論文2000-37SD-2-8

스팩트럴 방법을 이용해 트랙 밀도를 최소화 할 수 있는 효과적인 데이터패스 배치 알고리즘

(An Efficient Datapath Placement Algorithm to Minimize Track Density Using Spectral Method)

成 桄 洙 *

(Kwang-Su Seong)

요 약

본 논문에서는 트랙 밀도를 최소화할 수 있는 효과적인 데이터패스 배치 알고리즘을 제안한다. 주어진 n개의 데이터패스 element 각각을 한 개의 클러스터라 놓고 이들 클러스터 중 가장 강하게 연결된 두 개를 선택하고 병합하는 과정을 한 개의 클러스터만 남을 때까지 반복한다. 병합될 두 클러스터내의 element들은 이미 각각 선형배열되어 있으므로 병합 시 이 두 선형배열을 연결하면 되며, 최종적으로 남은 클러스터의 선형배열의 처음과 끝을 연결하면 회전선형배열을 만들 수 있다. 이 회전선형배열에서 인접한 두 element 사이를 절단하면 서로 다른 n개의 선형배열을 만들 수 있으며 제안된 알고리즘에서는 이들 중 트랙밀도가 가장 낮은 선형배열을 선택한다. 본 논문에서는 스펙트럴방법을 이용해 d차원에 사상시킨 벡터의 내적이 최대가 되면 대응되는 두 클러스터가 강하게 연결되었음을 보였으며, 이를 이용해 병합될 두 클러스터를 찾는다. 기존 GA/SA[2] 방법과 비교하여 제안된 방법은 트랙밀도 면에서 유사한 성능을 내지만 수행시간 면에서 상당히 향상되었다.

Abstract

In this paper, we propose an efficient datapath placement algorithm to minimize track density. Here, we consider each datapath element as a cluster, and merge the most strongly connected two clusters to a new cluster until only one cluster remains. As nodes in the two clusters to be merged are already linearly ordered respectively, we can merge two clusters with connecting them. The proposed algorithm produces circular linear ordering by connecting starting point and end point of the final cluster, and n different linear ordering by cutting between two contiguous elements of the circular linear ordering. Among the n different linear ordering, the linear ordering to minimize track density is final solution. In this paper, we show and utilize that if two clusters are strongly connected in a graph, the inner product of the corresponding vectors mapped in d-dimensional space using spectral method is maximum. Compared with previous datapath placement algorithm GA/SA[2], the proposed algorithm gives similar results with much less computation time.

* 正會員, 嶺南大學校 電氣電子工學部

(Yeungnam university School of Electrical Electronic Engineering)

接受日字 : 1999年10月26日, 수정완료일 : 2000年1月19日

Ⅰ. 서 론

데이터패스는 체계적(hierarchy), 규칙적(regularity), 모듈방식(modularity)과 같은 성질을 가지며 구조화(structured)되어 있다.[1,2] 데이터패스에서는 m비트 데

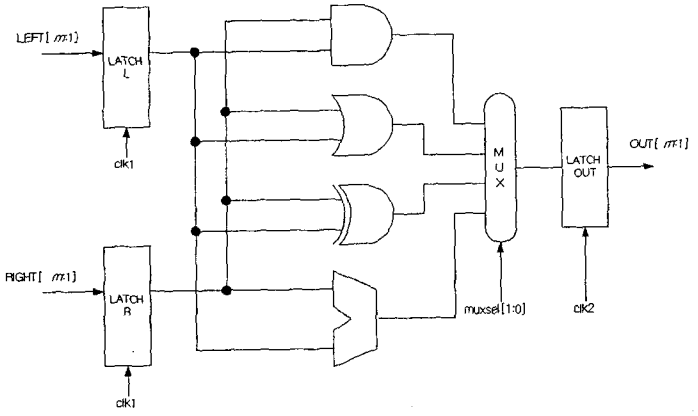


그림 1. 8개의 데이터패스 element로 구성된 간단한 mqlxm ALU회로도
Fig. 1. A simple m-bit ALU with 8 datapath elements.

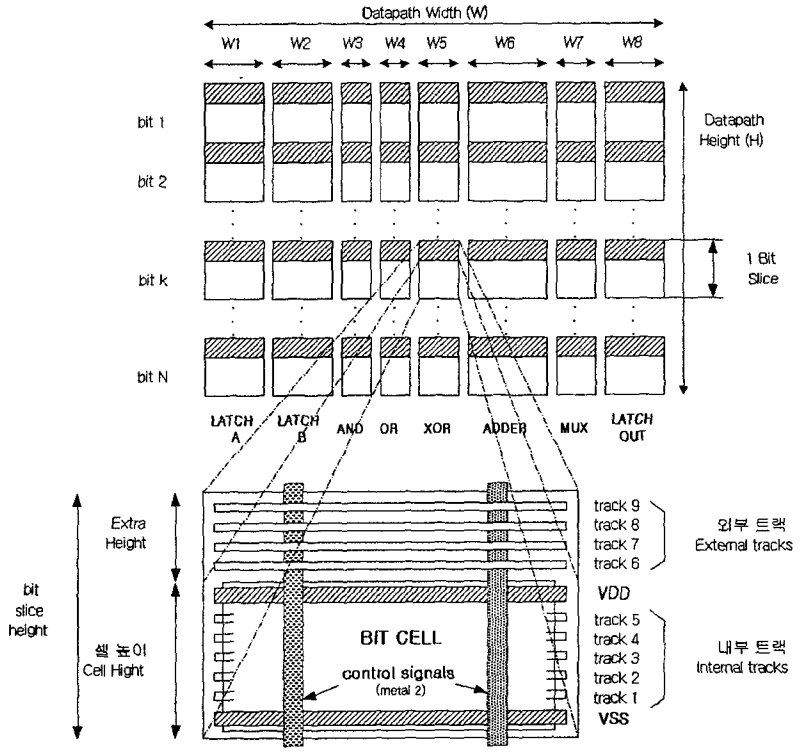


그림 2. 8개의 데이터패스 element로 배치한 레이아웃과 비트셀의 내부 구조
Fig. 2. Layout of datapath with 8 elements and detailed view of bit cell.

이터가 처리되기 때문에 m개의 동일한 회로를 이용하여 구현할 수 있다. 이와 같은 비트 슬라이스 방식의 데이터 패스의 장점을 이용하면 고속, 저전력 그리고 콤팩트한 레이아웃을 할 수 있으므로 마이크로프로세서와 디지털 시그널 프로세서와 같은 곳에서 널리 이용되고 있다[3,4]. 또한 이들 칩에서 데이터패스 면적이 전체 칩 면적의 30%에서 60%정도 되므로 데이터패스를 최적화 하는 것이 매우 중요하다.[2,5,6]

그림 1은 8개의 데이터패스 element로 구성되어 있으며 m비트 데이터를 처리할 수 있는 m비트 ALU회로도이다. 이 데이터패스는 비트 슬라이스 방식으로 그림 2과 같이 구현될 수 있다. 그림 2에서와 같이 데이

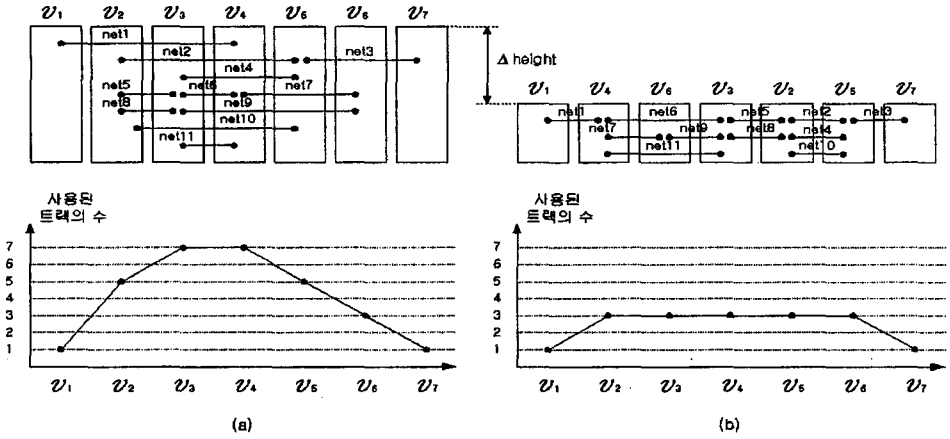


그림 3. 7개의 데이터패스 element와 11개의 net을 가진 회로를 일차원에 배치하는 순서에 따라 사용되는 track의 개수에 큰 차이가 있음을 나타내고 있다. (a)의 경우 7개의 track이 사용되지만 (b)의 경우 3개의 track만으로 구현할 수 있다

Fig. 3. Example having seven elements with 11 net shows the effect of element ordering on track density. Total number of tracks in (a) is 7, while that in (b) is 3.

터패스내의 그림 7 모든 비트 cell의 cell 높이는 일정하다. 또한 각각의 데이터패스 element에 대해 폭이 다르지만 각 element내에 있는 cell들의 폭은 일정하다. Element와 element를 연결하는 net은 내부 트랙을 이용하며 구현되며, 내부 트랙만으로 연결을 할 수 없으면 외부 트랙을 사용하게 된다. 그림 2에서 데이터패스의 면적 $S = W \times H$가 됨을 알 수 있다. 여기서 W는 모든 element들의 폭의 합으로 element를 어떻게 배치하더라도 상수가 된다. H는 8개의 element 중 bit slice 높이가 가장 높은 것에 의해 결정된다. N개의 데이터패스 element로 구성된 데이터패스에서 H는 수식 (1)과 같이 표현된다.

$$H = m \times \max_{i=1}^{n}(bit\_slice\_height[i]) \tag{1}$$

그림 2에서 bit slice 높이는 cell의 높이 더하기 extra track 때문에 발생하는 extra 높이를 더한 것이다. 만일 1개의 extra track을 사용하는데 K만큼의 높이가 증가한다고 하면 i번째 bit slice 높이는 수식 (2)와 같이 표현된다.

$$\begin{aligned} &bit\_slice\_height[i] \\ &\quad = CellHeight[i] + ExtraHeight[i] \\ &\quad = CellHeight[i] + K \\ &\qquad \times Number\ of\ Extra\ Tracks\ \text{in}\ element[i] \end{aligned} \tag{2}$$

여기서 모든 cell의 높이는 일정하므로 bit slice 높이를 줄이기 위해서는 extra track의 수를 최소화해야 함을 알 수 있다.

그림 3은 7개의 element $\{v_1, v_2, \ldots, v_7\}$과 11개의 net으로 이루어진 데이터패스에서 한 개의 bit slice에 대한 회로 결선도를 나타낸 것이다. 이들 데이터패스 element를 어떻게 일차원에 배치하느냐에 따라 사용되는 트랙의 수에 많은 차이가 있음을 알 수 있다. 그림 3 (a)와 같은 순서로 element를 배치하면 이를 구현하는데 7개의 track이 필요하며 그림 3 (b)와 같은 순서로 element를 배치하면 3개의 track으로 구현된다. 즉, 어떻게 데이터패스 element를 배치하느냐에 따라서 사용되는 track의 개수가 결정되고 이에 때라 데이터패스의 크기가 결정된다. 데이터패스 배치에 따른 면적의 변화를 설명하기 위해 참고문헌 [2]에서 그림 2와 3을 인용하였다.

이와 같이 n개의 element $\{v_1, v_2, \ldots, v_n\}$를 일차원에 배치하는 문제를 선형배열(linear ordering)[7,8]이라 한다. 주어진 n개의 element $\{v_1, v_2, \ldots, v_n\}$에 대한 선형배열 $[v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}]$은 전단사 함수 $\pi[1\ldots n] \rightarrow [1\ldots n]$로 정의된다. 만일 $\pi(j) = i$이면, v_i는 선형배열 j 번째에 해당된다. 이렇게 선형배열이 구해지면 $(n-1)$개의 점에서 선형배열을 2개로 분할 할 수 있으며 $v_{\pi(i)}$와 $v_{\pi(i+1)}$사이를 분할하였을 때 끊어지는 net

의 수를 Ei라 하면 그 선형배열 순서대로 데이터패스 element를 배치하는데 $\max_{i=1}^{n-1} E_i$ 만큼의 track이 필요하게 된다. 그림 3(a)에서는 $v_{\pi(3)}$와 $v_{\pi(4)}$에서 E_i의 값이 최대치인 7이 되므로 필요한 track 개수는 7개이며, (b)에서는 E_i의 최대 값이 3이므로 필요한 트랙의 수는 3개가 된다. 따라서 데이터패스 선형배열의 최적해는 $\min\{\max_{i=1}^{n-1} E_i\}$이 되도록 데이터패스 element를 배치하는 것이다. 앞으로 기존 선형배열 및 회로분할에서 사용되는 기호를 사용하기 위하여 element를 노드라 하기로 한다.

이와 같이 데이터패스의 트랙을 최소화하는 문제는 데이터패스 element를 일차원에 배치하는 선형배열 문제로 NP-hard 문제이다[9]. 선형배열 문제를 풀기 위해 다양한 휴리스틱(heuristic)이 제안되었으며 크게 회로분할(partitioning)을 이용한 방법, 스팩트럴 방법 그리고 stochastic을 이용한 simulated annealing과 genetic 방법이 있다. 회로분할을 이용한 방법은 빠른 시간에 해를 구할 수 있으나 국소 최적해에 빠지는 경우가 많은 단점을 지니고 있다[10]. 이를 극복하기 위해 클러스터링 기법이 많이 이용되기도 하였다.[11,12] 반면 simulated annealing[13] 방법은 hill climbing이 가능하므로 *global optimum*을 찾을 수 있으나 많은 시간이 소요된다는 단점을 가지고 있었다. 이를 극복하기 위해서 simulated annealing과 genetic 알고리즘을 동시에 이용하는 방법이 제안되어 simulated annealing과 유사한 결과를 내면서도 simulated annealing보다 빠른 방법이 제안되었으나 partitioning 기법이나 스펙트럴 방법에 비해 많은 시간을 요하고 있다[2]. 최근 그래프의 고유벡터와 고유치를 이용하는 스펙트럴 방법에서 많은 성과가 있었다. 이 방법은 그래프의 고유벡터와 고유치를 이용해 각 노드를 d차원 공간의 벡터로 사상한 후 이들 벡터를 분할하여 그에 대응하는 그래프를 분할하는 방법이다. 그래프 분할과 벡터 분할 사이에 정확한 관계가 있음을 보인 후 이 분야에서 괄목할 만한 성과가 있었다[7]. 그러나 스펙트럴 방법을 이용해 구한 선형배열의 비용함수(cost function)은 스케일드 비용(scaled cost)임으로 데이터패스 배열 비용함수와 다소 차이가 있다[7,8,14,15].

본 논문에서는 주어진 n개의 각각의 노드를 하나의 클러스터로 취급하여 n개의 클러스터로부터 시작한다. 그리고 이들 클러스터들에서 가장 강하게 연결된 두 개의 클러스터를 병합하여 하나의 클러스터로 만드는 과정을 한 개의 클러스터만 남을 때까지 반복한다. 병합될 클러스터 내부 노드들 간에 이미 선형배열이 되어 있으므로, 새로운 클러스터는 이들을 연결하여 만들게 된다. 최종적으로 남은 클러스터가 선형배열이 되며, 이 선형배열의 처음과 끝을 연결하여 회전선형배열(circular linear ordering)을 만들게 된다. 이렇게 만들어진 회전선형배열에서 인접한 노드와 노드사이를 절단하면 서로 다른 n개의 선형배열을 구할 수 있으며, 제안된 알고리즘에서는 이들 중 트랙 수를 최소화하는 선형배열을 해로 선택한다. 본 논문에서는 두 개의 클러스터를 선택할 때 스펙트럴 방법을 이용하였다. 클러스터 사이의 에지 값은 d-차원에 사상된 두 클러스터의 벡터의 내적(inner product)에 비례함을 보였으며, 벡터들 사이에 내적이 가장 큰 두 클러스터를 병합하였다. SA/GA에 비해 제안된 알고리즘은 사용된 트랙 수면에서 유사하지만 수행 시간 면에서는 상당히 개선되었다.

본 논문은 다음과 같이 구성되어 있다. 제 Ⅱ 장에서는 스펙트럴 방법에 관한 내용이 있으며, Ⅲ장에서는 이 알고리즘을 제안하게된 동기에 대해 설명하며, Ⅳ장에서는 제안된 알고리즘에 대해 설명한다. Ⅴ장에서는 실험 결과를 보이고 있다.

Ⅱ. Spectral Method

스펙트럴 방법을 이용해 데이터패스 element를 배치하기 위해서는 주어진 회로 결선도를 그래프로 변환해야 한다. 하이퍼그래프로 표현되는 회로 결선도를 정확하게 그래프로 변환하는 방법이 없으므로 클리크(clique) 모델을 이용해서 근사화된 그래프로 변환한다[7,8,16]. 변환된 그래프 $G(V, E)$는 노드 집합 $V = v_1, v_2, \cdots v_n$과 에지 집합 $E = e_1, e_2, \cdots e_l$로 구성되며, 이에 대해 다음과 같이 adjacency 행렬 A, degree 행렬 D 그리고 Laplacian 행렬 $Q(=D-A)$를 정의할 수 있다. 대칭인 adjacency 행렬 $Q(=D-A)$로 표현되며 여기서 a_{ij}는 두 노드 v_i와 v_j 사이의 에지 값이다. 그리고 diagonal 행렬인 degree 행렬은 $D=(d_{ii})$로 표현되며 d_{ii}는 그 노드의 degree가 된다. 즉 $d_{ii}=\sum_{j=1}^{n} a_{ij}$이다.

스팩트럴 방법은 그래프 G에 대한 Laplacian 행렬

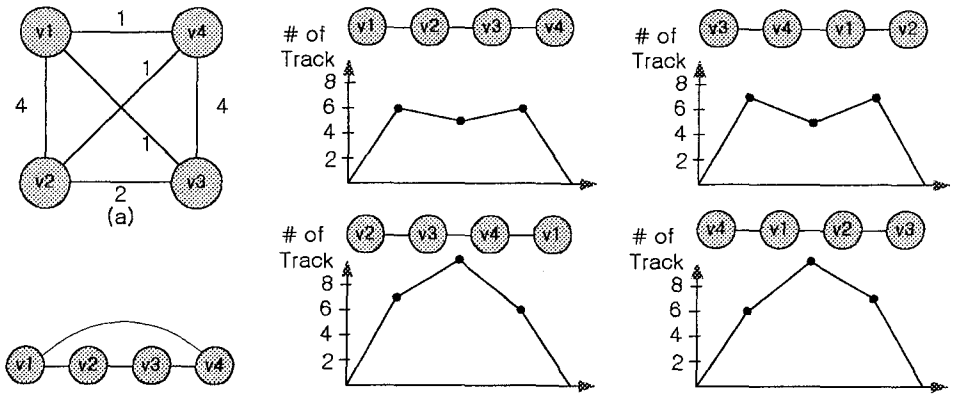


그림 4. 4개의 노드와 6개의 에지로 이루어진 회로도로부터 최적의 선형배열을 얻는 방법. (a)는 주어진 회로도를 표시한 것이고, (a)로부터 (b)와 같이 최적의 회전선형배열을 찾는다. 이렇게 찾은 선형배열에서 인접한 두 노드사이를 절단하여 (c)와 같이 4개의 선형배열을 얻으며 이 중 트랙 개수가 가장 적은 v_1, v_2, v_3, v_4의 선형배열이 최적의 선형배열이 된다

Fig. 4. A method to obtain optimal linear ordering from a circuit netlist with 4 nodes and 6 edges. (a) represents circuit netlist, and (b) shows optimal circular linear ordering of circuit (a), and (c) are linear orderings produced by cutting between contiguous two nodes of circular liner ordering. Among four linear orderings in (c) and v_1, v_2, v_3, and v_4 sequence is optimum linear ordering.

Q의 고유백터와 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 고유백터 집합는 $U=\{\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_n}\}$라하고 그에 대응하는 고유치를 $\lambda_1, \lambda_2, \ldots, \lambda_n$이라 한다. 이때 고유치 사이에 $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$관계가 성립한다고 가정한다. 임의의 상수 $H \ge \lambda_d$에 대해 $n \times d$ 스케일드 고유백터 행렬 V_d를 다음과 같이 정의할 수 있다.

$$V_d = \{\vec{\mu_1}\sqrt{H-\lambda_1}, \vec{\mu_2}\sqrt{H-\lambda_2}, \cdots, \vec{\mu_d}\sqrt{H-\lambda_d}\} \quad (3)$$

여기서 $d \le n$ 이다. 이를 이용해 그래프 $G(V, E)$에 있는 노드 v_i를 d차원 공간에 벡터 $\vec{y_i^d}$로 사상시킬 수 있다. 여기서 $\vec{y_i^d}$는 V_d의 i번째 행에 해당한다. 가장 작은 고유치 $\lambda_1(=0)$에 대응하는 고유벡터가 $\vec{\mu_1}=\left[\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}\right]^T$이고 이것은 그래프 분할에 영향을 미치지 못하므로 식 (4)에서 첫 번째 열을 제거할 수 있다[7]. 모든 고유벡터가 사용되었을 때, 그래프 클러스터 C_i와 그에 대응되는 벡터클러스터 S_i에 대해서 다음과 같은 관계가 성립된다[7].

$$\text{degree}(C_i) = H - \| \vec{Y_i} \|^2 \quad (4)$$

여기서 $\vec{Y_i}$는 C_i에 대응되는 벡터클러스터 S_i 내부에 있는 벡터들의 합으로 $\vec{Y_i} = \sum_{y_j \in S_i} \vec{y_j}$ 로 표현되며, degree(C_i)는 클러스터 C_i의 degree를 나타낸다. 보다 자세한 내용은 참고문헌 [7]를 참고하기 바란다.

Ⅲ. Motivation

본 논문에서는 주어진 n개 각각의 노드를 하나의 클러스터로 취급한다. 그리고 이들 클러스터들에서 가장 강하게 연결된 두 개의 클러스터를 병합하여 하나의 클러스터로 만드는 과정을 반복한다. 병합될 두 개의 클러스터에서는 이미 각 클러스터 내부에 있는 노드들 간에 선형배열이 되어 있으므로, 두 개의 클러스터를 병합한 클러스터내의 선형배열은 기존 두 개의 클러스터를 연결하여 얻을 수 있다. 이와 같이 두 개의 클러스터를 반복적으로 선택 병합하면 최종적으로 하나의 선형배열이 나오며, 이 선형배열의 처음과 끝을 연결하여 회전선형배열(circular linear ordering)을 만들게 된다. 이렇게 만들어진 회전선형배열에서 인접한 노

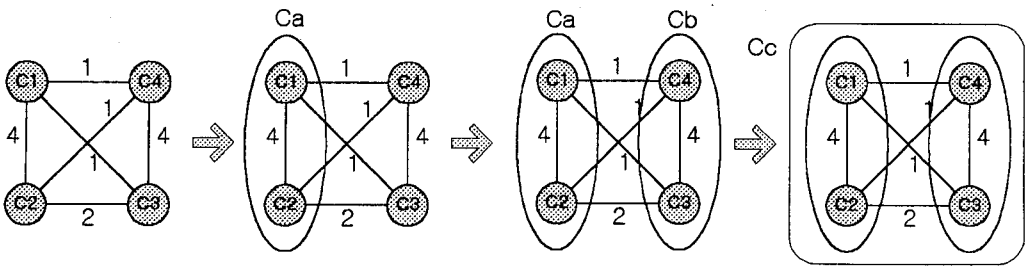


그림 5. 초기에 각 클러스터는 정확하게 한 개의 노드만 포함하고 있다. C_1, C_2, C_3 그리고 C_4는 각각 v_1, v_2, v_3 그리고 v_4를 포함하고 있다. 이들 중 가장 강하게 연결된 두 클러스터 C_1과 C_2를 병합하여 새로운 클러스터 C_a를 만들고, C_3와 C_4를 병합하여 C_b를 만든다. 그리고 최종적으로 C_a와 C_b를 병합하여 C_c를 만들며 이것이 회전선형배열이 된다

Fig. 5. For a given netlist, each cluster has only one disjoint nodes. Thus cluster C_1, C_2, C_3 and C_4 have v_1, v_2, v_3 and v_4, respectively in the initial stage. As C_1 and C_2 are the most strongly connected among the four cluster, those are merged into new cluster C_a And then C_3 and C_4 are merged into C_b among three clusters. Finally C_a and C_b are merged to C_c which corresponds to the final circular linear ordering.

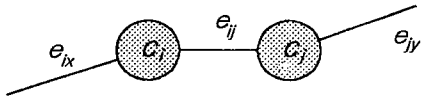
드와 노드사이를 절단하면 서로 다른 n개의 선형배열을 구할 수 있으며, 제안된 알고리즘에서는 이들 중 가장 좋은 선형배열을 해로 선택한다.

그림 4 (a)는 4개의 노드와 6개의 net이 연결된 회로도를 나타내고 있다. 이 회로도를 이용해 선형배열을 만들 경우 두 노드의 weight가 강한 노드 v_1과 v_2, v_3와 v_4그리고 v_2와 v_3를 가까이 배치하는 것이 최적의 방법임을 알 수 있다. 이와 같이 배치하면 (b)와 같이 회전선형배열을 얻을 수 있으며 선형배열에서 인접한 두 노드 사이를 끊으면 (c)와 같이 서로 다른 4개의 선형배열을 얻을 수 있다. 이 4개의 선형배열 중 track 수가 6개로 최소인 v_1, v_2, v_3, v_4 순서의 선형배열이 최적해임을 알 수 있다.

최적의 회전선형배열을 구하기 위해서 본 논문에서는 1개의 노드만으로 이루어진 n개의 클러스터 중 가장 강하게 연결된 두 개의 클러스터를 새로운 클러스터로 병합하여 $n-1$개의 클러스터로 만들며, 이런 과정을 연속적으로 수행하여 최종적으로 한 개의 클러스터가 될 때까지 계속 병합한다. 최종적으로 남은 하나의 클러스터가 선형배열이 되고 배열의 처음과 끝에 있는 두 개의 노드를 연결하여 회전선형배열을 구하게 된다. 이 방법을 이용해 그림 4 (a) 회로도의 회전선형배열을 얻는 과정은 그림 5와 같다.

그림 5 (a)에서 v_1, v_2, v_3 그리고 v_4를 각각 클러스터 C_1, C_2, C_3 그리고 C_4라 하자. 여기서 가장 강하게 연결된 것 중 하나인 C_1과 C_2를 병합하여 (b)와 같이 새로운 클러스터 C_a를 만든다. 그리고 C_a, C_3, C_4 3개의 클러스터 중 가장 강하게 연결된 C_3와 C4를 병합하여 새로운 클러스터 C_b를 만든다. 그 다음 나머지 두 클러스터 C_a와 C_b를 병합하여 최종적으로 C_c라는 클러스터를 만들게 된다. 클러스터 C_c는 C_a와 C_b로 구성되어 있으며 C_a와 C_b역시 다른 두 클러스터를 병합하여 만든 것이다. 이 경우 C_a내부에 있는 C_1, C_2와 C_b 내부에 있는 C_3, C_4사이의 배열은 그림 6과 같이 네 가지가 된다. 이 중 track 개수를 최소화하는 (a)가 선택되며 이 선형배열의 시작점과 끝점을 연결하여 그림 4 (b)의 회전선형배열을 구하게 된다.

두 클러스터 C_i와 C_j사이의 에지 비용은 다음과 같이 구할 수 있다. 아래 그림에서 두 클러스터 C_i와 C_j 사이의 edge cost는 e_{ij}이며 이것은 다음과 같다.



$$2e_{ij}=(e_{ix}+e_{ij})+(e_{iy}+e_{ij})-(e_{ix}+e_{iy}) \quad (5)$$

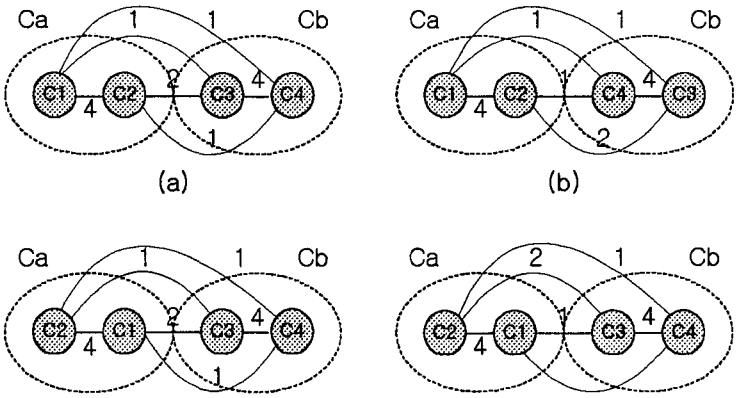


그림 6. 클러스터 C_a와 C_b는 각각 C_1, C_2 그리고 C_3, C_4로 이루어져 있다. 클러스터 C_a와 C_b를 병합하는 방법은 (a), (b), (c) 그리고 (d)와 같이 네 가지가 있으며, 트랙의 개수를 최소화하는 (a) 방법으로 병합하게 된다

Fig. 6. Cluster C_a and C_b consists of C_1, C_2 and C_3, C_4 respectively. There are four methods to merge C_a and C_b like (a), (b), (c) and (d). We choose method (a) to merge those two clusters for minimizing number of tracks.

여기서 e_{ix}는 클러스터 C_i에 연결된 edge cost 중 C_j와의 edge cost를 제외한 모든 edge cost이고, ejy는 클러스터 C_j에 연결된 edge cost 중 C_i와의 edge cost를 제외한 모든 edge cost이다. degree(C_i)는 클러스터 Ci에 연결된 모든 edge cost의 합으로 정의되므로 식 (5)는 다음과 같이 표현할 수 있다.

$$2e_{ij} = \text{degree}(C_i) + \text{degree}(C_j) - \text{degree}(C_i + C_j) \qquad (6)$$

또한 스팩트럴 방법에서 그래프 분할과 벡터분할 사이에 식 (4)와 같이 $\| \overrightarrow{Y_i} \|^2 = H - \text{degree}(C_i)$ 관계가 있으므로 (6) 식은 다음과 같이 표현된다. 여기서 $\overrightarrow{Y_i} = \sum_{y_j \in S_i} \overrightarrow{y_j}$이다.

$$2e_{ij} = (H - \| \overrightarrow{Y_i} \|^2) + (H - \| \overrightarrow{Y_j} \|^2) - (H - \| \overrightarrow{Y_i} + \overrightarrow{Y_j} \|^2) \qquad (7)$$

정리하면

$$2e_{ij} = H + \| \overrightarrow{Y_i} + \overrightarrow{Y_j} \|^2 - \| Y_i \|^2 - \| \overrightarrow{Y_j} \|^2 = H + 2\ \overrightarrow{Y_i} \cdot \overrightarrow{Y_j} \qquad (8)$$

여기서 H는 상수이므로 사상된 두 벡터 사이의 내적이 최대가 되는 두 클러스터를 병합하는 것이 결과적으로 강하게 연결된 두 클러스터를 병합하는 것이 됨을 알 수 있다. 따라서 본 논문에서는 병합될 두 클러스터를 찾을 때 두 클러스터에 해당하는 벡터들의 내적이 최대가 되는 클러스터를 선택하게 된다.

Ⅳ 제안된 알고리즘

제안된 알고리즘에서는 주어진 n개의 노드 각각을 하나의 클러스터로 만들고 이들을 순차적으로 병합하여 회전선형배열을 만들고, 그 회전선형배열에서 인접한 두 노드 사이를 절단하여 n개의 선형배열을 얻는다. 그리고 n개의 선형배열 중 사용된 트랙의 개수 즉, $\max_i^{n-1} E_i$ 를 최소화하는 선형배열을 선택하게 된다. 회전 선형배열을 구하는 방법은 다음과 같다. 한 개의 노드로 이루어진 n개의 클러스터에서 가장 강하게 연결된 두 클러스터를 식 (8)을 이용해 선택한다. 이렇게 선택된 두 클러스터는 새로운 클러스터로 병합되면 전체적으로 클러스터의 개수는 하나 줄게 된다. 이와 같은 일을 $n-1$번 반복하게 되면 결국 한 개의 클러스터만 남아 선형배열이 구해지며, 이 선형배열의 처음과 끝을 연결하여 회전선형배열을 얻게 된다. 이에 대한 회전선형알고리즘이 그림 7에 있다. 이렇게 회전선형배열이 만들어지면 회전선형배열에서 인접한 두 노드 사이를 절단하면 n개의 서로 다른 선형배열을 만들 수

있으며 이 중 track 수를 최소로 사용하는 선형배열을 선택하게 된다.

제안된 회전선형배열 알고리즘	
1	$P=C_1, C_2, \cdots, C_n$ 각 클러스터 C_i는 정확하게 서로 다는 하나의 노드만 포함하고 있다.
2	$m=n$
3	집합 P에서 클러스터에 대응되는 벡터의 내적이 최대가 되는 C_a와 C_b를 찾는다
4	C_{new}= merge (C_a , C_b)
5	$V=(V \cup \{C_{new}\}) - \{C_a\} - \{C_b\}$
6	$m=n-1$; if $m > 1$ goto step 3
7	선형배열의 처음과 끝을 연결하여 회전선형배열을 만든다.

그림 7. 제안된 회전선형배열 알고리즘
Fig. 7. Proposed circular linear ordering algorithm.

다음은 두 개의 클러스터 C_a와 C_b를 병합하는 방법에 대해 설명하고 있다. C_a는 C_{a1}과 C_{a1}를 병합하여 만들었고, C_b는 C_{b1}과 C_{b2}를 병합하여 만들었다고 하자. 이들 두 개의 클러스터를 병합하는데는 그림 8과 같이 4 가지 방법이 있다. 이 중 트랙 개수를 최소화하는 배열이 선택하게 된다.

D개의 고유벡터를 이용할 경우 제안된 알고리즘의 time complexity는 회전선형배열을 구하는 것에 의해 결정된다. 두 개의 클러스터를 선택하는데 $O(dn^2)$가 들므로 전체적으로 $O(dn^3)$이 된다. Element의 개수가 많아지면 큰 문제를 푸는데 상당한 시간이 소요되지만, 현재 사용되는 데이터패스 element의 개수가 약 수 백개 이내이므로 빠른 시간에 해를 구할 수 있다. Element의 수가 많아지면 [8]에서 사용한 자료구조(data structure)를 이용해 time complexity를 낮출 수 있다.

V. 실 험

제안된 알고리즘을 C언어로 구현하여 Ultra Sparc에서 실험하였으며 기존 선형배열 알고리즘인 GA-SA[2], MELO[7]와 비교하였다. 각 회로 결선도는 클리크 모델을 이용하여 그래프로 변환하였다. 즉, p-pin으로 이루어진 네트는 크리크 모델로 변환되었으며 클리크의 각 에지는 $w=\frac{4}{p(p-1)}$의 가중치가 할당되었다. 이렇게 그래프로 변환한 다음 고유벡터와 고유값을 구하였다. 실험을 위해 HK486[6] 마이크로프로세서 설계에서 이용

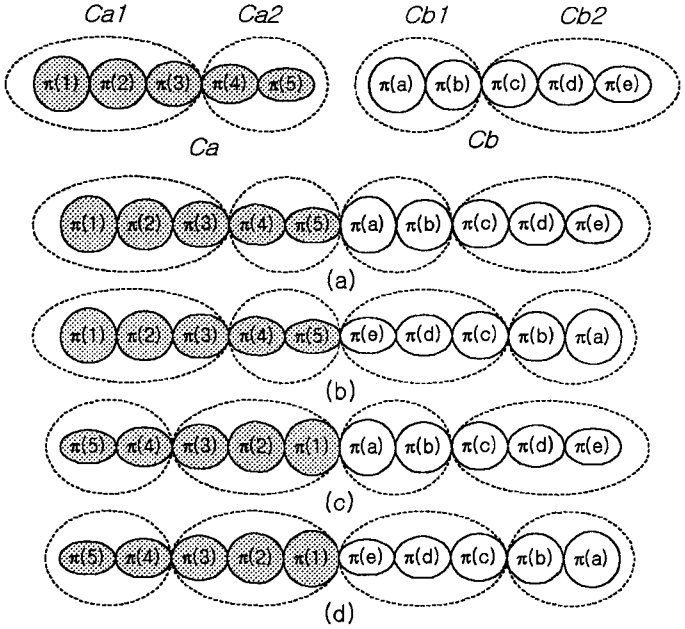


그림 8. 클러스터 C_a와 C_b를 병합하는 4가지 방법
Fig. 8. Four methods to merge cluster C_a and C_b.

표 1. B실험에 이용된 9개 데이터패스의 노드 수와 네트 수를 나타내고 있으며, 기존 선형배열 알고리즘인 GA/SA[2], MELO[7]와 제안된 알고리즘의 성능평가를 나타내고 있다

Table 1. Number of nodes and nets for nine datapath example, and comparison of proposed algorithm with GA/SA[2] and MELO[7].

데이터패스 이름	노드 수	네트 수	GA/SA		MELO		제안된 방법	
			트랙 수	수행시간	트랙 수	수행시간	트랙 수	수행시간
cache	25	28	5	133.8	8	< 0.1	6	0.1
dunit	28	33	5	274.4	7	< 0.1	5	< 0.1
funit	16	27	3	84.3	4	< 0.1	4	< 0.1
gunit	43	59	7	588.7	7	< 0.1	8	0.1
punit	46	57	8	606.0	6	< 0.1	6	0.1
sunit	64	105	9	1645.4	10	< 0.1	9	0.4
xcore	82	101	9	1250.1	10	0.1	8	0.3
xrfile	48	89	9	1358.1	13	0.1	10	0.6
xtlat	76	98	9	1834.3	7	0.1	7	0.7
sum	428	597	64	7,775.1	72	0.5	63	2.7

된 9개의 예제를 이용하였다. 실험에서는 첫 번째 고유치를 제외한 10개의 고유치를 이용하였으며 $H=\lambda_{11}+\lambda_2$ 를 이용하였다.

이 실험에서 보는바와 같이 수행시간 면에서 time complexity가 $O(n^2)$인 MELO가 가장 빠른 것을 볼 수 있다. 그러나 제안된 알고리즘 역시 모든 예제에 대해 1초 이내의 수행 시간을 보여 실제 응용면에서 큰 차이가 없음을 나타내고 있다. 앞으로 칩 크기가 커지며, 데이터패스 기능이 많아지게 되면 element의 수가 많아지리라 기대된다. 이 경우 참고문헌 [8]에서 사용한 데이터 구조를 이용해 time complexity를 줄일 수 있다. 그러나 GA/SA과 비교하면 수행 시간 면에서 많은 차이가 있음을 보이고 있다. GA/SA를 이용할 경우 전체 9개 예제를 선형배열하기 위해서는 약 2시간 정도가 소요됨을 알 수 있다. 제안된 방법은 사용된 트랙수 면에서 MELO보다 평균 12.5%정도 성능향상을 나타냈으며 GA/SA와 유사한 결과를 나타내고 있다.

VI. 결 론

본 논문에서는 데이터패스 면적을 최소화하기 위한 데이터패스 element 배치에 관한 효과적인 알고리즘을 제시하였다. 데이터패스 면적을 최소화하기 위해서는 데이터패스 element의 순서를 잘 배열하여, 사용되는 extra 트랙의 개수를 최소화하는 것이 중요하다. 이를 위해 주어진 n개의 element 각각을 하나의 클러스터로 보고, 이들 중 가장 강하게 연결된 두 개의 클러스터를 하나로 병합하는 과정을 반복하여 회전선형배열을 얻었다. 이 선형배열에서 인접한 element 사이를 절단하여 n개의 서로 다른 선형배열을 얻었으며, 이 중 트랙 개수를 최소화하는 선형배열을 최종 해로 선택하였다. 본 논문에서는 스펙트럴 방법을 이용하여 두 클러스터에 대응하는 벡터의 내적이 최대가 되는 클러스터들이 원래 그래프에서도 가장 강하게 연결됨을 보였으며 이를 이용해 병합할 두 클러스터를 구하였다. 제안된 알고리즘은 사용된 트랙수 면에서 GA/SA[2]와 유사하며 수행시간 면에서 수행시간 면에서 2800배 이상 빨랐다.

감사의 글

실험에 필요한 모든 예제와 기존 알고리즘을 제공한 LG 중앙연구소 임준서 박사에게 감사의 뜻을 전합니다.

참 고 문 헌

[1] Neil H.E. Weste and Kamran Eshraghian, Principles of CMOS VLSI design, Addison Wesley, p. 513, 1992.

[2] J.S.Yim and C.M.Kyung, "Datapath layout optimization using genetic algorithm and simulated annealing", IEE Proc. Comput. Digit. Tech., vol. 145, no. 2, pp. 135-141, March 1998.

[3] Y. Tsujihashi, H. Matsumoto, H. Nishimaki, H. Nako, O. Kitada, S. Iwada, S. Kayano, and M. Sakao, "A high-density datapath generator with stretchable cells", IEEE JSSC, 1994, 29, (1), pp. 2-7.

[4] H. Imahashi, K. Okujima, H. Ido, H. Ariyoshi, and I. Shirakawa, "Optimization algorithms of a datapath with reversible elements", Proceedings of IEEE APC-CAS, 1992, pp. 196-200.

[5] C.M.Kyung, "HK386: an x86-compatible 32bit CISC microprocessor", Proceedings of ASP-DAC'97, 1997, pp. 661-662.

[6] J. S. Yim, Y.H. Hwang, C.J.Park, H.Choi, W.S.Yang, H.S.Oh, I.C.Park, C.M.Kyung, "A C-based RTL design verification methodology for complex microprocessor", Proceedings of ASP-DAC, 1997, pp. 83-88.

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

[8] 성광수, "회로 결선도 분할을 위해 점진적 병합을 이용한 선형배열", 대한전자공학회 논문집 제 32권 C편 제 9호 9월 pp. 699-706, 1998년 9월

[9] M.R.Garey and D.S.Johnson, "Computers and intractability: a guide to the theory of NP-completeness", W.H.Freeman and Co., SF, 1979.

[10] S.Kang, "Linear ordering and application to placement", Proc. of the 20th DAC, pp. 457-464, 1983.

[11] M.K.Goldberg and M.Burstein, "Heuristic improvement technique for bisection of VLSI networks", Proc. IEEE Intl. Conf. Computer Design, pp. 12-125, 1983.

[12] T.Lengauer, Combinatorial algorithms for integrated circuit layout, Wiley-Teubner, 1990.

[13] C. Sechen, "An improved simulated annealing algorithm for row-based placement", Proc. ICCAD, pp. 478-481, 1987.

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

[15] 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.

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

저 자 소 개

成 枕 洙(正會員)

1990년 2월 한양대학교 전자공학과 졸업(B.S). 1992년 2월 한국과학기술원 전기 및 전자공학과 졸업(M.S). 1997년 2월 한국과학기술원 졸업(Ph.D). 1997년 3월부터 1998년 2월까지 미국 SandCraft 연구원 (MIPS R5400 마이크로프로세서 개발). 1998년 3월부터 1999년 2월까지 영남대학교 전기전자공학부 객원 교수. 1999년 3월부터 현재까지 영남대학교 전기전자 공학부 전임강사. 주관심분야는 마이크로프로세서 설계, DSP 칩 설계, 설계 자동화 그리고 화상회의 시스템 구현