 위한 수평선보다 위에 놓여져야 한다. 마찬가지로 네트 2에 대한 수평선은 네트 1에 대한 수평선 보다 위에 놓여져야 하는 등 네트간에 배선을 위한 일련의 조건이 필요한데, 이를 수직제한(vertical constraint)이라 한다.

수직제한(vertical constraint)에 의한 다음과 같은 그래프를 구성 할 수 있다. 각 네트당 노드를 만들고, 네트 n_i를 위한 수평트랙 보다 네트 n_j를 위한 수평트랙이 위에 놓여져야 한다면 네트 n_i에 대한 노드 v_i로 부터 네트 n_j에 대한 노드 v_j로 방향성 에지(directed edge)를 만든다. 이렇게 하여 형성한 그래프를 수직제한그래프(vertical constraint graph)라고 하며, 그림 3에 그림 1(a)에 보인 채널배선문제에 대한 수직제한그래프를 보인다. 수직제한그래프에 사이클이 존재하지 않을 경우 수직제한그래프의 최장경로(longest path)를 구할 수 있고, 이에 포함되는 노드의 갯수를 v_{max}라 하면, 이때 각 네트를 하나의 수평선을 사용하여 배선하였을 경우 최소 v_{max}개의 수평트랙이 필요하다.

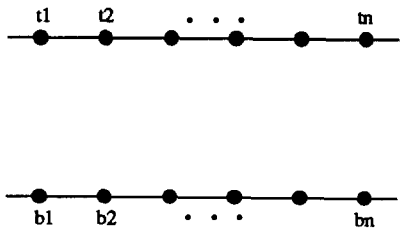


그림 2. 채널배선문제.

그림 3. 그림 1(a)의 예에 대한 수직조건그래프.

수직제한 그래프에 사이클이 존재할 경우도 있다. 그림 4(a)에 보인 채널배선문제의 수직제한그래프를 그림 4(b)에 보였는데, 이는 사이클을 형성하고 있다. 이 사이클이 존재하는 경우는 채널배선에 상당한 장애요인이 되며 그림 4(c)와 같이 도그레그(dog-leg)를 사용하여야만 배선이 가능하다.

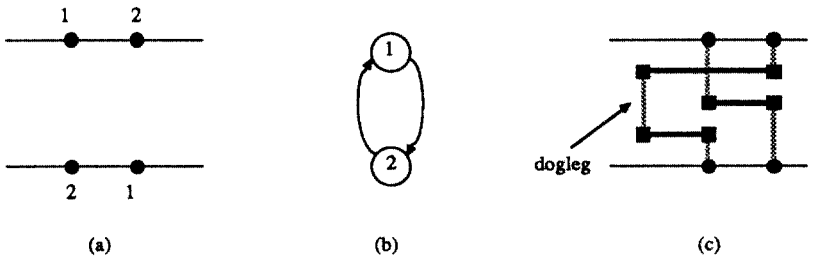


그림 4. 수직 제한 그래프에서의 사이클.

3. 전처리 알고리즘

3.1. 동기 및 문제 정의

그림 5(a)에 보인 논리회로에서 신호선 E 와 F를 그림 5(b)와 같이 바꾸어도 그 논리회로의 기능에는 변함이 없다. 마찬가지로 A와 B또는 C와 D를 바꾸어도 마찬가지이다. 이와 같이 연결선, 또는 신호명을 바꾸어도 그 기능에는 변함이 없는 터미날들을 "논리적으로 동등한" 터미날이라 한다.

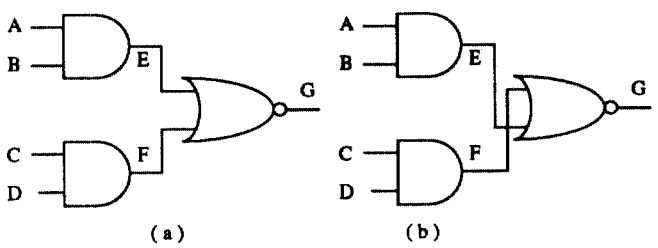


그림 5. 논리적으로 동등한 터미날에 대한 설명.

논리적으로 동등한 터미날들을 이용하여 채널배선문제의 d_{max}, v_{max}등을 더욱 감소시킬 수 있으며, 또한 수직제한그래프에서 사이클이 존재할 경우 이를 제거할 수도 있다. 일반적으로 게이트어레이나 스텐다드셀 레이아웃에서는 NAND, NOR등 기초셀 라이브러리를 구성하고 필요에 따라 이들을 그림 6 에 보인 바와 같이 수평열에 배치하고 배선한다. 따라서 이에 따른 채널배선문제의 터미날 중에는 많은 갯수의 논리적으로 동등한 터미날 집합들이 존재한다.

그림 1(a)의 채널배선문제가 스텐다드셀 레이아웃의 일부라고 가정하고 그림 7(a)와 같이 터미날 t_2, t_3, t_4가 AND게이트에 해당하는 기초셀이라 하자. 이 경우 t_2와 t_3는 논리적으로 동등한 터미날이고, 따라서 그림 7(b)와 같이 네트리스트를 바꾸어도 IC의 기능에는 아무런 변화가 없다. 그러나, 그림 7(b)에서 d_{max}=3 이고, 따라서 그림 7(c)와 같이 세 개의 수평트랙만으로도 배선이 가능하다.

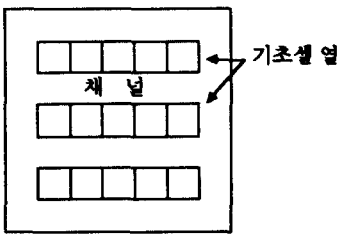


그림 6. 스텐다드셀 레이아웃에서 기초셀의 배치.

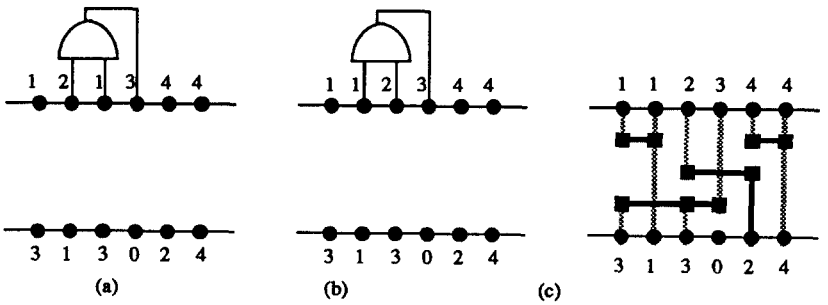


그림 7. 논리적으로 동등한 터미날들을 이용한 향상된 채널 배선.

마찬가지로 만약 초기의 채널배선문제에 대한 수직제한그래프에 사이클이 존재할 경우에는 이를 논리적으로 동등한 터미날들을 이용하여 그 사이클을 제거할 수 있으며 또한 v_{max}의 감소도 생각할 수 있다.

따라서 채널배선의 전처리과정으로 다음과 같은 문제를 정의한다. 즉, 채널배선문제가 주어지고 그의 터미날 중 논리적으로 동등한 터미날들의 집합의 모임이 있을 때 이들을 교환하여 d_{max}, v_{max}등을 최소화하거나 또는 수직제한그래프의 사이클을 제거하는 문제이다. 지금까지 이 문제에 대하여 d_{max}, v_{max}의 최소화, 또는 사이클의 제거 등을 동시에 해결하는 방법은 아직 알려진바 없다.

본절에 이어 다음 소절에서는 d_{max}를 최소화 하는 알고리즘등을 서술하고, 위의 문제와 비슷하거나 또는 축소된 경우에 대하여 기존 발표된 결과를 간단히 소개한다.

3.2. d_{max}의 최소화

채널배선문제가 주어졌을 때 $f(\cdot)$를 각 터미날에 네트번호를 할당하는 할당함수라고 정의한다. 예를 들어, 그림 1(a)의 채널배선문제에서 $f(t_1)=1$, $f(t_2)=2$, $f(t_3)=1$, $f(t_4)=3$, $f(t_5)=4$, $f(t_6)=4$, $f(b_1)=3$, $f(b_2)=1$, $f(b_3)=3$, $f(b_4)=0$, $f(b_5)=2$, $f(b_6)=4$이다. 할당함수 f 가 주어졌을 때 행 i 의 밀도 d_i를 $d_i(f)$로, 최대 밀도 d_{max}를 $d(f)$로 표시한다.

여기서, 논리적으로 동등한 터미날의 집합의 모임이 주어졌을 때 이들 터미날을 교체하여 다른 할당함수 f'를 얻을 수 있다. 예를 들어 그림 7(b)와 같이 네트 1 을 t_2에 할당하고 네트 2를 t_3에 할당하면 $f'(t_2)=1$, $f'(t_3)=2$ 이고 나머지 $f'(t_i)=f(t_i), i=2,3$, $f'(b_j)=f(b_j)$, $j=1,2,\ldots,6$, 으로 새로운 할당함수를 얻는다. 따라서, d_{max}를 최소화하는 것은 논리적으로 동등한 터미날을 이용하여 $d(f)$가 최소인 f를 찾는 것과 동일하며, 이는 NP-hard라고 알려져 있다[10].

본 소절에서는 이러한 f를 찾는 휴리스틱방법으로서 반복적 향상 알고리즘과 시뮬레이티드 어닐링에 의한 알고리즘을 소개 한다[10].

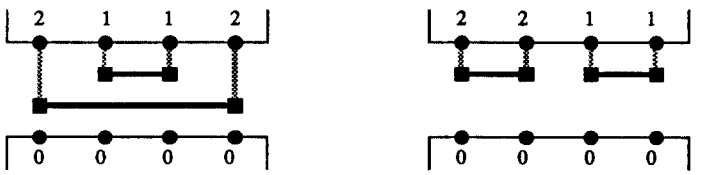
임의의 할당함수 f와 $d_i(f)$, $i=1,\ldots,n$, 가 주어졌을 때 두개의 교환가능한 터미날을 바꾸어 새로운 할당함수 f'를 얻었다 하자. 이때 f'을 f의 이웃이라 하며, 본 소절에서 소개하는 두 알고리즘에서는 f의 이웃 f'에 대한 $d_i(f')$를

신속히 계산할 필요가 있다. 두개의 터미날을 교환하였을 때 이 두 터미날을 포함하는 네트를 k_1, k_2 라 하자. 터미날 교환에 의해 네트 k_1과 k_2에 생기는 변화에 따른 d_i의 변화는 각 네트 입장에서 간단히 계산할 수 있다. 다음은 네트 k_1 입장에서의 d_i의 변화를 계산하는 과정이며 네트 k_2의 입장에서의 d_i의 변화도 마찬가지로 계산할 수 있다.

```
if ℓ'(k1) < ℓ(k1)   then
    for i:=ℓ'(k1)   to  ℓ(k1)-1   do   di:=di+1;
if ℓ'(k1) > ℓ(k1)   then
    for i:=ℓ(k1)    to  ℓ'(k1)-1  do   di:=di-1;
if r'(k1) > r(k1)   then
    for i:=r(k1)+1  to  r'(k1)    do   di:=di+1;
if r'(k1) < r(k1)   then
    for i:=r'(k1)+1 to  r(k1)     do   di:=di-1;
```

위의 식에서 $\ell(k_1)$과 $r(k_1)$은 네트 k_1의 가장 왼쪽과 가장 오른쪽에 있는 터미날의 위치를 의미하고, $\ell'(k_1)$과 $r'(k_1)$은 네트 k_1의 변화로 인한 가장 왼쪽, 오른쪽에 있는 터미날의 위치를 의미한다.

그림 8(a)에 보인 채널배선문제의 t_2와 t_4를 교환함으로써 생긴 네트 1과 2, 그리고 d_i의 변화를 그림 8(b)에 나타냈다.



l(1) = 2, r(1) = 3, l(2) = 1, r(2) = 4.
d1 = 1, d2 = 2, d3 = 2, d4 = 1.
(a) 교환전의 f.

l'(1) = 3, r'(1) = 4, l'(2) = 1, r'(2) = 2.
d1 = 1, d2 = 1, d3 = 1, d4 = 1.
(b) t2 와 t4를 교환한 후의 f'.

그림 8. 두터미날의 교환에 의한 di의 변화.

3.2.1. 반복적 향상 알고리즘

반복적 향상 알고리즘은 초기 할당함수 f로 부터 시작하여 f 보다 비용이 적은 이웃 f'를 찾고, 다시 f'로 부터 같은 작업을 그 비용에 더 이상의 향상이

발견되지 않을 때까지 반복하면서 최적의 해답을 구하는 방법이다. f로 부터 f'를 얻을 때 필요한 비용함수 두가지를 소개한다.

두가지 비용함수 중 하나는 각 행 i의 밀도를 벡터 형태로 표시한 비용함수이다. 어떤 할당함수 f에 대해 비용함수는 $D(f)=(d_1,d_2,\ldots,d_n)$으로 표시한다. 두개의 할당함수 f와 f'가 있을 때 모든 i=1,...,n에 대해 $d_i(f') \leq d_i(f)$이면, f'로 부터 최적의 해답을 찾기 위해 노력한다. 그림 8 에서의 비용함수는 D(f)=(1,2,2,1), D(f')=(1,1,1,1)이고, D(f)의 밀도 d_i들이 D(f')의 밀도 보다 크거나 같다. 이 경우 f' ≤ f이므로 f'로부터 다음 이웃을 찾는다. 그러나, 그림 9에 나타난 f와 f'에서처럼 $d_2(f)=2 \geq d_2(f')=1$ 이고 $d_5(f)=2 \leq d_5(f)=3$ 이어서, f와 f' 중 어떤 할당함수에서의 비용이 적을지 알 수 없는 경우도 있다. 이 경우에는 f'은 f 와 비교가 가능하지 않아 새로운 해로 여기지 않는다.

밀도를 벡터 값으로 나타낸 비용함수 D의 장점은 비용을 계산할 필요없이 각 밀도 값만 비교하면 되므로 알고리즘 수행속도가 매우 빠르다. 비용함수를 실수 값으로 잡으면 할당함수 f에 대해 비용함수 C(f)를 다음과 같이 정의할 수 있다.

$$C(f) = \lambda d^2(f) + \frac{1}{n}\sum_{i=1}^{n} d_i^2(f)$$

비용함수 C에서 전자는 채널의 최대 밀도를 줄이기 위한 것이고, 후자는 각 행 i의 d_i를 전체적으로 줄이려는 것이다. λ는 두 함수 사이의 상대적인 중요성을 나타내는 값이다. 실험을 해본결과 λ=0.5 일때 가장 좋은 결과를 나타냈다. 이 비용함수의 경우는 f'을 찾기 위해서 실제로 비용을 계산해야 하기 때문에 앞서 설명한 비용함수보다 시간은 걸리지만 보다 좋은 결과를 낸다.

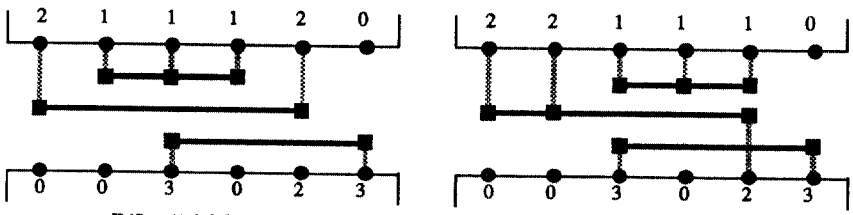


그림 9. f1과 f2의 교환으로 f와 f'의 관계를 알 수 없는 경우.

3.2.2. 시뮬레이티드 어닐링을 이용한 알고리즘

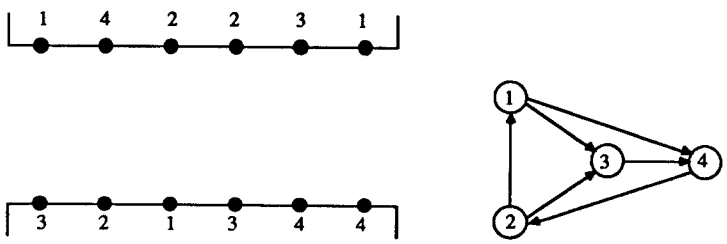
시뮬레이티드 어닐링 방법은 반복적인 향상 방법을 확장시킨 것으로써 전역의(global) 최적값에 (보다 더) 가까운 해를 얻기위한 방법으로 Kirkpatrick이 제안하였다[7]. 시뮬레이티드 어닐링을 이용한 알고리즘에서는 반복적인 향상 방법에서 사용한 두번째 비용함수와 같은 비용함수 C(f)를 사용한다.

$$C(f) = \lambda d^2(f) + \frac{1}{n}\sum_{i=1}^{n} d_i^2(f)$$

비용함수 C(f)의 각 함수와 λ는 앞서 설명한 것과 같다. 이 알고리즘에서는 초기온도 T_0로 부터 시작하여 $T=r^kT_0$, k=0,1,2,.... 의 식을 이용하여 점차 T를 감소 시킨다. 여기서, r은 0과 1사이의 값을 갖는 온도의 변화를 결정하는 상수이다. 초기온도 T_0는 밀도 d가 증가하는 경우에도 모두 해로서 받아들일 수 있을 정도의 큰 값이어야 한다. 따라서, 비용함수 C에 의하여 T_0는 λd에 비례하는 값으로 설정한다. N을 이웃의 수라고 정의할 때 각 T에서의 반복횟수는 f' < f 인 f'를 얻었을 경우 2N번이거나, 그렇지 않은 경우는 N 번의 이웃을 찾는 시도를 할 때 까지이다. 온도가 여섯번 변할동안 계속 d(f)의 변화가 없거나 f' < f인 f'가 5% 이내로 발견될 경우 멈춘다.

3.3. 수직제한 그래프에서의 사이클의 제거

입력으로 주어진 네트의 갯수를 S라 하자. 그림 10(a)는 S가 4일 때의 각 터미널에 할당된 네트를 보이고, 그림 10(b)에 이러한 할당에 따른 수직제한 그래프를 나타낸 것이다. 그림 10(b)의 경우 3개의 사이클이 존재한다.



(a) f에 의한 할당. (b) 수직 제한 그래프 G(f).

그림 10. 초기의 할당 상태.

다른 어떤 할당함수 f로 부터 논리적으로 동등한 관계에 있는 터미날의 집합을 이용하여 여러가지 다른 할당함수를 얻을 수 있다. 그림 10(a)의 할당함수 f에 t_1과 t_2, t_4와 t_5, b_2와 b_3, b_5와 b_6이 서로 동등한 관계에 있을 때의 예를 보자. 할당함수 $f(t_1)$과 $f(t_2)$, $f(t_4)$과 $f(t_5)$, $f(b_2)$와 $f(b_3)$를 교환했을 때 얻어진 새로운 할당함수 f'은 그림 11와 같다. 할당함수 f'에 대한 그래프에서의 사이클의 수는 0이다.

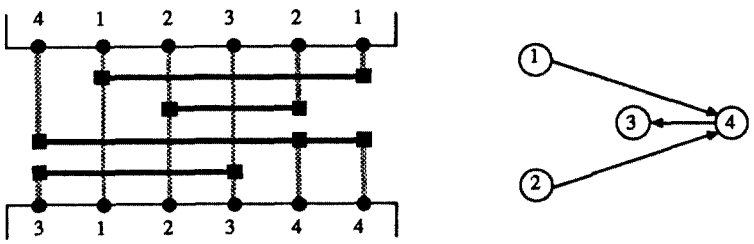
논리적으로 동등한 관계가 있는 터미날의 집합이 있어서, 이 두 터미날에 할당된 네트 번호를 바꾸었을 때 사이클의 감소가 생기면, 이 교환이 적절하다고 한다. 이런 적절한 교환을 찾아 할당함수 f에서 사이클의 갯수가 최소인 수직제한그래프를 갖는 할당함수를 찾는 경험적인 알고리즘을 아래에 제시한다.

단계 1 : i = 1.

단계 2 : 어떤 터미날 x_i에 대해 교환이 적절한 터미날 x_j가 존재하면 x_i와 x_j에 대해 교환작업을 한다.

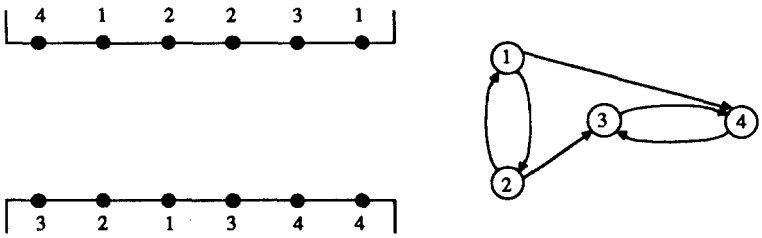
단계 3 : i < n 이면 i를 증가시키고 단계 2로 간다.
i = n 이고 단계 1부터 한번도 교환이 적절하지 않았다면 멈춘다. 그 이외는 단계 1로 간다.

그림 10(a)의 예에서 터미날 t_1과 t_2에 대해 네트 번호를 바꾸면 그림 12와 같이 되고, 이때의 사이클의 갯수는 2이다. t_1과 t_2에 대한 교환은 적절하므로 두 터미날을 교환한다. 나머지 논리적으로 동등한 관계의 터미날에 대해 이와 같은 과정을 반복하면, 그림 11과 같은 결과를 얻게된다.



(a) f'에 의한 할당. (b) 수직제한 그래프 G(f').

그림 11. 터미날의 교환으로 사이클이 없어진 경우.



(a) f"에 의한 할당. (b) 수직 제한 그래프 G(f").

그림 12. 두 터미날의 교환이 적절한 예.

터미날 할당을 향상시키는 알고리즘의 수행시간을 알아보자. 주어진 f에서 수직제한그래프 G(f)와 한번 교환 후의 G(f')를 만드는 데 필요한 시간은 각각 O(n)이다(n은 행의 갯수). G(f)를 만든 후 사이클을 찾는데 필요한 시간이 G(f)안에 정점과 간선을 V와 E라고 할 때 O(|V| + |E|)이다[5]. G(f')에서도 사이클을 찾는 시간은 G(f)에서의 그것과 같다. 그래서, 하나의 교환이 적절한지를 알아보는데 필요한 시간은 O(n + |V| + |E|)이 되고, 터미날 할당 알고리즘에서 전체 반복 횟수는 많아야 초기 사이클수 v_0 만큼이므로, M을 하나의 논리적으로 동등한 관계의 집합에서의 최대 터미날수라고 하면, 전체 알고리즘에 필요한 시간은

$$\begin{aligned} & O(v_0 \cdot 2n \cdot M(n + |V| + |E|)) \\ \leq\ & O(v_0 \cdot 2n \cdot M(n + 2n + n)) \\ \leq\ & O(v_0 \cdot M \cdot n^2) \end{aligned}$$

이다.

터미날 할당을 향상시키는 알고리즘의 단계 2 는 모든 논리적으로 동등한 터미날에 대하여 교환이 적절한지를 검사한다. 한편 논리적으로 동등한 터미날에 대해서 교환이 적절하지 않은 경우를 특성에 따라 구분하여 불필요한 계산을 줄이는 방법이 있다[14]. 이 방법은 사이클이 많은 경우 소개한 터미날 할당 알고리즘보다 속도가 빠르다.

3.4. 특별한 경우

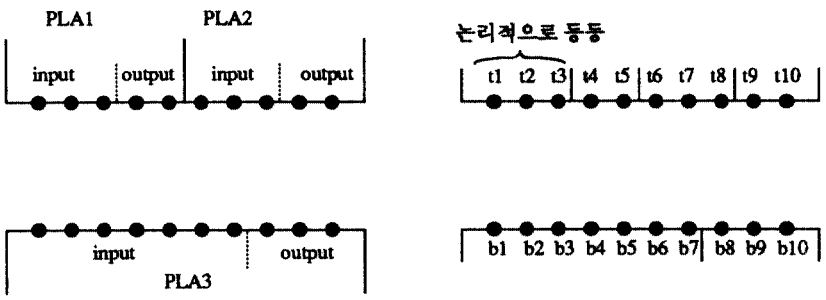
앞의 두 소절에서는 가장 일반적인 경우, 즉 주어진 채널배선문제에서 논리적으로 동등한 터미날들의 집합이 임의로 주어졌을 경우에 대한 알고리즘들을 기술하였다. 본 소절에서는 논리적으로 동등한 터미날들이 특별한 형태를 갖는 경우를 소개한다.

그림 13(a)와 같이 채널배선문제에서 위 아래 모듈이 PLA이고 이들에 필요한 터미날 갯수와 네트리스트만 정의하였다면, 한 PLA의 입력 또는 출력에 속한 터미날들은 논리적으로 서로 동등하다. 따라서, 터미날들은 d_{max} 또는 v_{max}가 작도록 그리고 수직제한그래프에 사이클이 존재하지 않도록 배정하고, 차후 PLA의 설계 및 채널배선을 수행하면 보다 효율적으로 칩 면적을 사용할 수 있다.

이를 보다 간략히 표시하면 그림 13(b) 처럼 논리적으로 동등한 터미날들이 연속적으로 나타나는 경우라고 생각 할 수 있다. 이 경우 해결 해야 할 문제는 2 단계로 다음과 같이 기술 할 수 있다[11].

1) 논리적으로 서로 동등한 터미날들을 교환하여 같은 네트에 속하는 터미날 중 가능한 많은 갯수의 터미날들이 같은 수평위치의 위 아래에 배치되도록 한다. 터미날들이 같은 네트에 속하도록 한다. 이는 d_{max}를 작게 하는 것과 밀접한 관계가 있으며, 또한 수직제한 그래프를 단순화 시키는 장점도 있다.

2) 수평위치가 같은 위 아래 터미날들이 같은 네트에 속할 경우 이들은 수직제한그래프의 형성에 아무런 영향을 미치지 않는다. 따라서, 나머지 터미날들을 수직제한그래프에 사이클이 생기지 않고, 또한 v_{max}가 최소가 되도록 배정한다.



(a) PLA 모듈들에 대한 채널배선.

(b) 논리적으로 동등한 터미날이 연속적으로 존재하는 경우로의 변환.

그림 13. 논리적으로 동등한 터미날들의 특별한 경우.

문제 1)에 대해서는 최근 선형알고리즘이 발표되었으나[11], 위의 두 단계를 모두 고려한 방법은 아직 발표되지 않았다.

3.5. 터미날의 위치에 변경이 가능한 경우

스텐다드셀 또는 게이트어레이 레이아웃에서 먼저 회로가 주어 졌을 때, 이에 필요한 셀들을 기초셀 라이브러리로부터 가져와 칩의 수평열에 배치하게 된다. 이때, 칩의 수평열의 길이를 L이라 하고 이 열에 배치된 기초셀들의 너비가 각각 $\ell_1, \ell_2, \ldots, \ell_c$라 할 때 $\sum_{i=1}^{c} \ell_i \leq L$ 을 만족하여야 한다. 그러나, 일반적으로 $\sum_{i=1}^{c} \ell_i < L$ 이며, 따라서 각 기초셀들은 칩의 수평열에서 좌우 이동이 가능하다.

이로부터 논리적으로 동등한 터미날 대신 터미날들이 좌우로 움직일 수 있을 경우 이들은 조정하여 d_{max}, 또는 v_{max}를 적게 하거나, 또는 사이클을 없게하는 방법을 생각할 수 있다. 예를 들어 그림 1(a)의 예에서 채널의 위 아래가 하나의 모듈로 구성되어 있고 이 모듈의 수평이동이 가능하다면, 그림 14에 보인 바와 같이 채널의 아래 모듈을 오른쪽으로 한 행을 이동하여 d_{max}를 셀으로 만들 수 있고, 따라서 세개의 수평트랙만 가지고 배선할 수 있다. 이에 관련된 결과로 $O(n^2 \log n)$ 시간(n은 네트수)에 d_{max}를 최소화시키는 알고리즘이 발표되었고[9], 이 문제를 보다 단순한 형태에서 연구한 많은 이론적인 결과가 참고문헌[6]에 나타나있다.

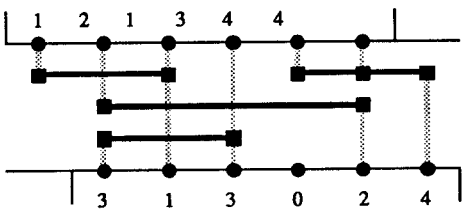


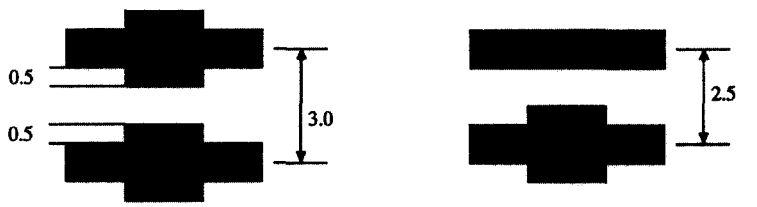
그림 14. 모듈의 수평이동이 가능한 경우 향상된 채널배선.

4. 후처리 알고리즘

4.1. 동기및 문제정의

지금까지 주어진 채널배선문제에 대하여 논리적으로 서로 동등한 터미날이나, 혹은 이동이 가능한 터미날들을 이용하여, 채널배선의 dmax 또는 vmax 등을 최소화시키거나 수직제한그래프에 사이클이 존재하지 않도록 하는 등의 전처리 과정에 대하여 소개하였다. 반면에 본 절에서는 채널배선이 완료되었을 때 이 배선을 변경하여 채널의 높이를 보다 작게하는 후처리 알고리즘에 대하여 기술한다.

채널배선에서 컴팩션의 필요성은 via의 크기로 부터 발생한다. 일반적으로 via의 높이와 너비는 배선에 사용하는 선들의 폭보다 크다. 예를 들어 선의 폭이 1.0, via의 높이와 너비가 각각 2.0이고, 선 또는 via 경계간의 분리가 최소한 1.0이어야 한다면, 배선의 결과는 그림 15(a)와 같이 수평트랙 간의 간격이 최소한 3.0이어야 한다. 그러나, 그림 15(b)의 경우처럼 수평으로 동일한 위치에 두 via가 위 아래로 인접하게 놓여진 경우가 없으면, 수평 트랙간의 간격을 2.5로 할 수 있어 채널의 높이를 보다 작게 할 수 있다.



(a) 두개의 via가 수직으로 인접한 경우의 수평트랙 간의 최소간격.

(b)두개의 인접한 수평트랙중 한곳에만 via가 있는 경우의 수평트랙간의 최소간격.

그림 15. 수평트랙간의 최소간격에 대한 설명.

이로부터, 채널배선의 후처리 과정으로서, 채널의 높이를 작게 하기 위한 방법으로, 주어진 배선을 변경하여 수직으로 서로 인접한 via의 갯수를 최소화하는 문제를 생각할 수 있다. 이를 해결하는 방법으로 주어진 배선의 수평트랙위치를 재조정하거나(트랙 permutation), 배선을 국한적으로 다시 하는(재배선) 등의 방법이 있으며, 이들을 4.2와 4.3의 두 소절에 걸쳐 간략히 기술한다.

이 밖에도 채널의 높이를 작게하는 방법으로써 via offsetting[4]이라는 기법을 사용할 수 있다. 주어진 채널배선결과에서 수직으로 인접한 두 via를 서로 인접하지 않도록 할 수 없을 경우, 해당 수평트랙간의 간격은 3.0이어야 한다(그림 15(a)). 그러나, 그림 16에서와같이 두 via를 위 아래로 0.1씩 이동하면 수평트랙간의 간격을 2.8로 할수 있다.

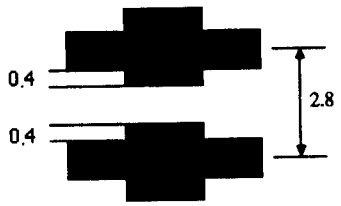


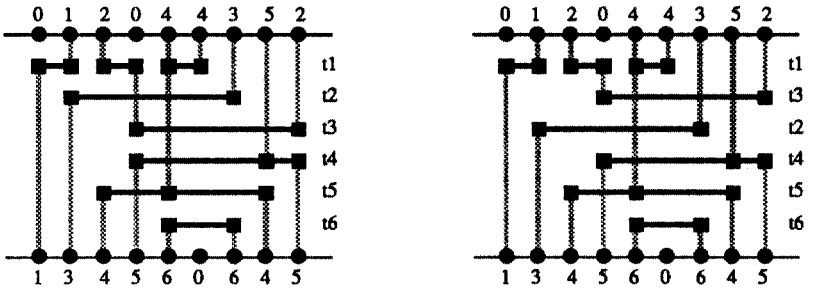
그림 16. ViaOffsetting기법에 대한 설명.

마지막으로, via shifting기법은 via를 이에 연결된 선을 따라 좌우상하로 이동하여 수직으로 인접한 via쌍의 갯수를 적게하는 방법이다. 그런데, 최근에는 via의 이동을 통하여 이러한 갯수를 줄이기보다는 전체채널의 높이를 최소화하는 알고리즘이 발표되었다. 4.4 에서는 이에 관하여 간략히 기술한다.

4.2. Track Permutation 방법.

채널배선 결과가 주어졌다고 하자. 임의의 두 인접한 수평트랙에 수직으로 인접한 두 via가 각각 존재한다면, 이 두 수평트랙을 충돌트랙쌍이라고 한다. 예를들어 그림 17(a)에서 트랙t_1과 트랙t_2의 두번째 행에 via가 각각 존재하므로 이 두 트랙은 충돌트랙쌍이다.

본 소절에서는 충돌트랙쌍의 갯수를 줄이는 한 방법으로써 트랙 permutation 이라는 기법을 소개한다. 트랙 permutation은 주어진 채널배선에서 수평트랙들의 위치를 교환하여 충돌트랙쌍의 갯수를 최소화하는 수평트랙의 permutation을 구하고, 이로부터 새로운 채널배선결과를 얻는 방법이다. 예를 들어 그림 17(a)에 보인 채널배선결과의 수평트랙 t_2와 수평트랙 t_3의 위치를 교환하면, 그림17(b)에 보인 바와 같은 새로운 채널배선 결과를 얻으며, 충돌트랙쌍의 갯수는 다섯개에서 두개로 적어진다.



(a) 주어진 채널배선결과. (b) 트랙 t_2와 t_3를 교환한 결과.

그림 17. 트랙 permutation에 의한 충돌트랙쌍의 최소화.

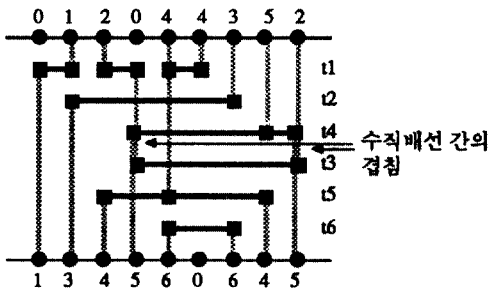


그림 18. 그림 17(a)의 배선결과에서 t_3와 t_4를 교환하여 얻은 배선결과.

수평트랙 permutation으로부터 항상 새로운 채널배선 결과를 얻을 수는 없다. 예를 들어 그림17(a)에서 보인 배선에서 트랙 3과 트랙 4의 위치를 교환하면, 그림18에서 보는 바와 같이 서로 다른 네트에 대한 수직배선간의 겹침이 생긴다. 따라서, 수직배선의 겹침이 없으면서 충돌트랙쌍의 갯수를 최소화하는 수평트랙 permutation을 얻어야 하는데, 이는 주어진 채널배선으로부터 유향비순환그래프(directed acyclic graph)를 구성하고 다음에 정의하는 그래프 문제(separation 문제)를 해결하므로써 쉽게 구할 수 있다.

먼저, 주어진 채널배선 S로부터 그래프 G를 구성한다. G의 정점은 S의 각 수평트랙을 나타내고, S에서 트랙 t_i의 어떤 via가 채널의 같은 행에서 트랙 t_j의 via(트랙 t_i의 via와 다른 네트에 관련된 via에 한해서) 위에 존재할때, 트랙t_i를 나타내는 G의 정점으로부터 트랙 t_j를 나타내는 정점으로 간선을 연결한다. 예를 들어, 그림17(a)의 채널배선으로부터 구성한 그래프를 그림19에 보인다. 여기서 수평트랙 t_i를 나타내는 정점을 편의상 't_i'로 동일하게 표시한다. 이때, separation 문제는 구성된 그래프의 각 정점에 1부터 정점의 갯수까지의 번호를 다음의 두 조건을 만족시키도록 부여하는 문제이다.

i) 두 정점 t_i와 t_j에 부여된 번호 p, q가 $p < q$이면, t_j로부터 t_i로 연결된 에지가 없어야 한다.

ii) 두 정점 t_i와t_j에 부여된 번호가 연속적일때 두 정점간을 연결하는 에지를 충돌에지라고 하면, 이 충돌에지의 총 갯수가 최소가 되어야 한다.

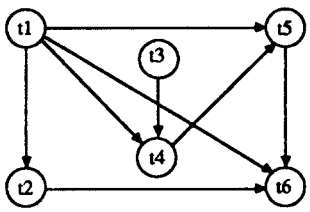


그림19. 그림 1(a)로 부터 구성한 유향 비순환 그래프.

주어진 채널배선 문제로부터 구성한 그래프에 대하여 separation 문제에 대한 해를 얻었다고 하자. 그래프의 각 정점들의 고유번호에 따라서 채널배선의 각 트랙의 순서를 결정한다. 이때 조건 i)에 의해서 임의의 두 트랙 t_i와 t_j간에는 아무런 수직배선의 겹침이 일어나지 않는다. 한 조건 ii)에 의하여 충돌트랙쌍의 갯수가 최소가 된다. 예를 들어 그림 19의 그래프에 대한 separation 문제의 해를 구하고, 이로부터 채널배선의 결과를 얻으면 그림17(b)와 동일하다.

Separation 문제는 선형시간 즉, O(e)시간에 풀수 있다[15]. 여기서 e는 주어진 그래프의 간선의 갯수이다. 따라서, 수직선의 겹침이 없고 충돌트랙쌍의 갯수가 최소인 수평트랙 permuatation을 선형시간에 구할 수 있다.

4.3. 재배선 방법

재배선 방법은 주어진 채널배선의 일부를 수정하여 충돌트랙쌍의 갯수를 줄이는 방법으로 보통 4.2절의 트랙 permutation 방법을 적용한 후 수행한다. 이 방법은 maze 배선을 이용하는데, 예를 들어 설명하면 다음과 같다.

배선의 일부분을 나타내고 있는 그림20(a)에서 via v_1을 없애면 그림 20(b)와 같이 네트 n_1에 대한 배선이 둘로 양분된다. 이 양분된 배선간에 maze 배선을 이용하여 재배선하면 그림20(c)와 같이 수직으로 인접한 via쌍이 없는 배선결과를 얻을 수 있다.

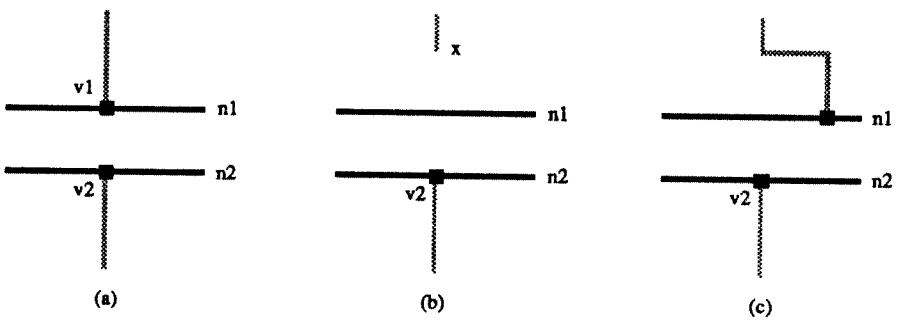


그림 20. 재배선에 의한 충돌트랙쌍 갯수의 감소.

그림21은 그림17(b)의 내용에 재배선 기법을 적용시켜서 얻은 배선으로, 여덟번째 컬럼의 배선일부를 아홉번째 컬럼으로 재배선 시킴으로써 층돌트랙쌍의 갯수를 하나 줄였다.

그러나, 주어진 채널배선이 매우 복잡한 상태일 때에는 이러한 부분적인 재배선 방법이 효과가 없는 경우가 생긴다. 이 경우, 빈 트랙을 배선이 가장 복잡한 부분에 새로 추가하여 재배선방법을 수행하게 되면 층돌트랙쌍의 수가 현저히 감소할 수도 있다.

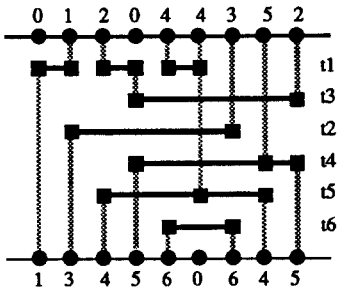


그림 21. 그림 17(b)에 보인 채널배선 결과에서 네트 4를 재배선하여 얻은 결과.

4.4 Via Shifting에 의한 방법

임의의 채널배선문제에 대한 배선결과가 주어졌다고 가정할때, 이 배선에서 via의 높이를 α, 선의 폭을 β, 그리고 선 또는 via 외곽선간의 최소거리를 γ라 하자. 채널의 행 c에서 via가 있는 수평트랙의 갯수를 v_c, 순수한 선만 존재하는 수평트랙의 갯수를 w_c라하면, 이 행을 최대한으로 압축하여 얻을 수 있는 행의 높이 h_c는

$$h_c = \alpha v_c + \beta w_c + \gamma (v_c + w_c + 1)$$
$$= (\alpha + \gamma) v_c + (\beta + \gamma) w_c + \gamma$$

이고, 따라서, 실현가능한 가장 작은 채널의 높이 H는

$$H = \max_c \{(\alpha + \gamma) v_c + (\beta + \gamma) w_c + \gamma\}$$

이다.

Via는 그에 연결된 선을 따라 다른 네트에 대한 배선이나 via를 만나기 전까지 상하좌우로 이동할 수 있고, 이를 통하여 새로운 채널배선 결과를 얻을 수 있다. 그림 22에 각 via가 움직일 수 있는 위치를 굵은 선으로 표시하여 보인다.

참고문헌[16]에서는 via의 수평이동만 고려하여, 채널의 높이 H를 최소화 시키는 새로운 via들의 위치를 얻는 알고리즘을 발표하였다. 발표된 알고리즘은 네트워크흐름(network flow)기법을 사용한다. 이를 위하여, 주어진 채널배선결과에 대하여 다음과 같이 네트워크 $G\psi=(N,E,\Psi)$를 구성한다. 여기서, $G=(N,E)$는 방향성 그래프이고, Ψ는 G의 각 에지에 지정된 음이 아닌 정수로 표시되는 용량(capacity)을 나타낸다.

$N = \{s,t\} \cup \{v|v$는 각 via에 대응$\} \cup \{c|c$는 각 행에 대응$\}$

$E = \{(s,v)|$모든 via에 대해 적용$\} \cup \{(c,t)|$모든 행에 적용$\} \cup$

$\{(v,c)|$via v가 행 c로 이동할 수 있는 경우$\}$,

$\Psi(s,v) = 1$, 모든 via v 에 대하여,

$\Psi(v,c) = 1, (v,c) \in E$ 인 경우,

$\Psi(c,t) = \lfloor (x-(\beta+\gamma)f_c-\gamma)/(\alpha-\beta) \rfloor$, 여기서 x는 임의의 음이아닌 수이며, $f_c = v_c+w_c$이다. 그림22에 보인 채널배선결과로부터 구성한 네트워크 $G\psi$를 그림23에 보인다. 구성된 네트워크에서 시작점 s로부터 종점 t까지의 최대흐름(1/0 흐름[5])을 F라 할때, 이것이 via의 갯수와 같으면, 이 흐름 F를 '합당한 흐름'이라고 정의한다. $G\psi$로부터 합당한 흐름 F를 찾아냈을 경우 이로부터 via의 위치를 얻을 수 있다. 즉, F가 합당한 흐름이므로 F는 via의 갯수와 같고, 따라서, 임의의 via v로부터 행들로 연결된 간선 중 그 흐름이 1인 간선이 정확히 하나 존재한다. 이로부터 v를 흐름이 1인 간선에 연결된 행에 배치하며, 나머지 다른 via들에 대해서도 같은 방법으로 그들이 배치될 행들을 찾을 수 있다. 다음의 정리는 F를 이용하여 H가 최소가 되기 위한 via들의 위치를 찾는데 유용하다.

정리. 주어진 x에 대한 $G\psi$의 최대흐름 F가 합당한 흐름이면 이 흐름에 의한 via의 위치조정을 통하여 채널의 높이를 x로 할 수 있으며, 그 역도 성립한다.

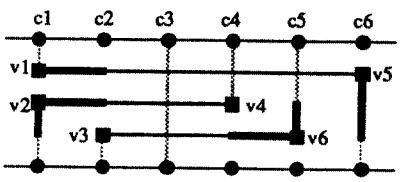


그림 22. Via의 이동이 가능한 구간(굵은 선으로 표시)

FEASIBLE(x)를 주어진 x에 대하여 G_ψ에 합당한 흐름의 존재여부를 조사하는 프로그램이라고 하자. 주어진 채널배선에 대하여 먼저 H가 가질수 있는 최소값 L(채널의 각 행에 via가 없다고 가정했을 때의 H값)과 최대값 U(채널의 각 행의 모든 수평트랙에 via가 존재한다고 가정했을 때의 H값)를 계산한다. 다음, x를 L과 U사이의 값으로 정한 FEASIBLE(x)를 이분검색(binary search)에 사용하여 FEASIBLE(x)가 '참'인 최소의 x(=H)값을 계산하고, 이때 얻어진 via의 위치를 통하여 높이가 최소인 채널배선결과를 얻는다. 이를 위한 전체 수행시간은 n_t, n_c, n_v를 각각 주어진 채널배선결과에서 수평트랙의 갯수, 행의 갯수, 그리고 via의 갯수라고 할때, $O(n_t n_c (n_v+n_c)\log^2(n_v+n_c))$이다.

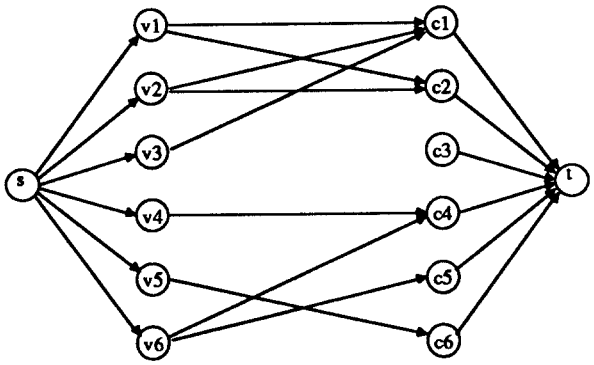


그림 23. 그림 20의 예로 부터 구성한 그래프 G(N,E)

5. 결론

본장에서는 채널배선에서 전처리 또는 후처리 과정을 통하여, 배선된 채널의 높이를 더욱 작게 하는 방법에 대하여 기술하였다. 전처리 과정으로서는 주로 논리적으로 동등한 터미날들의 집합을 이용하여 dmax를 작게 하였고, 문제의 수직제한그래프에 사이클이 존재하지 않도록 하는 방법등에 대하여 기술하였다. 그러나 지금까지 알려진 방법들은 하나의 목표(즉, dmax의 최소화 또는 사이클의 제거등)만을 위한 방법으로 여러 목표를 동시에 고려한 방법에 대한 연구가 필요하다. 후처리 과정으로서 본장에서는 주어진 배선을 수정하여 수직으로 인접한 via들이 적은 배선으로 바꾸는 방법들에 대하여 기술하였다.

6. 참고 문헌

[1] H.N.Brady, "An Approach to Topological Pin Assignment, "IEEE Trans. on CAD, Vol.CAD-3, No 3, pp.250-255, July 1984.

[2] C.K.Cheng and D.N.Deutsch, "Improved Channel Routing by Via Minimization and Shifting," Proc. 25th DAC, pp 677-680 , 1988.

[3] J.Cong and D.F.Wong, "How to Obtain More Compactable Channel Routing Solutions, "Proc. 25th DAC, pp 663-666, 1988.

[4] D.N.Deutsch, "Compacted Channel Routing," Proc. ICCAD, pp 223-225 1985.

[5] S.Even, Graph Algorithm, Computer Science Press, Rockville, MD, 1979

[6] I.S.Gopal, D.Coppersmith and C.K.Wong, "Optimal Wiring of Movable Terminals," IEEE Trans.on Computers, Vol C-32 , No-9, Sept 1983.

[7] A.Kirkpatric, C.D.Gelatt.and M.P.Vecchi, "Optimization by Simulated Annealing , "Science, Vol.220, No.4598, pp.671- 680, May 1983.

[8] H.Kobayashi and C.E.Drozd, "Efficient Algorithms for Routing Interchangeable Terminals, "Proc.ICCAD, pp.79-81, 1984.

[9] A.S.Lapaugh and R.Y.Pinter, "On Minimizing Channel Density by Lateral Shifting, "Proc.ICCAD, pp.121-122, 1983.

[10] H.W.Leong and C.L.Liu , "Permutation Channel Routing," Proc. ICCAD1985, pp.579-584.

[11] L.S.Lin and S.Sahni, "Maximum Alignment of Interchangeable Terminals, "IEEE Trans.on Computers, Vol.37, No 10, Oct 1988.

[12] T.Ohtsuki, ed, Layout Design and Verification, North-Holland, Amsterdam, the Netherlands, 1986.

[13] T.G.Szymanski , "Dogleg Channel Routing is NP-Complete," IEEE Trans.on CAD, Vol.CAD-4, pp.31-41, 1985.

[14] M.Terai, "A Method of Improving the Terminal Assignment in the Channel Routing for Gate Arrays, "IEEE Trans.on CAD, Vol.CAD-4, No-3, pp.329-336, July 1985.

[15] D.F.Wong and C.L.Liu, "Compacted Channel Routing with Via Placement Restrictions,"Integration , the VLSI Journal, North-Holland, pp. 287-307, 1986.

[16] D.F.Wong and C.L.Liu, "Optimal Via - Shifting in Channel Compaction,"Proc. EDAC, pp 186-190 1990.

[17] X.Yao, M.Yamada and C.L.Liu,"A New Approach to the Pin Assignment Problem,"Proc. 25th DAC, pp. 566-572, June 1988.