論文98-35C-9-3

회로 결선도 분할을 위해 점진적 병합을 이용한 선형배열

(Linear Ordering with Incremental Merging for Circuit Netlist Partitioning)

成 桄 洙 *

(Kwang-Su Seong)

요 약

본 논문에서는 회로결선도 분할을 위해 LIME이라는 효과적인 선형배열 알고리즘을 제안한다. LIME은 제안된 비용함수를 이용해 하나의 세그먼트가 남을 때까지 두 개의 세그먼트를 병합한다. 마지막에 남은 하나의 세그먼트가 선형배열에 해당한다. LIME은 회로 결선도의 성긴 특징을 이용하므로 상당히 빠르게 수행된다. 제안된 알고리즘은 기존 방법보다 선형배열을 만드는데 약 8배 빠른 수행 속도를 보이며, 이를 이용한 회로 결선도 분할 결과도 스케일드 비용 면에서 약 17% 향상되었다.

Abstract

In this paper, we propose an efficient linear ordering algorithm, called LIME, for netlist partitioning. LIME incrementally merges two segments which are selected based on the proposed cost function until only one segment remains. The final resultant segment then corresponds to the linear ordering. LIME also runs extremely fast, because it exploits sparsity of netlist. Compared to the earlier work, the proposed algorithm is eight times faster in producing linear ordering and yields an average of 17% improvement for the multi-way scaled cost partitioning.

Ⅰ. 개 요

과거 수년간 시스템 및 칩의 복잡도가 현저하게 증가하여 회로결선도 분할 문제는 VLSI/CAD분야에서 그 중요성이 더욱 부각되고 있다. 이 문제는 시스템 혹은 큰 블록의 회로를 현재 존재하는 CAD 툴이나 공정기술로 쉽게 다룰 수 있는 크기로 분할하는 것을 말한다. 따라서 좋은 분할 결과는 복잡도를 줄일 뿐 아니라 전체 시스템(혹은 칩)의 성능 향상에 큰 영향을 미친다. 특히 CAD 분야에서 회로 결선도 분할은 회로 배치(placement), 병렬 시뮬레이션(parallel simulation), 로직 이뮬레이션(logic emulation) 등에서 응용되고 있다.

일반적으로 n개의 노드 $\{v_1, v_2, ..., v_n\}$로 구성된 회로 결선도는 하이퍼그래프(hypergraph) $H(V,E)$로 표현된다. 회로결선도 분할 문제는 주어진 n개의 노드를 중첩되지 않도록 k개의 클러스터 $\{C_1, C_2, .. C_k\}$로 분할하는 것을 말하다. 여기서 $|C_i|$는 클러스터 C_i에 있는 노드들의 수, 즉 클러스터의 크기를 나타내고, E_i는 클러스터 C_i의 경계를 통과하는 네트 개수라 정의한다. 결선도 분할에서 클러스터들이 크기에 대해 균형을 이루고, 클러스터링에 의해 잘리는 네트의 수를 최소화하는 것이 좋은 해가 된다. 이와 같은 두 가지 요소를 고려한 비용함수 중 하나가 스케일드 비용함수[1]로 아래와 같이 정의된다.

* 正會員, 嶺南大學校 電氣電子工學部
(YeungNam University Electrical Engineering Department)

接受日字:1998年5月28日, 수정완료일:1998年8月27日

$$F_{scaled}(P_n^k) = \frac{1}{n(k-1)} \sum_{i=1}^{k} \frac{E_i}{|C_i|} \quad (1)$$

이 비용함수는 위에서 설명한 두 가지 요소를 하나의 수식에 포함시킨 것이 특징이다. k = 2이면 (1)식은 다음과 같이 표현된다.

$$F_{scaled}(P_n^2) = \frac{E_1}{|C_1||C_2|} = \frac{E_1}{|C_1||n-|C_1||} \quad (2)$$

(2)식에서 보는바와 같이 분모는 클러스터들의 크기에 대한 균형을 잡는 요소로 이용되고, 분자는 클러스터링에 의해 잘리는 네트의 수가 최소가 되도록 하는 특징을 가지고 있다. 따라서 (1)식을 최소로 하는 클러스터링을 스케일드 비용함수 면에서 최적해라고 정의한다. 이 비용함수는 클러스터 크기를 인위적으로 제한하지 않을 뿐 아니라 선형배열의 질을 측정하는 척도로 기존에 많이 사용되었으므로 본 논문에서 선택하였다.

회로 결선도 분할에 대해 좋은 결과를 얻기 위해 이동을 기초로 한 접근 방법(move-based approach)[2,3,4], 스펙트럴 방법(spectral method)[1,5,6,7,8] 그리고 클러스터링을 기초로 한 접근 방법(clustering-based approach)[1,2,9,10] 등이 제안되었다. 이동을 기초로 한 회로 결선도 분할은 어떤 임의의 초기 해에서 인접하는 가장 좋은 해를 얻는 것으로 이를 반복적으로 수행하여 점차적으로 좋은 해를 얻는 방법이다. 그러나 이 방법은 국소 최적해(local optimum solution)를 구하는 경우가 많아 이를 극복하기 위해 클러스터링과 같은 여러 방법이 제안되었다.[2,9,10]

스펙트럴 방법은 회로 결선도 H를 그래프 G로 변환한 후 그래프 대한 고유벡터와 고유치를 이용해 그래프의 각 노드를 d차원의 공간에 사상(mapping)시킨 다음 이를 분할하여 그에 대응하는 회로 분할을 얻는다. 이 분야에는 최근 많은 진전이 있었으며, 특히 선형 배열을 이용한 회로결선도 분할이 많이 연구되었다[1,2,5,6,7]. 그 이유는 주어진 선형 배열에 대해 최적해를 구할 수 있는 회로결선도 분할 알고리즘이 제안되었기 때문이다.[7] 즉, 질이 좋은 선형배열을 구하면 이를 이용해 좋은 결과의 회로 결선도 분할을 할 수 있으므로 회로결선도 분할 문제를 선형배열 문제로 바꿀 수 있다.

SFC[7]에서는 그래프의 각 노드를 d개의 고유벡터를 이용해 d차원 공간에 사상시킨 후 Traveling Salesman 문제의 휴리스틱인 SFC(Space Filling Curve)를 이용해 선형 배열을 구하였다. MELO[5]에서는 d개의 고유벡터와 d개의 고유치를 이용해 각 노드들을 d차원 공간에 사상하였다. 사상된 각각의 점들은 원점을 시점으로 하는 벡터로 볼 수 있으며, 이들 벡터를 분할하여 그에 대응하는 그래프 분할을 얻는 벡터분할이 제안되었다. 또한 이 방법을 제안한 Alpert와 Yao는 그래프분할과 벡터분할 사이에 정확한 관계가 있음을 보였다. 즉, 최적의 벡터분할을 구하면 그에 대한 최적의 그래프 분할을 얻을 수 있음을 말한다. MELO에서는 이 관계를 이용해 선형배열을 얻었다. SFC와 MELO에서 구해진 선형배열은 DP-RP[7]라는 알고리즘을 이용해 k개의 클러스터로 분할되었다. DP-RP는 주어진 선형배열에 대해 스케일드 비용함수 면에서 최적의 해를 구하는 알고리즘이다. 그러나 기존 방법들[5,6,7]은 선형배열을 구할 때 회로분할의 비용함수가 스케일드라는 점을 완전히 이용하지 못하였다. 즉, 클러스터의 크기에 대한 요소를 고려하지 않았다.

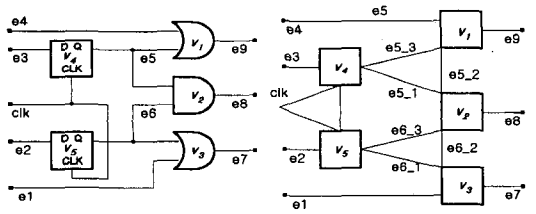
본 논문에서는 새로운 선형배열 알고리즘을 제안한다. 이 방법에서는 선형배열의 임의의 연속적인 구간을 세그먼트로 정의한다. 따라서 각 노드 역시 하나의 세그먼트가 된다. 본 알고리즘은 제안된 비용함수를 이용해 두 개의 세그먼트를 선택한 후 하나의 새로운 세그먼트를 만든다. 이런 과정을 계속하면 마지막에 하나의 세그먼트가 만들어지며, 이것이 바로 구하고자 하는 선형배열이 된다. 두 개의 세그먼트를 선택하는 비용함수는 스케일드 비용함수를 고려한 것으로, 주어진 q개의 세그먼트에서 두 개의 세그먼트를 병합하여 $(q-1)$개의 세그먼트를 만들 때, 스케일드 비용 관점에서 최적해가 되도록 두 개의 세그먼트를 선택한다. 제안된 알고리즘을 효과적으로 구현하기 위해 회로결선도의 성긴 특징(sparsity)을 이용하여 기존 선형배열 알고리즘 MELO 보다 8배 속도향상 되었다. 제안된 알고리즘의 성능을 알아보기 위해 구한 선형배열은 DP-RP를 이용해 k개의 클러스터로 분할하여 기존 방법 MELO에 비해 스케일드 비용함수 면에서 약 17% 성능이 향상되었다.

본 논문은 다음과 같이 구성되어 있다. 제 II장에는 스펙트럴 방법에 관한 내용이 있으며, III장에서는 제안된 알고리즘에 관해 설명하며, IV장에서는 실험 결과를 보이고 있다.

Ⅱ. 스펙트럴 방법

1. 회로결선도의 그래프 변환

스펙트럴 방법을 이용해 회로 결선도를 분할하기 위해서는 주어진 회로 결선도를 그래프로 변환해야한다. 하이퍼 그래프로 표현되는 회로 결선도를 정확하게 그래프로 변환하는 방법이 없으므로, 클리크(clique)모델을 이용해 근사적으로 변환한다.[1,2,5] 그림 1은 클리크 모델을 이용해 회로 결선도를 그래프로 변환시킨 예제이다. 네트 6 (e6)은 module v_2, v_3 그리고 v_5에 연결되어 있다. 이에 대한 클리크는 그림 1(b)의 e6_1, e6_2 그리고 e6_3이다.



(a) (b)

그림 1. 주어진 회로 결선도를 클리크 모델을 이용해 그래프로 변환
(a) 결선도 (b) 클리크 모델을 이용해 변환된 그래프

Fig. 1. Transform a given netlist into the graph by using clique model.
(a) Netlist (b) transformed graph using clique model

변환된 그래프 $G(V,E)$는 노드집합 $V = \{v_1, v_2, ..., v_n\}$와 에지 집합 $E = \{e_1, e_2, ..., e_m\}$로 구성되며, 이에 대해 다음과 같이 adjacency 행렬 A, degree 행렬 D 그리고 Laplacian 행렬 $Q(=D-A)$를 정의할 수 있다. 대칭인 adjacency 행렬은 $A = (a_{ij})$로 표현이 된다. 이때 a_{ij}는 두 노드 v_i와 v_j 사이의 에지 값이다. 그리고 diagonal 행렬인 degree 행렬는 $D = (d_{ii})$로 표현된다. 여기서 $d_{ii}(=\deg(i)=\sum_{j=1}^{n} a_{ij})$는 노드 v_i의 degree를 나타낸다. 그래프 분할문제는 n개의 노드들을 중첩되지 않게 k개의 클러스터로 나누는 것을 의미한다. 이때 나뉘어진 그래프에 대한 스케일드 비용함수 $F_{scaled}(P_g^k)$는 식 (1)과 같지만 E_i의 의미는 다음과 같다.

$$E_i = \sum_{v_k \notin C_i} \sum_{v_l \in C_i} a_{kl} \qquad (3)$$

여기서 E_i의 의미는 클러스트 C_i의 경계를 통과하는 에지 값들의 합이다.[1)]

2. 벡터 분할과 그래프 분할 사이의 관계

스펙트럴 방법은 그래프 G에 대한 Laplacian 행렬 Q의 고유벡터와 고유치를 이용한다. $Q\vec{\mu}=\lambda\vec{\mu}$를 만족하는 n차원 벡터 $\vec{\mu}$를 Q의 고유벡터라 하고 λ를 그에 대응하는 고유치라 한다. 여기서 Q의 고유벡터 집합을 $U=\{\vec{\mu_1}, \vec{\mu_2}, ..., \vec{\mu_n}\}$라 하고 그에 대응하는 고유치를 $\lambda_1, \lambda_2, ..., \lambda_n$라 한다. 이때 고유치 사이에 $\lambda_1 \le \lambda_2 \le ... \le \lambda_n$관계가 성립한다고 가정한다. 임의의 상수 $H \ge \lambda_d$에 대해 $n \times d$ 스케일드 고유벡터 행렬 V_d를 다음과 같이 정의할 수 있다.

$$V_d = \{\vec{\mu_1}\sqrt{H-\lambda_1}, \vec{\mu_2}\sqrt{H-\lambda_2}, ..., \vec{\mu_d}\sqrt{H-\lambda_d}\} \qquad (4)$$

여기서 $d \le n$이다. 이를 이용해 그래프 $G(V,E)$에 있는 각 노드 v_i를 d 차원 공간의 벡터 $\vec{y_i^d}$로 사상시킬 수 있다. 여기서 $\vec{y_i^d}$는 V_d의 i번째 행에 해당한다.[5] 가장 작은 고유치 $\lambda_1(=0)$에 대응하는 고유벡터가 $\vec{\mu_1}=[\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, ..., \frac{1}{\sqrt{n}}]^T$이고, 이것은 그래프 분할에 영향을 미치지 못하므로, (4)식에서 첫 번째 열을 제거할 수 있다.[5]

벡터 분할은 d차원에 사상된 벡터집합 $Y=\{\vec{y_1^d}, \vec{y_2^d}, ..., \vec{y_n^d}\}$를 k개의 중첩되지 않는 벡터 클러스터($S_1, S_2, ..., S_k$)로 분할하는 것을 말한다. 주어진 벡터분할에 대해 다음과 같은 비용함수를 정의할 수 있다.

$$F_{ratio}(P_v^k) = \sum_{i=1}^{k} \frac{\| \vec{Y_i^d} \|^2}{|S_i|} \qquad (5)$$

여기서 $\vec{Y_i^d} = \sum_{\vec{y_i^d} \in S_i} \vec{y_i^d}$ 이고, $|S_i|$는 벡터클러스터 S_i의 크기, 즉 S_i에 있는 벡터 개수를 의미한다. 만일 모든 $\vec{y_i^d} \in S_i$에 대하여 $v_i \in C_i$이면, 그래프분할 P_g^k는

1) 그림 1에서 클리크를 이용해 각 네트를 변환할 때 변환된 클리크의 에지에 uniform edge weight w = 1/(p-1)를 할당하면, e6_1, e6_2 그리고 e6_3의 에지 값은 1/2이 된다. 여기서 p는 어떤 네트에 연결된 핀(pin)의 갯수이다. 만일 노드 v_3가 클러스터 C_i인 경우, 결선도 (a)에서 e1, e6 그리고 e7이 잘리므로 E_i=3이 되고, 그래프 (b)에서는 Ei = weight(e1) + weight(e7) + weight(e6_1) + weight(e6_2) = 1 + 1 + 1/2 + 1/2 = 3이 된다.

벡터분할 P_v^k에 대응한다라고 말한다. 또한 그래프분할 P_g^k가 벡터분할 P_v^k에 대응하면, 모든 $\overrightarrow{y_i^d} \in S_i$에 대하여 $v_i \in C_i$인 관계가 성립하게 된다. 만일 $d = n$이고, P_v^k가 P_g^k에 대응하면 다음과 같은 등식이 성립한다.[5]

$$\sum_{i=1}^{k} \frac{\| \overrightarrow{Y_i^d} \|^2}{|S_i|} = kH - \sum_{i=1}^{k} \frac{E_i}{|C_i|} \quad (6)$$

따라서 식 (5)와 (6)로부터 $F_{ratio}(P_v^k)$가 최대가 되면, 정해진 k에 대해 그에 대응하는 그래프 분할의 스케일드 비용이 최소가 됨을 알 수 있다. 즉, 최소화 문제가 최대화 문제로 변환된 것이다. 스펙트럴 방법에 대해 보다 자세한 것은 참고분헌 [3] 에서 찾을 수 있다.

Ⅲ. 제안된 선형배열 알고리즘 (LIME : Linear Ordering with Incremental MErging)

1. 제안된 비용함수

기본적으로 제안된 알고리즘은 두 개의 클러스터를 병합하여 새로운 클러스터를 만드는 과정을 계속한다. 여기서 병합될 두 개의 벡터클러스터를 선택하는 비용함수를 제안한다. 먼저 주어진 q개의 벡터 클러스터 $\{S_1, S_2, ..., S_q\}$에서 S_l과 S_m이 병합되어 $(q-1)$개의 벡터 클러스터가 만들어 졌다고 하면 그에 대한 $F_{ratio}(P_v^{q-1})$은 다음과 같다.

$$\begin{aligned} F_{ratio}(P_v^{q-1}) &= \sum_{i=1, i\neq l, i\neq m} \frac{\| \overrightarrow{Y_i^d} \|^2}{|S_i|} + \frac{\| \overrightarrow{Y_l^d} + \overrightarrow{Y_m^d} \|^2}{|S_l \cup S_m|} \\ &= F_{ratio}(P_v^q) + \frac{\| \overrightarrow{Y_l^d} + \overrightarrow{Y_m^d} \|^2}{|S_l \cup S_m|} - \frac{\| \overrightarrow{Y_l^d} \|^2}{\| S_l \|} - \frac{\| \overrightarrow{Y_m^d} \|^2}{\| S_m \|} \end{aligned} \quad (7)$$

$F_{ratio}(P_v^q)$는 이미 알고 있으므로, $F_{ratio}(P_v^{q-1})$을 최대화하기 위해서는 다음 식이 최대가 되도록 두 개의 벡터 클러스터를 선택해야 한다.

$$F_{select}(S_l, S_m) = \frac{\| \overrightarrow{Y_l^d} + \overrightarrow{Y_m^d} \|^2}{|S_l \cup S_m|} - \frac{\| \overrightarrow{Y_l^d} \|^2}{\| S_l \|} - \frac{\| \overrightarrow{Y_m^d} \|^2}{\| S_m \|} \quad (8)$$

이미 주어진 선형배열에 대해 스케일드 비용함수 관점에서 최적해를 구하는 알고리즘이 있으므로 위 식을 이용해 주어진 그래프를 k개로 직접 분할하지 않고 선형 배열을 만들어 회로 결선도를 분할한다.

2. 제안된 선형 배열 알고리즘의 기본 아이디어

제안된 선형 배열 알고리즘을 설명하기 전에 선형 배열과 세그먼트에 대해 먼저 정의한다.

정의: *노드들의 선형 배열*

주어진 노드 집합 $V = \{v_1, v_2, ..., v_n\}$에 대해 노드들의 선형 배열 $[v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}]$은 전단사 함수 $\pi[1...n] \rightarrow [1...n]$에 의해 정의된다. 만일 $\pi(j) = i$이면, 노드 v_i는 선형배열에서 j 번째 노드가 된다.

정의: *세그먼트*

주어진 선형 배열 $[v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}]$에 대해 세그먼트는 선형배열에서 임의의 연속적인 노드들로 정의된다.

즉, 선형배열에서 i번째와 j번째 노드사이의 노드들 $[v_{\pi(i)}, \ldots, v_{\pi(j)}]$가 세그먼트가 된다. 본 논문에서는 δ와 Δ를 각각 세그먼트와 세그먼트 집합으로 정의한다.

제안된 알고리즘 LIME은 초기에 n개의 세그먼트로부터 시작한다. 초기 세그먼트 집합은 $\Delta^n = \{\delta_1, \delta_2, \ldots, \delta_n\}$이며, 각 세그먼트는 정확하게 하나의 노드로 구성되어있다. 제안된 알고리즘은 세그먼트 집합에서 두 개의 세그먼트를 선택하여 하나의 세그먼트로 병합한다. 여기서 두 개의 세그먼트에 대응하는 벡터 클러스터들은 수식 (8)을 최대화시키는 것들이다. 이와 같은 과정을 계속하면 결과적으로 하나의 세그먼트만 남게 되고, 이것이 구하고자하는 노드들의 선형배열이 된다.

LIME의 기본 알고리즘

1 $\Delta^n = \{\delta_1, \delta_2, \ldots, \delta_n\}$ 즉, 각 세그먼트 δ_i는 정확하게 서로 다른 하나의 노드를 포함하고 있다.
2 Set $m = n$
3 두개의 서로 다른 세그먼트 δ_a 와 δ_b를 Δ^m 에서 찾는다. 이때 이들 두 세그먼트에 해당하는 벡터클러스터는 수식(7) 즉, F_{select}를 최대화시키는 것들이다.
4 $\delta_{new} = merge(\delta_a, \delta_b)$. 즉, 두 세그먼트를 병합한다.
5 $\Delta^{m-1} = (\Delta^m \cup \{\delta_{new}\}) - \{\delta_a\} - \{\delta_b\}$
6 m = m - 1; if m > 1, goto step 3

그림 2. LIME에 대한 기본 알고리즘
Fig. 2. Basic idea of LIME.

그림 2는 제안된 LIME의 기본 알고리즘을 나타낸

것이다. 여기서 Δ^m은 m개의 세그먼트로 이루어진 세그먼트 집합을 나타낸 것이다. 그리고 δ_a와 δ_b는 Δ^m에 속해있는 세그먼트들이며 δ_{new}는 두 개의 세그먼트 δ_a와 δ_b를 병합하여 만든 세그먼트이다. 두 개의 세그먼트를 병합하는 방법은 그림 3에서와 같이 4가지로 구분할 수 있다. 세그먼트 δ_b의 모양을 결정하는 방법에 대해서 먼저 설명한다. 그림 3 (a)에 있는 δ_b를 A형(As is form)이라 하고, (b)에 있는 δ_b를 R형(Reflected form)이라 하자. 그리고 세그먼트 δ_b는 전 단계에서 δ_{b1}과 δ_{b2}를 병합하여 만들어 졌다고 가정한다. 먼저 δ_a가 δ_{b1}와 가까이 있는 것이 좋은지 혹은 δ_{b2}와 가까이 있는 것이 좋은지 알아본다. 만일 δ_a가 δ_{b1}와 가까이 있는 것이 좋으면 (a)와 같이 A형이 선택되고, 그렇지 않으면 (b)와 같이 R형이 선택된다. 이를 위해 두 개의 클러스터 $C_1 = (\{\delta_a\} \cup \{\delta_{b1}\})$ 과 $C_2 = (\{\delta_a\} \cup \{\delta_{b2}\})$를 정의하고 이에 대응하는 벡터 클러스터를 각각 S_1과 S_2라고 하자. 그리고 수식(5)를 최대화하도록 세그먼트 δ_b의 모양을 선택한다. 즉, $F_{ratio}(S_1) > F_{ratio}(S_2)$이면, A형이 선택되고 그렇지 않으면 R형이 선택된다. 세그먼트 δ_a모양도 이와 유사한 방법으로 결정할 수 있다.

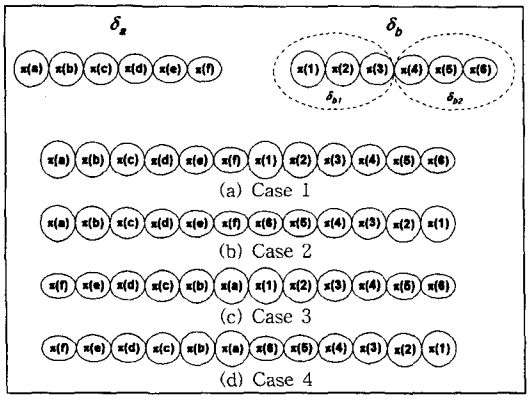


그림 3. 두 개의 세그먼트 δ_a와 δ_b를 병합하는 4가지 방법

Fig. 3. Four possible case to merge segment δ_a and δ_b.

그림 2에서 d개의 고유벡터를 이용할 경우, 제안된 알고리즘의 타임 컴플렉시티가 $O(dn^3)$이 되어 실제 회로결선도 분할에 응용하기가 어렵다. 다음절에서는 회로결선도의 성긴 특징을 이용하여 $O(dn^3)$보다 낮은 타임 컴플렉시티를 갖는 알고리즘을 제안한다.

3. LIME의 효과적인 구현

제안된 알고리즘을 효과적으로 구현하기 위해 본 논문에서는 회로결선도의 성긴 특성을 이용한다. 두 개의 세그먼트가 회로 결선도에서 직접 연결되어 있으면 그렇지 않은 경우보다 병합될 기회가 더 많다. 따라서 회로 결선도에서 직접 연결된 경우만 고려한다.

여기서 D^i를 회로 결선도에서 δ_i에 직접 연결된 세그먼트의 집합으로 정의한다. 이때 δ_i는 집합 D^i에 포함되지 않는다. 그러면 D^i에서 δ_i의 BMC(best merging candidate)를 정의할 수 있다. D^i에 있는 세그먼트 중 δ_i와 병합되었을 때, 제안된 비용함수 수식(8)을 최대로 하는 세그먼트를 BMC라 정의한다.

초기에 각 세그먼트 δ_i는 정확하게 하나의 노드만 가지고 있으며 각 노드에 대한 BMC를 구할 수 있다. 그러면 다음과 같이 집합 T를 정의할 수 있다.

$$T = \{(\delta_i, \delta_j) | \delta_i \in \Delta^n, \delta_j \in D^i\text{는 } \delta_i\text{의 BMC이다}\}$$

집합 T에서 제안된 비용함수 F_{select}를 최대로 하는 두 개의 세그먼트를 선택한다. 그리고 그들을 새로운 세그먼트로 병합한 후 집합 T를 갱신한다. 이와 같은 일을 반복적으로 수행하여 최종적으로 남은 하나의 세그먼트가 구하고자하는 선형 배열이 된다.[2)]

세그먼트 δ_c와 이에 직접 연결된 세그먼트로 이루어져 있는 세그먼트집합 D^C를 살펴보자. 만일 D^C가 변화되지 않았다면, δ_c의 새로운 BMC를 새롭게 고를 필요가 없다. 따라서 두 개의 세그먼트가 하나의 새로운 세그먼트로 병합된 후 병합된 세그먼트에 직접 연결되어 있는 세그먼트와 새로 만들어진 세그먼트에 한해 BMC를 새로 선택한다.

이 데이터를 잘 이용하기 위해 자료 구조는 수정된 heap을 이용하였다. Heap 자료 구조의 특징은 모든 노드의 key값이 그것의 어떤 자손의 key 값보다 크거나 같다는 것이다.

제안된 자료구조는 그림 4과 같다. 여기에는 2개의 double-linked list와 하나의 pointer가 있다. 두 개의 double-linked list 중 하나는 heap을 구현하는데 사용되었고, 다른 하나는 세그먼트와 그에 해당하는 노드를 연결하는 데 이용되었다. 그리고 pointer는 그 세그먼트의 BMC를 가리키는데 사용되었다. Heap 자

2) 이에 대한 time complexity는 $O(np)$가 된다. 여기서 p는 회로 결선도에서 전체 핀(pin)의 갯수와 같으며, $p = O(n)$이므로 $O(n^2)$알고리즘이 된다.

료구조에서 각 노드의 key 값은 그에 해당하는 세그먼트와 그 세그먼트의 BMC에 의해 결정된다. 그림 4에서 heap의 top 노드의 key값은 $F_{select}(S_a, S_b)$가 되며, S_a와 S_b는 δ_a와 δ_b에 해당하는 벡터 클러스터이다. 이 자료 구조에서 하나의 노드가 선택되면 그에 해당하는 세그먼트와 그에 BMC를 heap으로부터 O(log n) 만에 제거할 수 있다.

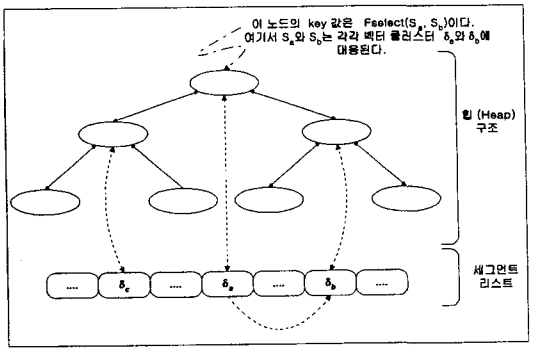


그림 4. 제안된 알고리즘에서 사용된 자료구조
Fig. 4. Data structure for the proposed algorithm LIME.

제안된 선형 배열 알고리즘

1 $\Delta^n = \{\delta_1, \delta_2, \ldots, \delta_n\}$ 즉, 각 세그먼트 δ_i는 정확하게 서로 다른 하나의 노드를 포함하고 있다.

2 For each $\delta_i \in \Delta^n$
δ_i의 BMC(Best Merging Candidate)를 찾아서 제안된 heap자료구조에 넣는다.

3 Set $m = n$

4 Heap의 top에 있는 세그먼트 δ_a와 그의 BMC δ_b를 heap에서 꺼낸다.

5 $\delta_{new} = merge(\delta_a, \delta_b)$

6 $\Delta^{m-1} = (\Delta^m \cup \{\delta_{new}\}) - \{\delta_a\} - \{\delta_b\}$

7 δ_{new} 의 BMC를 찾고 δ_{new}를 heap에 넣는다.

8 For each $\delta_k \in D^{new}$
δ_k를 heap에서 제거한다.
δ_k의 새로운 BMC를 찾는다.
δ_k를 heap에 넣는다.

9 $m = m - 1$; if $m > 1$, goto step 4

그림 5. 제안된 선형배열 알고리즘 LIME
Fig. 5. Proposed linear ordering algorithm LIME.

그림 5는 이를 바탕으로 만들어진 알고리즘이다. Step1과 step2는 초기화를 위한 부분이고 step 4, 5 그리고 6은 두 개의 세그먼트를 선택한 후 하나의 세그먼트로 병합하는 부분이다. 그리고 step 7과 8은 BMC를 갱신하는 부분이다. 제안된 알고리즘의 time complexity를 분석하기 위해 D^{new}와 D^k에 있는 세그먼트의 수를 각각 $n1$ 그리고 $n2$라 하면, step 8의 time complexity는 $O(n1(\log_2 n + dn^2))$가 된다. 여기서 worst case에 $n1 = O(n)$, $n2 = O(n)$ 그리고 $n1\ n2 = O(p)$이므로 제안된 알고리즘의 time complexity는 $O(n^2 \log_2 n + dp)$가 된다. 그러나 실제 $n1 << n$ 그리고 $n2 << n$ 임을 실험에서 알 수 있다.[3)]

Ⅳ. 실험

LIME이라고 불리는 제안된 선형배열 알고리즘은 C 언어로 구현되었으며 기존 선형 배열 알고리즘인 SFC [7], MELO [5] 와 비교되었다.

표 1. 10개의 고유벡터(d=11)를 이용해 선형배열을 만들 때 얻은 $n1$과 $n2$의 평균값 그리고 기존 선형배열 알고리즘 MELO와의 수행시간 비교
Table 1. Average value of $n1$ and $n2$ during product linear ordering with non-trivial ten eigenvectors and comparision of MELO and previous work in terms of run time.

Test Case	# modules	# nets	# pins	$n1$ 평균값	$n2$ 평균값	수행시간(단위: 초)	
						LIME	MELO
19ks	2844	3282	10547	44.2	66.8	3.5	32.5
prim1	833	902	2908	36.7	28.1	1.0	3.1
prim2	3014	3029	11219	60.4	48.9	4.8	36.4
test02	1663	1720	6134	50.7	58.6	2.9	11.4
test03	1607	1618	5807	46.5	47.0	2.2	10.6
test04	1515	1658	5975	47.2	43.6	2.2	9.5
test05	2595	2750	10076	58.1	62.6	6.1	27.2
test06	1752	1541	6638	48.5	88.4	1.8	12.7
balu	801	735	2697	32.5	35.7	0.9	2.8
struct	1952	1920	5471	23.5	29.1	2.9	15.4
biomed	6514	5742	21040	54.5	60.2	12.2	175.1
SUM	25090	24897	88512	503.1	569.0	40.4	336.8

3) 앞에서 heap 구조를 사용하지 않고 알고리즘을 구현하면 time complexity를 $O(np) = O(n^2)$로 줄일 수 있다. 그러나 실험에서 보인 것과 같이 $n1$ 과 $n2$가 거의 상수에 가까우므로 이 알고리즘은 $O(n \log n)$ behavior를 보이고 있다. 따라서 이 알고리즘이 기존 $O(n^2)$ 알고리즘보다 더 빠름을 알 수 있다.

각 회로 결선도는 클리크 모델을 이용하여 그래프로 변환되었다. 즉, *p-pin*으로 이루어진 네트은 클리크 모델로 변환되었으며 클리크의 각 에지는 $w=\frac{4}{p(p-1)}\frac{(2^p-2)}{2^p}$ 의 가중치 할당되었다. 이렇게 그래프로 변환된 다음 고유값과 고유치를 구하였다.

표 2. 제안된 알고리즘 LIME과 기존 SFC, MELO와의 스케일드 비용함수 면에서 성능비교 (스케일: 10^{-5})

Table 2. Comparison of LIME with SFC and MELO in terms of scaled cost function with the scaling factor of 10^{-5}.

Test case		Number of Clusters (k)									SUM
		10	9	8	7	6	5	4	3	2	
19ks	SFC	15.1	14.3	13.8	13.2	12.2	11.1	8.37	7.48	5.44	100.99
	MELO	9.85	9.10	8.31	7.87	6.84	6.14	5.32	4.99	4.79	63.21
	LIME	7.07	6.70	6.27	5.96	5.67	5.37	5.07	4.71	4.68	51.50
prim1	SFC	38.9	36.7	35.2	31.7	28.8	26.0	21.8	14.6	13.4	247.10
	MELO	44.6	41.9	39.7	37.0	34.0	29.4	22.5	17.1	13.4	279.60
	LIME	32.7	30.0	28.6	26.3	24.1	20.9	16.4	14.0	13.4	206.40
prim2	SFC	13.7	13.3	12.8	12.1	11.0	9.43	7.95	6.86	5.05	91.19
	MELO	13.7	12.7	12.0	11.2	10.1	9.18	7.95	6.76	4.71	88.30
	LIME	9.67	9.40	9.12	8.77	8.37	7.83	7.33	6.76	5.06	72.31
test02	SFC	25.5	24.1	22.8	20.9	18.5	16.1	13.4	10.9	8.07	160.27
	MELO	21.1	19.9	18.5	17.0	15.4	13.9	12.4	10.7	8.07	136.97
	LIME	17.9	16.4	15.7	15.1	14.2	13.1	12.6	11.7	8.42	125.12
test03	SFC	22.6	21.1	19.2	17.1	16.2	15.2	14.3	13.0	10.2	148.90
	MELO	19.0	17.6	16.7	15.3	14.6	13.7	12.5	11.6	9.25	130.25
	LIME	15.8	15.3	14.7	13.9	12.8	12.5	11.2	10.2	8.81	115.21
test04	SFC	22.2	19.9	17.8	17.6	16.5	15.1	11.6	8.17	5.78	134.65
	MELO	13.2	12.3	11.5	10.8	9.97	9.32	8.21	6.83	5.78	87.91
	LIME	11.7	11.0	10.1	9.62	9.17	8.54	7.93	6.52	5.78	80.36
test05	SFC	9.88	8.66	8.06	7.84	7.32	6.56	5.49	4.90	3.09	61.80
	MELO	7.42	7.03	6.53	6.11	5.79	5.50	4.85	4.35	3.09	50.67
	LIME	6.17	5.98	5.63	5.42	5.24	4.74	4.33	3.98	3.09	44.58
test06	SFC	27.1	25.1	23.7	20.2	18.4	16.5	13.7	11.3	9.21	165.21
	MELO	21.3	20.2	18.5	16.7	14.7	13.5	11.3	9.54	8.80	134.54
	LIME	16.3	15.2	14.2	13.1	12.2	11.3	10.7	9.41	7.68	110.09
balu	SFC	82.0	79.1	74.1	70.3	64.9	62.2	49.4	47.3	17.6	546.90
	MELO	54.0	50.1	46.5	43.2	40.0	36.7	32.3	24.4	17.6	344.80
	LIME	43.7	41.1	39.7	37.3	35.0	30.8	24.9	22.6	17.0	292.10
struct	SFC	12.1	11.2	10.5	9.41	8.65	7.93	7.05	6.42	4.85	78.11
	MELO	12.9	12.0	10.9	9.82	8.46	7.56	6.53	5.54	4.25	77.96
	LIME	8.73	8.28	7.94	7.57	7.18	6.26	5.67	4.83	4.10	60.56
bio	SFC	1.84	1.69	1.59	1.47	1.51	1.48	1.25	1.15	0.85	12.83
	MELO	1.87	1.73	1.62	1.49	1.34	1.23	1.11	0.89	0.61	11.89
	LIME	1.52	1.42	1.34	1.27	1.20	1.08	0.91	0.87	0.61	10.22
SUM	SFC	270.92	255.15	239.55	221.82	187.60	203.98	154.31	132.08	83.54	1749.0
	MELO	218.94	204.56	190.76	176.49	161.20	146.13	124.97	102.70	80.351	1406.1
	LIME	171.26	160.78	153.30	143.31	135.13	122.42	107.04	95.58	78.63	1167.5

실험을 위해 ACM/SIGDA benchmark을 이용하였다. 표 1에서 보는바와 같이, *n1*과 *n2*가 노드 수 *n*에 비해 상당히 작다는 것을 알 수 있다. 즉 평균적으로 $\frac{n1}{n}$ 그리고 $\frac{n2}{n}$이 약 2% 정도밖에 되지 않음을 알 수 있다. 일곱 번째와 마지막 열은 제안된 LIME과 기존 MELO의 Sparc 20에서 CPU time이다. 이 시간은 주어진 10 개의 고유벡터를 이용하여 선형 배열을 구하는데 걸린 시간을 나타낸 것이다. 실제 상황에서 *n1*과 *n2*가 거의 상수와 같으므로 LIME이 time complexity $O(n^2)$인 MELO에 비해 약 8배 정도 빠름을 알 수 있다.

다음 실험은 d ($2\le d\le 11$) 개의 고유벡터를 이용하여 선형 배열을 구하였다. 즉 d = 2 이면 두 번째 고유벡터만을 이용하였고, d = 10이면 두 번째부터 열한번 째까지의 고유벡터를 이용하여 선형 배열을 구하였다. 스펙트럴 방법에서 모든 고유벡터가 사용되지 않은 경우, 상수 H의 선택이 선형 배열에 영향을 미친다.[5] MELO에서는 H를 선택하는 몇 가지 방법을 보였으며, 그 중 $H=\lambda_d+\lambda_2$ 가 좀 낳은 결과를 낳았다. 여기서도 같은 상수를 이용하여 선형 배열을 구하였다.

이렇게 구한 선형 배열은 DP-RP[7]라는 프로그램을 통하여 k개의 분할을 얻었다. DP-RP는 주어진 선형 배열에서 스케일드 비용함수 면에서 최적의 k개 회로 분할을 할 수 있는 프로그램이다. 표 2에서는 10개의 선형배열과 DP-RP를 이용해 얻은 k개의 회로 분할 중 가장 좋은 결과를 실었다. 제안된 알고리즘은 기존 MELO나 SFC에 비해 스케일드 비용함수 면에서 각각 약 17% 그리고 33% 개선되었음을 알 수 있다.

V. 결 론

본 논문에서 회로분할을 위한 새로운 선형 배열 알고리즘을 제안하였다. 먼저 선형배열의 임의의 연속적인 구간을 세그먼트라 정의하였다. 따라서 각각의 노드들 역시 하나의 세그먼트가 된다. LIME은 초기에 *n*개의 세그먼트로부터 시작한다. 이때 각각의 세그먼트는 하나의 노드로 구성되어 있다. LIME은 제안된 비용함수를 이용해 두 개의 세그먼트를 점진적으로 병합하여 최종적으로 하나의 세그먼트를 구하였으며, 최종적으로 남은 하나의 세그먼트가 구하고자하는 선형 배열이 된다. LIME을 효과적으로 구현하기 위해 회

MELO에 비해 8배정도 속도 향상을 얻을 수 있었다. 뿐만 아니라 제안된 알고리즘은 스케일드 비용함수 면에서 약 17% 정도 성능이 향상되었다.

참 고 문 헌

[1] P.K.Chan, M.D.F.Schlag and J.Zien, "Spectral k-way ratio-cut partitioning and clustering", *IEEE Trans. on CAD* 13(9), pp. 1088-1096, Sept. 1994.

[2] C.J.Alpert and A.B.Kahng, "Recent directions in netlist partitioning: a survey", *INTEGRATION, the VLSI journal*, vol 19, pp. 1-81. 1995.

[3] C.M.Fiduccia and R.M.Mattheyses, "A linear time heuristic for improving network partitions", *Proc. ACM/IEEE Design Automation Conf.*, pp. 175-181, 1982.

[4] B.W.Kernighan and S.Lin, "An efficient heuristic procedure for partitioning graphs", *Bell Syst. Tech.J,* vol. 47, pp. 291-307, Feb. 1970.

[5] C.J.Alpert and S.-Z.Yao, "Spectral partitioning: The More Eigenvectors, the Better", *Proc. ACM/IEEE Design Automation Conf.*, pp. 195-200, 1995.

[6] L.Hagen and A.B.Kahng, "New spectral methods for ratio-cut partitioning and clustering", *IEEE Trans. on CAD* 11(9), pp. 1074-1085, Sept. 1992.

[7] C.J.Alpert and A.B.Kahng, "Multi-way partitioning via spacefilling curves and dynamic programming", *Proc. ACM/IEEE Design Automation Conf.*, pp. 652-657, 1994.

[8] B.M.Riess, K.Doll and F.M.Johannes, "Partitioning very large circuits using analytical placement techniques", *Proc. ACM/IEEE Design Automation Conf.*, pp. 646-651, 1994.

[9] C.J.Alpert and A.B.Kahng, "A general framework for vertex ordering with application to netlist clustering", *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 63-67, 1994.

[10] L.T.Liu, M.T.Kuo, S.C.Huang, and C.K. Cheng, "A gradient method on the initial partition of Fiduccia-Mattheyses algorithm", *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 229-234, 1995.

[11] C.J.Alpert and A.B.Kahng, "Geometric embedding for faster and better multi-way netlist partitioning", *Proc. ACM/IEEE Design Automation Conf.*, pp. 743-748, 1993.

저 자 소 개

成 桃 洙(正會員)

1990년 2월 한양대학교 전자공학과 졸업(B.S). 1992년 2월 한국과학기술원 전기및 전자공학과 졸업(M.S). 1997년 2월 한국과학기술원 졸업(Ph.D). 1997년 3월부터 1998년 2월까지 미국 SandCraft 연구원(MIPS R5400 마이크로프로세서 개발). 1998년 3월부터 현재까지 영남대학교 전기 전자 공학부 객원교수. 주관심 분야는 마이크로프로설계, DSP 칩 설계 그리고 설계 자동화