論文2003-40SD-11-9

버퍼 삽입 위치 및 배선 제한을 고려한 Buffered 배선 트리 구성

(Buffered Routing Tree Construction under Buffer Location and Wiring Constraints)

鄭 東 植 *, 金 德 歡 ***, 林 鍾 錫 **

(Dong-Sik Jeong, Deok-Hwan Kim, and Chong-S Rim)

요 약

본 논문에서는 매크로 또는 IP 블록 같은 장애물로 인하여 버퍼삽입과 배선에 제한이 있는 환경에서 연결 지연시간을 최소화하기 위한 배선 및 버퍼삽입위치를 동시에 구하는 방법을 제안한다. 제안한 방법에서는 새로운 격자그래프를 도입하여 배선 또는 버퍼삽입이 불가능한 영역을 효과적으로 표현하고 이 격자그래프 상에서 동적프로그래밍을 사용하여 배선 트리의 구성과 동시에 버퍼의 삽입여부 및 위치를 구한다. 제안한 방법은 기존 방법에 비하여 유사한 배선길이 및 작은 수의 버퍼를 삽입하면서도 평균 19% 정도의 여유 지연시간이 향상되었다.

Abstract

In this paper, a simultaneous buffer insertion and routing method is proposed under the constraints of wire and buffer locations by macro or IP blocks. A new grid graph is proposed to describe the regions in which buffers(or both wires and buffers) are not available. Under this grid we describe a method of constructing a buffered tree that minimize the maximum source to sink delay. The method is based on the dynamic programming with pruning unnecessary partial solutions. The proposed method improved the slack time of the delay by 19% on the average while using less buffers and similar wire length.

Keyword : buffer insertion, routing constraints performance driven ronting

I. 서 론

VLSI 기술의 급속한 발전으로 인하여 집적도가 높아짐에 따라 와이어 연결에 의한 지연시간(이하 '연결지연'이라 한다)이 상대적으로 커져 연결지연이 회로의 전체성능을 결정하는 주요한 요소가 되었다. 연결지연을 줄이기 위해 여러 방법이 제안되었는데 이 중 버퍼를 사용한 연결지연 단축방법이 가장 효과적이며 이에 대한 활발한 연구가 이루어지고 있다.

연결지연 감소를 위해 버퍼의 개수와 위치를 결정하는 고전적인 방법으로는 Ginneken의 동적프로그래밍을 이용한 방법을 들 수 있다[8]. 이 방법은 주어진 배선 트리 상에서 연결지연을 작게 하기 위한 버퍼의 수와

* 正會員, 三星電子
(Samsung Electronics Co., LTD.)
** 正會員, 西江大學校 컴퓨터學科
(Sogang University, Computer Science and Engineering)
※ 이 연구는 2003년도 서강대학교 교내 연구비 지원에 의하여 이루어졌음
接受日字:2002年12月23日, 수정완료일:2003年11月5日

위치를 결정한다. 그러나, 이 방법은 고정된 배선트리에서 와이어가 분기되는 점, 즉 스타이너 노드에서만 버퍼의 삽입 여부를 결정하기 때문에 최적의 해를 찾기 어렵다. 실제로 와이어 중간에 버퍼를 삽입하여 연결지연을 더욱 줄일 수가 있는데 Alpert 등[1]은 수학적인 근거에 의하여 버퍼삽입 영역을 확장하고 와이어 중간에도 버퍼를 삽입하는 방법을 제안하였다. 그런데 이 방법에서는 하나의 와이어 w에 대하여 버퍼삽입 여부 및 위치를 계산할 때 w의 하부 트리의 전체 정전용량을 사용하여 계산하기 때문에 정확한 결과를 기대하기 어렵다(실제로 하부트리에 버퍼가 삽입될 경우 w에서 하부로 보는 정전용량 값은 전체 값과 상당한 차이가 있다).

한편, Okamoto 등[10]은 배선과 버퍼삽입을 동시에 수행하는 방법을 제안하였다. 이는 A-tree를 구하는 방법에 Ginneken의 버퍼 삽입 방법을 접목한 방법으로 연결지연이 보다 작은 배선트리를 얻을 수 있다.

최근, VLSI는 단순한 표준셀에 의한 설계가 아닌 매크로 블록(또는 IP 블록)이 혼합된 형태의 설계가 이루어지고 있다. 이러한 형태의 배선에서는 매크로 블록의 상태에 따라 그 블록 내에 배선을 하지 못할 수가 있고 또는 배선이 가능하더라도 버퍼를 삽입하지 못할 경우도 있어 이러한 제한에 따른 적절한 배선방법이 필요하다. 예를 들어 <그림 1>과 같이 배선만 가능한 영역이 있을 경우, 최단 경로로 배선하거나 또는 이 영역을 우회하고 버퍼를 삽입하여 배선할 수 있고 이중 연결지연이 작은 배선을 선택하여야 하는데, 앞에서 기술한 배선방법들로는 이를 해결할 수 없다.

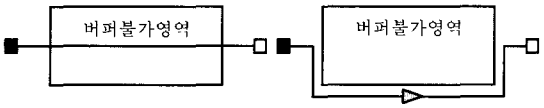


그림 1. 버퍼불가영역이 있는 경우의 배선.
Fig. 1. Routing with buffer constraint.

이러한 환경에서 Zhou는 소스와 하나의 터미널을 가진 네트에 대하여 배선과 버퍼 삽입을 동시에 수행하는 방법을 제안하였다[11]. 다중터미널 네트인 경우 Cong 등[6]이 버퍼불가영역과 배선불가영역을 고려하여 배선과 버퍼삽입을 동시에 수행하는 방법을 제안하였으나 이는 버퍼삽입 위치가 제한적일 경우에 대한 방법이다. 그리고 Hu 등[9] 이 제안한 방법에서는 배선트리를 C-트리[3] 형태로 구성한 후 버퍼 삽입이 가능하도록 트리형태를 수정하면서 Ginneken의 방법으로 버퍼를 삽입하고 있으나 이 역시 트리형태가 제한적이므로 최적의 배선과는 거리가 있다.

본 논문에서는 새로운 배선 및 버퍼삽입 방법을 제안한다. 배선 제약조건을 고려하기위해 새로운 격자 그래프를 구성하고 그 격자 그래프 상에서 배선 트리의 구성과 동시에 버퍼의 위치를 결정하는데 동적 프로그래밍을 사용하여 다양한 부분트리의 구성을 확장해 나간다. 부분트리의 확장 과정에서 격자 노드에서의 버퍼 삽입만을 고려하는 것이 아니라 버퍼삽입이 가능한 간선의 모든 영역에 대해 참고 문헌 [1]의 근거를 통하여 버퍼의 삽입 위치를 결정한다. 제안한 방법의 실험 결과 기존 방법에 비하여 유사한 배선길이 및 작은 수의 버퍼를 삽입하면서도 평균 19% 정도의 여유지연시간이 향상되었다.

이어지는 Ⅱ 장에서는 제안한 방법의 기술에 필요한 지연시간 모델, 용어, 문제정의 등을 기술하며, Ⅲ 장과 Ⅳ 장에서는 제안한 배선 및 버퍼 삽입 방법 및 그에 대한 실험결과를 보이고 V장에서 결론을 맺는다.

Ⅱ. 기본정리

1. 지연 시간 모델

본 논문에서는 연결지연에 Elmore 지연시간 모델[7]을 사용한다. 이를 위하여 각 와이어에 대해서 π 모델을 사용하는데 π 모델에서는 와이어의 정전용량이 와이어 저항을 중심으로 양쪽으로 균등하게 절반씩 분포되어 있다고 가정한다.

와이어의 단위길이 당 저항과 정전용량의 값을 각각 r_0와 c_0라고 하자. 와이어의 길이가 l_e이면 이 와이어의 저항 r_e와 정전용량 c_e는 각각 $r_e = r_0 l_e$, $c_e = c_0 l_e$이다. 만일 이 와이어에 연결된 다운스트림(downstream) 쪽의 정전용량을 c_s라고 하면 π 모델에 의한 와이어의 Elmore 지연시간 D_w는 다음과 같다.

$$D_w = r_e \left(\frac{1}{2} c_e + c_s \right)$$

버퍼는 이의 다운스트림 쪽 정전용량을 업스트림 쪽에서 보이지 않게 하는, 즉, 와이어를 단절하는 효과가 있다. 다시 말하여 버퍼의 업스트림 쪽에서는 버퍼의

입력 정전용량 c_b까지만 보이게 한다. 버퍼 자체의 지연시간 D_b는 버퍼 내부지연시간(intrinsic delay)을 d_b, 버퍼 구동저항(driving resistance)을 r_b, 버퍼 다운스트림 쪽의 정전용량을 c_s 라고 할 때 다음과 같다.

$$D_b = d_b + r_b \cdot c_s$$

2. 용어 정리

매크로 블록이나 IP 블록이 놓인 영역 중 배선은 가능하나 버퍼삽입이 가능하지 않은 영역을 버퍼불가영역이라고 한다. 배선불가영역은 버퍼뿐만 아니라 배선도 허용되지 않는 영역을 칭한다.

하나의 네트 T는 소스 s_o와 한 개 이상의 터미널 $s_1, s_2, \ldots, s_n$으로 구성되어 있다. 소스 s_o는 그의 위치와 구동저항 r_s가 주어지고, 각 터미널은 그의 위치와 도달소요시간(required arrival time) q_s, 부하 정전 용량 c_s가 주어져 있다.

와이어의 단위길이 당 저항 값과 정전용량을 각각 r_0와 c_0로 표시한다. 구현의 간단함을 위해 버퍼는 한 가지 종류만 사용한다고 가정하고 이의 극성도 무시하기로 하며 버퍼의 입력 정전용량을 c_b, 출력저항을 r_b, 내부지연시간을 d_b 로 표시한다.

제안한 방법에서는 동적프로그래밍에 기반한 상향식 방법으로 부분 트리를 구성한다. 이때 어떤 하나의 부분트리는 부분해 p 로 나타낸다. 현재 구성된 트리의 *root* 노드를 v라고 하자. 노드 v에서의 도달소요시간을 q_v, v에서 다운스트림 쪽으로 보이는 정전용량을 c_v, 부분트리의 전체 정전용량을 e_v라고 할 때, p 는 다음과 같은 트리플로 표현되어 노드 v에서 유지된다.

$$p = (q_v,\ c_v,\ e_v)$$

버퍼의 삽입여부를 나타내기 위하여 트리의 각 노드 v에서는 이의 왼쪽, 오른쪽 두 자식 노드까지의 간선에 놓여지는 버퍼에 대한 정보를 리스트로 가지고 있는데 이를 각각 b_l, b_r로 표시한다. 이들 리스트에는 v에서 각 버퍼까지의 거리 값이 저장되어 있다.

각 노드에서는 포함하는 터미널들이 같은 부분해들을 함께 모아서 저장된다. 즉, 노드 v에서 부분해들은 부분해들의 집합의 집합으로 다음과 같이 표시한다.

$$P_v = \{P_1, P_2, \ldots, P_k\},\ P_i = \{p_1, \ldots, p_m\}$$

위 식에서 각 P_i의 부분해들은 모두 동일한 터미널들을 포함하고 있으며 이 터미널들을 다음과 같이 표시한다.

$$TS_{P_i} = \{s_{x1}, s_{x2}, \ldots, s_{xh}\}$$

3. 문제 정의

소스가 s_0인 네트 $T = \{s_0, s_1, s_2, \ldots, s_n\}$의 각 터미널 $s_v\ (v \neq 0)$에 도달소요시간 q_v가 주어졌다고 하자. s_0에서 터미널 s_v까지의 지연시간을 $d(s_o, s_v)$라고 하면 $q_v - d(s_o, s_v)$를 s_v에서의 여유지연시간(slack)이라고 한다. 소스 s_0에 다음과 같이 각 터미널에서의 여유지연시간 중 최소값을 q_o로 정의한다.

$$q_0 = \min_{v \in sink} \{q_v - d(s_o,\ s_v)\}$$

본 논문에서 해결하고자 하는 문제는 q_o 값이 최대가 되도록 버퍼를 적절히 삽입한 배선트리를 구하는 것이다. 추가로 배선길이와 삽입된 버퍼의 개수를 가능한 줄이는 것을 부가목적으로 한다.

Ⅲ. 버퍼 삽입을 통한 다중 터미널 네트의 배선방법

본 장에서는 버퍼삽입과 동시에 배선 트리를 구하기 위한 배선방법을 기술한다. 입력으로 주어진 배선제약조건에 대하여 배선은 배선만 가능한 버퍼불가영역을 포함하여 최단거리로 배선한 결과와 버퍼불가영역을 우회하여 버퍼를 삽입하여 얻은 배선 모두를 고려할 수 있어야 한다. 이 장에서는 이러한 두 가지 배선을 모두 고려할 수 있는 격자 그래프를 기술하고 이 격자 그래프를 이용하여 버퍼 및 배선 트리를 구하는 방법을 설명한다.

1. 격자 그래프의 구성

버퍼불가영역이 없을 경우 단일 소스, 단일 터미널 네트는 최단거리 배선이 연결지연을 최소로 한다. 그러나 버퍼불가영역이 있을 경우 이를 우회하여 최단거리로 연결하고 버퍼를 적절히 삽입한 배선의 연결지연이

보다 작을 수 있다. 본 절에서는 이러한 두 가지 배선을 얻을 수 있는 새로운 격자그래프를 보인다.

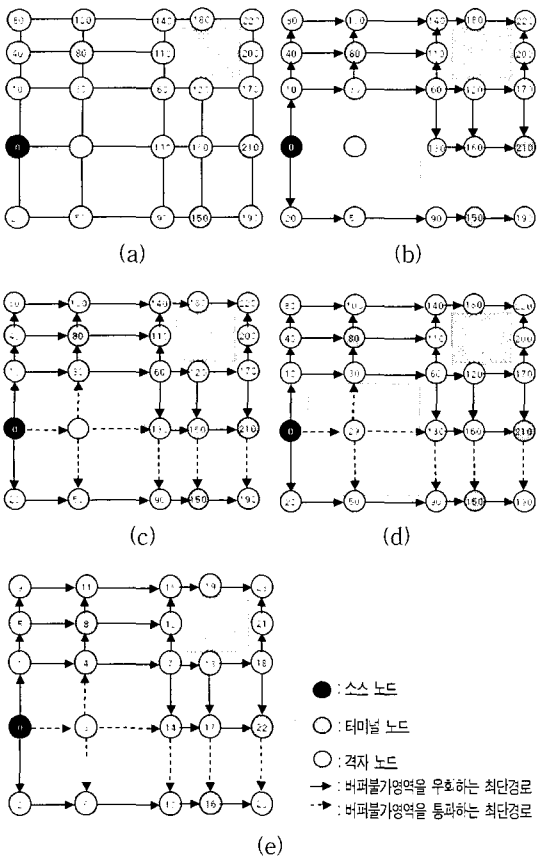


그림 2. ESPDAG 구성과정
Fig. 2. The Construction process of ESPDAG.

격자그래프를 $G=(V,E)$라고 하자. 주어진 입력에 대하여 먼저 일반 배선에서 사용되는 Hannan 그래프[2]를 구성하고 이때 생성되는 노드 중 배선불가영역의 노드를 제외한 나머지 노드들을 V에 포함시킨다.

노드 생성 후 소스에서 각 노드까지 최단거리를 계산한 후 그 거리에 따라 오름차순으로 각 노드에 순번을 정한다(<그림 2(a)>). 그림에서 순번은 10 단위로 증가하도록 하였는데 이는 현재 포함되지 않은 버퍼불가영역의 노드를 추가로 고려하기 위해서이다. 따라서 이 단위는 버퍼불가영역의 노드 수보다 큰 수로 정한다.

평면상에 이웃한 두 노드에 대하여 순번이 작은 노드에서 큰 노드로 에지(directed edge)를 생성하여 이를 E에 포함시키는데(<그림 2(b)>), 두 노드까지의 최단거리 차이가 두 노드간의 거리가 같을 경우만 에지를 생성한다.

버퍼불가영역의 노드를 V에 추가시키고 V의 서도 평면상에 서로 인접한 두 노드들에 대하여 에지가 없을 경우 소스에서 가까운 노드에서 먼 노드로 연결하는 에지를 생성하여 E에 추가한다(<그림 2(c)>).

버퍼불가영역의 노드에 대해서 소스에서 먼 노드부터 순번을 정한다. 노드의 순번은 그 노드의 자식노드의 순번 중 가장 작은 값에서 1을 뺀 값으로 한다.(<그림 2(d)>).

이제 모든 노드의 순번이 정해졌다. 노드의 순번을 순번간의 간격이 1이 되도록 다시 정한다(<그림 2(e)>).

이러한 과정을 통하여 구현한 그래프를 ESPDAG (Extended Shortest Path Directed Acyclic Graph)라고 한다. ESPDAG에서 버퍼불가영역 내부 노드들의 순번이 이 영역을 우회하는 경로를 따라 부여된 순번 사이로 결정이 되기 때문에 버퍼불가영역을 우회하는 경로와 통과하는 경로를 동시에 모두 고려할 수 있다.

ESPDAG에서는 버퍼불가영역을 우회하는 방향의 간선들과 버퍼불가영역을 무시하고 관통하는 방향의 간선들이 혼합되어 있다. 이 간선들의 방향은 모두 소스와 반대쪽을 향하며 사이클 없이 모두 연결되어 있기 때문에 간선들을 역으로 거슬러 올라갈 경우 반드시 소스와 만나게 된다. 또한, 소스에서 가까운 순으로 각 노드의 순번이 정해져있기 때문에 가장 큰 순번의 노드부터 시작하여 부분트리를 생성하여 이를 병합하며 소스 쪽으로 향하면 소스 노드에서 모든 터미널을 포함하는 모든 가능한 트리를 구성할 수 있다.

2. 부분해 파생 및 갱신

먼저 각 터미널 노드에 대하여 초기 부분해 $(q_s,\ c_s\ c_s)$를 생성한다. q_s와 c_s는 각각 입력으로 주어지는 터미널의 도달소요시간과 부하 정전 용량이다. 이렇게 생성된 부분해는 터미널 노드 하나로 이루어진 부분 트리를 의미한다.

현재 노드 u 에서 자식 노드 v 를 루트로 하는 부분트리를 u 로 확장한다는 것은 자식 노드 v 의 모든 부분해들을 간선 (u,v) 에 대한 버퍼 삽입을 고려하여 u 에서의 부분해들을 구하고 이들에 대한 q_u, c_u 그리고 e_u 를 새롭게 계산한다는 것이다.

본 논문에서는 참고문헌 [1]의 수식에 근거하여 노드

u 와 v 사이의 버퍼위치와 개수를 계산한다. r_s 를 구동 저항으로 갖는 소스와 c_s 의 부하정전용량을 갖는 터미널로 이루어진 길이 l인 와이어에 대하여 연결지연을 최소로 하기위한 버퍼의 개수 k와 위치 x, y 의 값들은 다음의 식으로 구할 수 있다.

(1) 버퍼의 개수

$$k = \lfloor -\frac{1}{2} + \frac{1}{2}\sqrt{1+\frac{2(r_0 \cdot c_0 \cdot l - r_0(c_b - c_s) - c_0(r_b - r_s))^2}{r_0 \cdot c_0(r_b \cdot c_b + d_b)}} \rfloor$$

(2) 소스로부터 첫 번째 버퍼까지의 거리

$$x = \frac{1}{k+1}(l + \frac{k(r_b - r_s)}{r_0} + \frac{c_s + c_b}{c_0})$$

(3) 버퍼와 버퍼 사이의 거리

$$y = \frac{1}{k+1}(l - \frac{r_b - r_s}{r_0} + \frac{c_s + c_b}{c_0})$$

여기서, k 개의 버퍼가 삽입될 때 이 버퍼들은 균일한 간격으로 놓여지는 것이 최적이며[1], 따라서 버퍼들 소스에서 거리 x 만큼 떨어진 후 y 의 균일한 간격으로 놓여진다.

참고문헌 [1]에서 제안된 WSA는 다중터미널에 대한 트리 형태가 주어졌을 때 연결지연이 작도록 버퍼를 삽입하는 방법인데, 위의 수식에 의하여 각 와이어 세그먼트를 일단 k 등분하여 내부노드를 추가한 후, 이 내부 노드와 스타이너 노드들에서의 버퍼 삽입여부를 Ginneken의 기법을 사용하여 결정한다.

WSA에서는 하나의 와이어 (u, v)를 나누기 위하여 앞의 식을 적용할 때 노드 v를 루트로 하는 다운스트림 쪽의 부분트리에 버퍼가 없다고 가정하고 계산한 정전용량(두 터미널 네트의 경우 버퍼입력의 정전용량)을 c_s로 사용하고 소스의 구동저항을 r_s 로 사용하기 때문에 그 정확도가 높지 않으며 각 와이어를 여러 개의 작은 와이어로 나누기 때문에 대단히 많은 수의 부분해가 만들어져 이를 처리하여야 하는 비효율적인 면이 있다.

본 논문에서는 WSA와는 달리 부분해를 생성하는 과정에서 와이어를 필요에 따라 분절하기 때문에 좀더 정확한 계산을 할 수 있다. 현재 와이어 (u, v)에 버퍼 삽입을 고려한다고 하자. 노드 u 가 소스쪽 노드이고 v 는 다운스트림쪽 노드로 u의 유일한 자식노드라고 가정하자. 노드 u 까지 트리를 확장할 때 v에는 v를 루트로 하는 부분해의 집합을 유지하고 있는데 이들 각각은 노드 v에서 다운스트림 쪽으로 보이는 정전용량 c_v을 갖고 있어 이 값을 c_s 로 사용할 수 있다. 그리고 노드 u에는 소스 s_0 까지의 최단거리 i_u 값이 격자 그래프 구성 과정 중에 구해진다. 노드 u에서 소스 s_0 까지의 경로 중 처음으로 만나게 될 버퍼의 위치를 z $(0 \le z < i_u)$라고 가정하면 이 z 값을 이용하여 업스트림의 저항값 r_{up}를 다음과 같이 계산하여 앞쪽의 식 적용을 위하여 필요한 r_s 로 사용할 수 있다.

$$r_{up} = r_b + z \cdot r_e$$

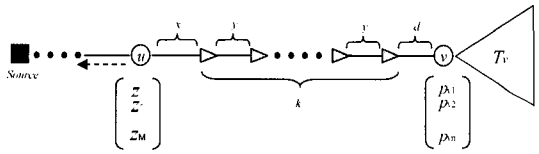


그림 3. 하나의 내부 와이어에 대한 버퍼삽입.
Fig. 3. Buffer insertion for a single internal wire.

이 때 부분해를 생성하는 과정에서는 z 값을 정확히 알 수 없으므로 각 부분해별로 $0 \le z < i_u$ 범위에서 M개를 선택하여 계산한다.

<그림 3>에 노드 v 에서의 부분해와 와이어 (u, v)에서의 버퍼삽입 여부 등을 결합하여 노드 u 에서의 부분해를 만드는 사항을 설명한다. 노드 v 에서의 각 부분해에 대하여 이의 다운스트림 쪽의 정전용량과 앞에서 설명한 업스트림쪽의 저항값을 사용하여 와이어 (u, v)에 삽입할 버퍼의 개수와 위치를 구하고 이에 따라서 노드 u 에서의 부분해를 구하여 u 에 저장한다. 이때 M개의 업스트림 저항값을 고려하므로 M개의 부분해가 생성될 수 있지만 계산한 k, x, y 값의 중복이 심하므로 무시할 수 있어 새로 생성되는 부분해는 많지 않다. 하나의 부분해가 생성되면 이때 와이어 (u, v)에 삽입된 버퍼들은 b_l에 리스트로 저장된다.

간선이 배선불가영역에 있을 경우에는 버퍼삽입 없이 와이어에 의한 지연시간과 정전용량을 추가하여 부분해를 생성한다. 노드 u 의 자식노드가 하나 이상일

경우에는 3.3에서 트리의 확장과정에 관한 설명 중에 기술하도록 한다.

3. MRB

본 절에서는 3.1에서 기술한 격자그래프와 3.2에서 논한 부분해의 확장을 바탕으로 동적 프로그래밍 기법에 의하여 격자그래프의 각 노드들을 탐색하면서 버퍼삽입을 고려하고 부분해를 생성 또는 제거하여 버퍼삽입 및 배선을 동시에 수행하는 MRB(Multiterminal Routing with Buffer Insertion) 방법에 대하여 기술한다.

MRB에서는 순번이 가장 큰 노드부터 작은 순으로 소스노드까지 방문하면서 부분트리의 생성 및 제거과정을 수행한다. 이때 각 노드에서는 자식 노드들을 루트로 하는 부분트리들을 그들이 포함하는 터미널 집합 TS를 기준으로 병합하여 현재 노드를 루트로 하는 새로운 부분 트리들을 생성한다. 또한, 병합 없이 현재 노드까지 루트가 확장되는 부분트리에 대해서도 이들의 정보를 유지한다. 이런 과정을 통해 ESPDAG 상에서 구성될 수 있는 모든 가능한 트리구조를 구성하여 최적의 buffered 배선트리를 구하게 된다.

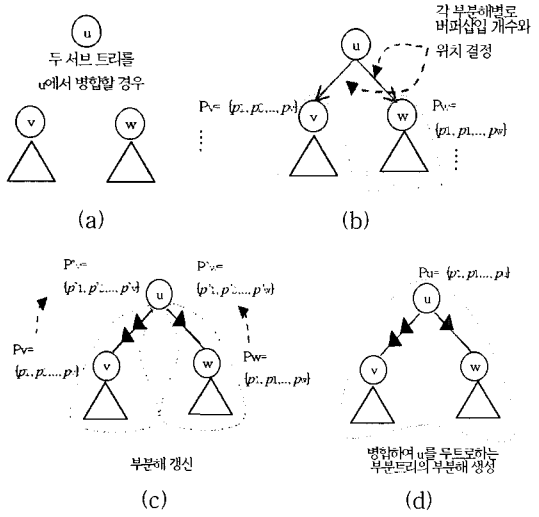


그림 4. 부분해의 확장과 부분트리의 병합
Fig. 4. Subtree extension and merging.

3.1. 버퍼불가영역을 고려한 배선트리 생성

전체 그래프를 탐색하면서 트리형태를 구성하는 방법은 RSA/G[5]와 유사하다. RSA/G는 장애물이 있을 때 MSPSA (Minimum Shortest Path Steiner Arborescence)를 구하는 방법이다.

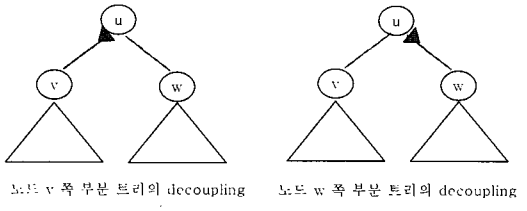


그림 5. 디커플링 버퍼의 삽입
Fig. 5. Decoupling buffer insertion.

MRB에서는 ESPDAG 상에서 간선을 따라 트리를 구성되는데 하나의 노드 u에서 u의 자식노드의 부분해들을 병합하여 u에서의 부분해를 형성하는 과정을 먼저 보인다.

<그림 4(a)>와 같이 노드 u의 두 자식 노드를 v와 w라고 하자. 우선 3.2절에서 기술한 방법으로 v과 w 각각의 부분해 집합의 모든 부분해에 대하여 노드 u까지의 간선에서의 버퍼삽입여부를 결정하고(<그림 4(b)>) 이 간선을 포함한 부분트리로 갱신하여 부분해 집합을 임시로 구성한다(<그림 4(c)>). 다음, 이렇게 구성한 v와 w의 부분해 집합 P_v'와 P_w'가 터미널을 공유하지 않으면(즉, $TS_{P_v'} \cap TS_{P_v'} = \emptyset$) 이들 두 집합의 부분해들을 병합하여 노드 u를 루트로 하는 부분해 집합을 구한다(<그림 4(d)>).

병합된 부분해의 도달소요시간 q_u는 도달소요시간이 작은 부분해의 도달소요시간으로 하여야 한다. 따라서 도달소요시간이 작은 부분해를 최적화 할 필요가 있다. 따라서 병합에 사용한 다른 부분해의 정전용량에 의해 연결지연이 증가되는 것을 막기 위하여 병합되는 곳에서 버퍼를 추가할 수 있다(decoupling)[4, 9].

이를 위해 위의 병합된 부분해 외에도 각 자식노드의 부분해에 <그림 5>과 같이 디커플링 버퍼를 삽입한 추가적인 부분해를 생성하여 이들을 병합에 사용하도록 한다.

갱신된 부분해 집합 P_v' 와 P_w'에 대해 그 원소인 부분해에 대한 병합은 다음과 같이 이루어진다. P_v'와 P_w'의 부분해 하나씩을 각각 p_v, p_w라고 하면 병합되어 새로이 생성되는 u의 새로운 부분해 p_u 는

$$p_u = (\min(q_v, q_w),\ c_v + c_w,\ e_v + e_w)$$

이다. 생성된 부분해 p_u 는 $TS_{p_u} = TS_{p_l} \cup TS_{p_r}$ 을 인덱스로 하는 부분해 집합 P_u 에 포함시키는데 p_l 와 p_{ur} 의 b_{rl}, b_{rr} 을 각각 p_u의 b_{ul}, b_{ur} 로 유지하게 된다.

이러한 방법으로 노드 u의 모든 자식노드들의 모든 부분해 집합의 부분해를 다른 자식노드의 부분해집합의 부분해와 병합을 수행하여 노드 u의 부분해 집합을 완성한다. 그리고 다른 부분해와 병합하지 않고 단순히 노드 u까지 확장한 부분해들을 u의 부분해 집합으로 추가시킨다.

3.2. 부분해의 제거

앞에서 기술한 방법으로 부분해들을 병합하면 대단히 많은 수의 부분해들이 생성될 수 있다. 그러나 병합을 계속하여 소스 노드에 도착하여도 최적해에 도달할 가능성이 없는 많은 부분해들이 있는데 각 노드에서의 병합 후 이들을 제거(pruning)하여 수행시간을 절약할 필요가 있다.

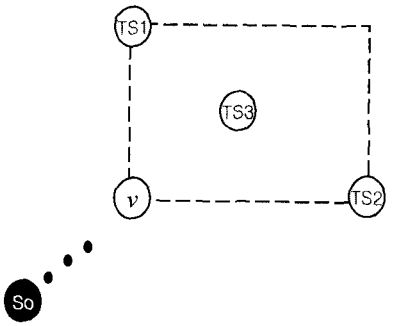


그림 6. 제한되는 부분해 집합의 병합
Fig. 6. A restricted solution merge.

두 부분해 p_1과 p_2가 있을 때

$$q_1 > q_2 \quad \text{이고} \quad c_1 \leq c_2$$

이면 p_2 는 p_1 보다 큰 q 값을 구성할 수 없고 따라서 더 이상의 부분해의 병합 또는 확장에서 p_2를 제외한다[6]. 그런데 이러한 조건에 의한 부분해의 제거 외에도 지연시간과 가능한 와이어 길이의 상관관계를 고려하여 부분해를 추가로 제거할 수 있는데 본 절의 나머지 부분에서는 이에 대하여 기술한다.

하나의 부분해 집합의 각 부분해들은 서로 동일한 터미널들을 포함하고 있다. 이들 터미널들을 포함한 bounding box(BB)의 half perimeter(HP)는 배선의 질을 구분하는 척도로 널리 사용되고 있다. 만일 부분해 하나가 그 전체 정전용량이 HP의 정전용량보다 훨씬 크다면 그 부분해는 같은 집합의 다른 부분해 보다 그 질이 나쁜 부분해일 가능성이 높다. 따라서, 하나의 노드에서 부분해 집합들을 갱신한 후, 각 부분해의 정전용량이 이의 HP 보다 매우 클 경우 이들을 부분해 집합에서 제거한다. 이는 limit_t라는 파라메터로 조정된다.

각 부분해집합의 부분해 중 도달소요시간이 유사한 부분해들을 모두 유지할 필요가 없다. 도달소요시간이 유사한 부분해들을 선별하고 이중 가장 작은 전체정전용량을 갖는 것만 유지하고 나머지는 버리도록 한다. 어느 정도의 부분해들을 유사부분해로 간주하느냐(adm_sol %이내)에 따라 최종해의 질과 전체수행시간을 조정할 수 있다.

소스노드에서 최종 해를 선택할 때는 최소 도달소요시간을 갖는 해보다 accept_req %정도 값이 큰 해 중에서 전체정전용량이 가장 작은 해를 선택한다.

마지막으로 두 개의 부분해를 병합할 때 그들이 포함하고 있는 터미널들이 속한 BB에 이들 터미널에 속하지 않는 터미널이 존재하는지 조사한다(<그림 6>에서 TS1과 TS2를 병합하고자 할 때 TS3). 만일 이러한 터미널이 존재한다면 이는 다른 부분해에 속한 것이고 이 부분해의 와이어는 병합된 부분해의 와이어와 서로 교차할 가능성이 있다. 그러나 이러한 동일 네트의 와이어의 교차는 배선길이를 길게 하고, 불필요한 congestion을 유발할 가능성이 있음으로 이러한 병합을 하지 않는다. 즉, BB내에 다른 터미널이 존재하지 않을 경우에만 병합을 수행한다. 이는 지리적으로 근접한 터미널들을 와이어의 교차가 없는 적절한 병합을 얻는데 효과적이다.

Ⅳ. 실험 결과

본 논문에서 제안하는 MRB 방법은 Sun Ultra-sparc 6.0 워크스테이션에서 C언어로 구현되었다. <표 1>과 <표 2>는 실험에 사용된 테크놀로지 파라미터와 네트 정보를 보여주고 있다. 이들은 참고문헌 [9]에서 사용한 자료와 동일하다. 모든 네트는 하나의 소스와 한 개 이상의 다중 터미널로 이루어져 있다. <표 2>에서 #block은 버퍼불가영역의 개수를 말한다. 저항값 추

표 1. 실험에 사용된 테크놀로지 파라미터
Table 1. Technology parameter.

unit wire resistance(Ω/㎛)	0.124
unit wire capacitance(fF/㎛)	0.143
buffer driving resistance(Ω/㎛)	474.565
buffer loading capacitance(fF/㎛)	11.92
buffer intrinsic delay(ns)	0.075

표 2. 실험에 사용한 네트
Table 2. Net Informations.

Net	#pin	#block
n071	8	4
m0s5	8	2
n313	9	4
n730	9	2
pnt3	10	4
m1s9	10	4
n905	11	2
n702	11	4
n866	12	4

limit_t = 25 % *adm_sol* = 0.5 %
accept_req = 3 %

표 3. WSA와 MRB의 결과
Table 3. Results of WSA and MRB.

Net		slack(ps)			# buffers		
test	#pin	WSA	MRB	%	WSA	MRB	+/-
n071	8	601.3	595.2	-0.1	5	5	0
m0s5	8	475.4	515.4	8.4	4	4	0
n313	9	591.2	590.8	-0.1	3	3	0
n730	9	1009	994.4	-1.4	3	2	-1
pnt3	10	1024.3	1023.6	-0.1	4	3	-1
m1s9	10	894.9	894.5	0.0	3	3	0
n905	11	913	910.6	-0.3	4	3	-1
n702	11	231	231	0.0	1	1	0
n866	12	561.7	564.5	0.5	3	3	0

출회수 M=10으로 하였고, 앞장에서 기술한 각 노드에서의 부분해 집합의 제한에 대한 파라미터 결정은 다음과 같다.

1. WSA 알고리즘과의 비교

우선 본 논문에서 사용하고 있는 버퍼 삽입의 타당성을 확인하기위해 참고문헌 [1]에서 제안된 알고리즘 WSA를 각 와이어 세그먼트의 분절 수 k 값을 참고문헌 [1]에서 보다 30배 크게 하여 적용하고 그 결과를 우리의 결과와 비교하여 보았다. k 값을 30 배 한다는 것은 와이어 임의의 위치에 버퍼를 삽입할 수 있다는 것을 의미하고 따라서 주어진 배선 트리에 대하여 거의 최적의 버퍼 삽입결과를 얻을 수 있다. WSA에서는 입력으로 배선트리가 주어져야 하는데 이를 위해 우리의 MRB 방법에 의하여 생성된 트리에서 버퍼를 제거한 트리를 입력하였다. 따라서 이 비교는 MRB 방법이 버퍼를 얼마나 효과적으로 삽입되는지를 확인하는 것이라고 할 수 있다.

<표 3>에 그 비교결과를 보인다. 대부분의 회로에서 여유지연시간의 차이가 1% 이내이고 버퍼의 개수도 거의 차이가 없음을 알 수 있다. 이것은 본 논문에서 사용한 버퍼 삽입 알고리즘이 최적의 해와 크게 차이가 없음을 보이고 있다. 모든 결과에서 삽입한 버퍼의 수가 같거나 하나 작은데 이는 부분트리의 분기점에서 도달 소요시간이 큰 쪽의 정전용량을 디커플링 버퍼로 배제시키는 방법에 의한 효과로 볼 수 있다.

2. 버퍼삽입과 동시에 트리구성 방법과의 비교

버퍼삽입과 동시에 배선트리를 구성하는 방법으로는 RMP 방법과[6]과 RIATA 방법[9] 등을 들 수 있다. RMP알고리즘은 버퍼 삽입이 가능한 위치가 고정된 경우 버퍼를 삽입한 배선트리를 얻는 방법이나, 실험에서는 버퍼불가지역을 제외한 모든 다른 영역에서 버퍼삽입이 가능하다고 하고 RMP를 수행하였다.

<표 4> 및 <표 5>에 실험결과를 보인다. <표 4>에서 여유지연시간을 보면 MRB, RMP, RIATA의 순으

표 4. RIATA과 RMP 알고리즘과의 여유지연시간과 버퍼 개수 비교
Table 4. A Comparison slack and number of buffer with RIATA and RMP.

Net		Slack(ps)					# buffers				
test	#p	RIATA	RMP	MRB	% vs RIATA	% vs RMP	RIATA	RMP	MRB	# vs RIATA	# vs RMP
n071	8	467	539	595.2	27.5	10.4	3	5	5	-2	0
m0s5	8	357	365	515.4	44.4	41.2	3	5	4	-1	1
n313	9	469	529	590.8	26.0	11.7	3	5	3	0	2
n730	9	961	738	994.4	3.5	34.7	3	3	2	1	1
pnt3	10	939	1011	1023.6	9.0	1.2	5	6	3	2	3
m1s9	10	747	818	894.5	19.7	9.4	3	6	3	0	3
n905	11	783	843	910.6	16.3	8.0	3	4	3	0	1
n702	11	151	202	231	53.0	14.4	2	3	1	1	2
n866	12	477	523	564.5	18.3	7.9	4	7	3	1	4

로 크다는 것을 알 수 있다. MRB에 의한 결과를 RIATA 와 RMP의 결과와 비교하면 각각 평균 24.2%와 15.4% 여유지연시간이 향상되었고 이를 통하여 우리의 MRB가 다른 방법에 비하여 지연시간의 감소에 매우 우수하다는 것을 알 수 있다.

버퍼도 다른 두 알고리즘에 비해 적은 수가 삽입된다. RIATA의 경우 평균 3.2개, RMP의 경우 4.89개의 버퍼가 삽입되지만 MRB의 경우 3개로 그 수가 작았다.

<표 5>는 MRB에 의한 결과의 전체 와이어 길이가 RIATA에 비하여 평균 10.4% 증가하고, RMP에 비하여 평균 17.8%의 감소됨을 보이고 있다. RIATA에 비해 와이어 길이가 증가하는 이유는 소요도달시간이 작

표 5. RIATA과 RMP 알고리즘과의 와이어 길이와 수행시간 비교

Table 5. A Comparison wire length and CPU time with RIATA and RMP.

Net		wire length					CPU time (sec)		
test	#p	RIATA	RMP	MRB	% vs RIATA	% vs RMP	RIATA	RMP	MRB
n071	8	5868	6422	7368	25.6	14.7	0.06	10.5	0.42
m0s5	8	7043	10207	8254	17.2	19.1	0.03	8.07	0.59
n313	9	5860	10876	8118	38.5	25.4	0.09	0.79	0.09
n730	9	2348	2491	2351	0.1	5.6	0.04	19.6	0.65
pnt3	10	7250	15093	5643	22.2	62.6	0.09	11.7	0.67
m1s9	10	4608	6554	7449	61.7	13.7	0.05	283	2.70
n905	11	3026	3379	2716	10.2	19.6	0.09	109	4.23
n702	11	3583	4294	3469	3.2	19.2	0.07	2645	2.29
n866	12	7730	10587	6631	14.2	37.4	0.09	1083	3.61

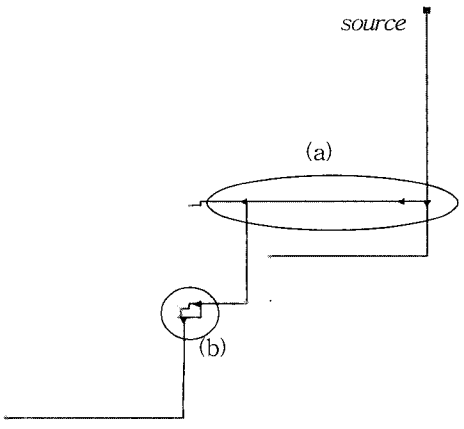


그림 7. Net n071의 배선결과.

Fig. 7. A Routinf result of Net n071.

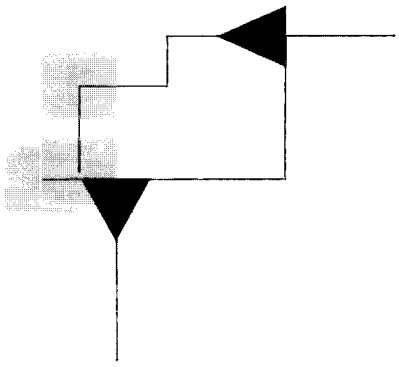


그림 8. <그림 7>의 (b)영역

Fig. 8. A area (b) of fig 7.

은 부분트리의 지연시간을 줄이기 위해 그 부분트리를 고립화(isolation)하는 방향으로 전체 트리가 구성되었기 때문이다.

<그림 7>에 n071의 배선 결과를 보인다. (a)영역의 배선을 보면 버퍼를 삽입할 수 있도록 버퍼불가영역을 피하면서 배선되었음을 보이고 있다. 그리고 <그림 7(b)>를 확대한 <그림 8>을 보면 세 개의 터미널 중 가장 왼쪽하단의 터미널의 연결지연시간을 줄이기 위해 분기되는 스타이너 점에서 각각 다른 부분트리를 디커플링 시키고 있음을 알 수 있다. 수행시간은 RIATA 알고리즘에 비해 큰 편이지만 RMP와 비교해 볼 때 상당하게 감소한다. 그러므로 본 논문에서 제안하고 있는 방법은 수행시간 측면에서 약간의 손해를 보더라도 최소의 버퍼수와 전체 트리의 와이어 길이를 갖으면서 최적에 가까운 여유지연시간을 얻을 수 있는 성능을 보였다.

V. 결 론

본 논문에서는 배선 트리의 형태를 구성해나가면 서 동시에 버퍼를 삽입할 수 있는 방법을 제안하였다. 제안한 방법에서는 새로운 격자 그래프의 구성을 통해 동적 프로그래밍을 이용하여 배선 트리 형태를 생성하고 보다 정확한 저항값의 추출, 디커플링 버퍼의 삽입, 그리고 휴리스틱을 이용한 보다 많은 부분해의 제거 등을 통하여 비교적 합리적인 시간에 적은 수의 버퍼와 작은 와이어 길이를 가지면서 여유지연시간의 큰 향상을 보이는 배선트리를 구성할 수 있다.

참 고 문 헌

[1] C. J. Alpert and A. Devgan, "Wire Segmenting for Improved Buffer Insertion," Proc. of DAC, 1997, pp. 588-593.

[2] C. J. Alpert, et. al., "Steiner Tree Optimization for Buffers, Blockages, and Bays," IEEE TCAD, Vol. 20, No. 4, pp. 556-562, Apr. 2001.

[3] C. J. Alpert, et. al., "Buffered Steiner Trees for Difficult Instances," Proc. of ISPD, 2001, pp. 4-9.

[4] C. J. Alpert, et. al., "A Practical Metho- dology for Early Buffer and Wire Resource Allocation," Proc. of DAC, 2001, pp 189-194.

[5] J. Cong, et. al., "Efficient Algorithms for the Minimum Shortest Path Steiner Arbore- scence Problem with Applications to VLSI Physical Design," IEEE TCAD, Vol. 17, No. 1, pp. 24-39. January 1998.

[6] J. Cong and X. Yuan, "Routing Tree Construction Under Fixed Buffer Location," Proc. of DAC, 2000, pp. 379-384.

[7] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifier," J. Applied Physics, pp. 55-63, 1948.

[8] L. P.P.P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmtore Delay," Proc. ISCAS, 1990, pp. 865-868.

[9] J. Hu, et. al., "Buffer Insertion with Adap- tive Blockage Avoidance," Proc. ISPD, 2002, pp. 92-97.

[10] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization," Proc. of ICCAD, 1996, pp. 44-49.

[11] H. Zhou, et. al., "Simultaneous Routing and Buffer Insertion with Restricts on Buffer Locations," IEEE TCAD, Vol. 19, No. 7, pp. 819-824. July 2000.

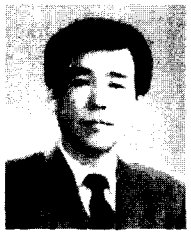
저 자 소 개

鄭 東 植(正會員)

1998년 : 서강대학교 컴퓨터학과 학사. 2002년 : 서강대학교 컴퓨터학과 석사. 2002년 8월~현재 : 삼성전자 DS총괄 근무.

金 德 歡(學生會員)

2003년 : 서강대학교 컴퓨터학과 학사. 2002년 3월~현재 : 서강대학교 컴퓨터학과 대학원 재학중



林 鍾 錫(正會員)

1981년 : 서강대학교 전자공학과 학사. 1983년 : 한국과학기술원 전기 및 전자공학과 석사. 1989년 : Univ. of Maryland, College Park, 전기공학과 박사. 1983년 3월~1990년 8월 : 한국전자통신연구소 연구원. 1990년 9월~현재 : 서강대학교 컴퓨터학과 교수.