2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

클러스터링 기법을 이용한 효과적인 회로분할 알고리즘

김동진, 배종국, 허성우
동아대학교 컴퓨터공학과
djkim123@chollian.net, jkbae@dspc.donga.ac.kr, swhur@daunet.donga.ac.kr

Efficient Circuit Partitioning Algorithm Using Clustering Technique

Dong-Jin Kim, Jong-Kuk Bae, Sung-Woo Hur
Dept of Computer Engineering, Dong-A University

요 약

회로분할 기법은 VLSI 칩 설계 시 핵심적인 기술로서 오랫동안 연구가 행해져 왔는데, 대부분의 회로분할 휴리스틱에서 Fiduccia-Mattheyses(FM) 알고리즘을 기본 기술로 사용하고 있다. 본 논문에서도 FM 알고리즘을 기본 분할 기술로 이용하되 선형배치 및 클러스터링 기법을 추가로 적용하여 효과적인 회로 분할 알고리즘을 제안한다. MCNC 벤치마크 회로를 이용하여 제안한 알고리즘과 FM 알고리즘을 실험적으로 비교하였다. 실험결과는 회로에 따라 적게는 14%, 많게는 57%까지 개선되는 것을 보여준다.

1. 서 론

회로분할 문제는 VLSI설계를 위한 CAD 분야에서 가장 중요한 문제 중의 하나로서 25년 이상 연구되어 왔다. 회로분할의 목적은 회로를 둘 이상으로 분할하되 각 컴포넌트의 크기는 미리 제시한 범위 내에 있도록 하면서, 컴포넌트 간의 컷(cut)을 최소화시키는 문제이다. 많은 휴리스틱들이 발표되었으나 반복-개선 (iterative improvement)에 기초한 방법이 주로 사용되어 왔다. 가장 널리 알려진 반복-개선 방법에 기초한 알고리즘으로는 Kernighan-Lin (KL)[1]과 Fiduccia-Mattheyses (FM)[2]이라고 볼 수 있다.

회로는 하이퍼그래프로 모델링가능하고, 그래서 회로분할 알고리즘의 대부분은 하이퍼그래프 분할을 위한 것이다. 회로의 각 소자는 하이퍼그래프에서 노드로, 회로의 시그널 넷은 하이퍼그래프에서 하이퍼에지로 모델링된다.

KL 알고리즘은 일반 그래프 모델에서만 적용가능하며, 서로 다른 분할에 속한 두 노드의 교체에 의해서만 컷이 개선되도록 설계된 반면, FM 알고리즘은 하이퍼그래프 모델에서도 적용가능하며, 한 노드만의 움직임도 허용하기 때문에 실제 칩 설계 시 적용가능하다. FM 알고리즘은 매우 효과적인 자료구조를 사용하기 때문에 실행시간도 결코 나쁘지 않는 알고리즘이다. 기본적으로 FM 알고리즘을 사용하고 있는 휴리스틱, 즉 변형된 FM 알고리즘이 그간 많이 발표되었지만 아직도 더 개선될 여지가 많이 있다.

Krishnamurthy[3]는 최대 이득을 갖는 노드가 여러 개 있을 때 지능적인 방법으로 한 노드를 선택하지 않으면 FM 알고리즘의 효율이 매우 나빠짐을 지적하였다. Hagen[4]과 그 외 많은 논문에서 LIFO 이득버켓(gain bucket)이 FIFO나 무작위(random) 버켓보다 해의 질이 우수함을 발표하였다.

시뮬레이티드 어닐링(simulated annealing) 기법, 클러스터링 기법, 네트웍 플로우를 기본으로 한 최대-흐름 최소-컷 알고리즘, 수치해석적 방법, 확률적 방법 등 수많은 기법들이 회로분할 문제를 위해 사용되어 왔다. Alpert등은 회로분할에 관한 문제를 심도있게 연구하였고[5], Hauck 등[6]은 여러 기법들을 실험적으로 비교 평가하였다. Alpert[7], Karypis [8] 등은 다단계 분할을 이용하면 성능이 개선됨을 보여 주었다.

1.1. 용어 설명

회로는 넷 리스트(net list)로 보통 표시된다. 넷이란 소자들 간에 공유하는 전기신호를 나타내는 선이라 볼 수 있다. 넷 리스트는 하이퍼그래프 $G(V, E)$로 나타낼 수 있는데 여기서 $V=\{v_1, v_2, \ldots, v_n\}$는 노드의 집합(회로에서 소자에 대응함), $E=\{e_1, e_2, \ldots, e_m\}$는 하이퍼에지의 집합을 나타낸다. 하이퍼그래프에서 각 에지 e는

IDEC에서 지원한 HW or/and SW를 사용하였음

V의 부분집합, 즉 $e \subset V$이다. 각 노드 v_i의 크기는 $s(v_i)$로 나타내고, 각 에지 e_i의 가중치는 $w(e_i)$로 나타낸다.

1.2. 연구목적 및 문제정의

본 논문에서는 하이퍼그래프로 표현된 회로를 두 개의 부분 집합 V_1과 V_2로 분할하되 컷 사이즈를 최소화시키는 효율적인 알고리즘을 제안한다. 각 부분에 속한 노드 크기의 합은 미리 정한 범위 내에 속해야 한다. 분할 P에 의해 컷되는 넷(하이퍼에지)은 $E(P)=E(V_1)$ (또는 $E(P)=E(V_2)$)로 나타낸다. 수학적으로 표현하면

$E(P)=\{e \in E \mid \exists u, v \in e \ s.t. \ u \in V_1 \text{ and } v \in V_2\}$ 로 표현될 수 있다. $S(V_i)=\sum_{v \in V_i} s(v)$로 나타내면 유사하게 $S(V)=\sum_{v \in V} s(v)$로 나타낼 수 있으며, 그러면 $S(V)=S(V_1)+S(V_2)$가 된다.

$W(P)=\sum_{e \in E(P)} w(e)$로 두자. 각 분할의 크기를 조정하기 위해 $r(0 \le r \le 0.5)$이 주어질 때 회로분할 문제는 다음과 같이 정의된다.

$\min W(P) \ s.t.$

$r \cdot S(V)-S_{\max} \le S(V_i) \le (1-r) \cdot S(V)+S_{\max}$

여기서 $S_{\max}$는 가장 크기가 큰 노드의 크기를 의미한다.

2. FM 알고리즘

하이퍼그래프 G에서 각 하이퍼에지에 속한 노드의 총합을 p라 하자. 즉, $p=\sum_{e \in E}|e|$이다. FM 알고리즘의 각 패스의 수행시간이 $O(p)$인데 이는 그림 1에서 보인 것처럼 잘 설계된 이득버켓의 자료구조 때문이다. 수행시간 개선의 핵심은 최대이득을 갖는 노드를 $O(1)$시간에 결정할 수 있는데 있다. 모든 이득이 정수이고, 각 이득은 $+\deg_{\max} \sim -\deg_{\max}$ 사이에 있기 때문에 효율적인 자료 구조가 가능하다.

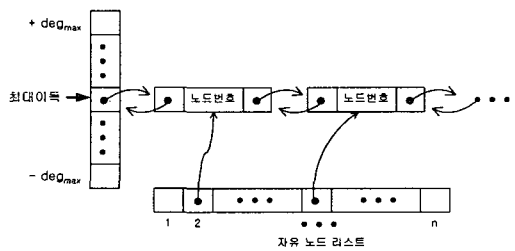

그림 1. 이득버켓의 자료구조

분할 V_1과 V_2 각각을 위해 이득버켓을 유지하며, 이들을 이용해 최대 이득을 갖는 자유 노드를 선택하여 상대편 분할로 옮긴다. 한 순간에 한 자유노드만 이동하고, 일단 이동된 노드는 그 패스 내에서는 더 이상 이동될 수 없도록 하기 때문에 자료구조의 갱신이 용이하다. 특히 자유노드를 유지하기 위해 그림 1에서 보는 것처럼 리스트를 유지하는 것은 실행시간 절약에 크게 기여한다.

FM 알고리즘의 개략을 그림 2에서 보였다. 최대 이득을 갖는 노드가 여러 개 존재하기 때문에 Hagen[4]의 실험에서처럼 여기서도 LIFO 방식으로 노드를 선택한다. 즉, 이동되는 노드는 해당 이득버켓 리스트의 처음에 삽입되고, 최대 이득 노드를 선택할 경우 최대 이득버켓 리스트의 첫 노드를 선택한다.

FM 알고리즘

```
While(TRUE)
    gain ← Single-Pass-of-FM()
    if(gain ≤ 0) Terminate
EndWhile
```

Single-Pass-of-FM()

```
Compute gains of all ndoes
While(moves of nodes are possible)
    (1) Select free node v
    (2) Update the data structure to reflect the
        tentative move of v
    (3) Extend the log of tentative moves
End while
Compute the prefix of the sequence of tentative
moves that achieves the maximum gain
return gain
```

그림 2. 개략적인 FM 알고리즘

3. 클러스터링을 이용한 회로분할 알고리즘

본 절에서는 클러스터링 기법을 이용한 2 단계 회로 분할 알고리즘을 제안한다. 클러스터링 기법은 결과의 질을 높일 뿐 아니라 실행 시간도 단축시킬 수 있는 기법으로써 이미 [7], [8]등에서 채택되어 사용되었다. 그러나 지금까지 발표된 대부분의 논문에서는 회로의 특성을 분석하여 연관도가 높은 노드들을 클러스터링하는 기법을 사용하였다. 이렇게 할 경우 연관도 높은 노드들의 집합이 미리 회로에 의해 결정되기 때문에 매 실행 시 서로 다른 압축 회로를 얻을 수 없다.

이런 문제를 극복하기 위해 본 논문에서는 회로의 특성을 분석하여 연관있는 노드들을 클러스터링하지 않고, 대신 매우 효과적인 다양한 방법으로 각 노드들을 선형배치한 후 선형배치된 결과로부터 인접한 노드들을 클러스터링하여 압축된 회로를 얻는다. 그 압축된 하이퍼그래프에 FM 알고리즘을 적용하여 분할을 구한다. 압축된 하이퍼그래프로부터 구한 분할이 더 이상 개선되지 않으면, 그것을 원래의 하이퍼그래프로 복원한다. 이 때 분할정보도 같이 이용하여 복원함으로써 원 하이퍼그래프에 대한 분할을 얻는다. 복원된 하이퍼그래프와 분할정보를 기초로 FM 알고리즘을 다시 적용한다.

제안한 알고리즘은 먼저 압축 하이퍼그래프에서 FM 알고리즘을 적용하고, 다음 원 하이퍼그래프에

서 FM 알고리즘을 또 적용하기 때문에 2 단계 회로분할 알고리즘으로 볼 수 있다. 그림 3에서 제안한 알고리즘의 개요를 보였다.

클러스터링을 이용한 회로분할 알고리즘
1. Find a linear ordering for all nodes 2. Construct a clustered hypergraph Gc 3. Apply FM algorithm to Gc 4. Restore the original hypergraph G 5. Apply FM algorithm to G 6. Return the cut-size

그림 3. 클러스터링 기법을 이용한 분할 알고리즘

제안한 알고리즘 (그림 3) 단계 1에서 우선 모든 노드에 대해 순서(ordering)를 구한다. 이 순서를 구하기 위해 그림 4에서 보인 바와 같이 Alpert와 Kahng[9]이 제시한 알고리즘을 이용한다.

순서 알고리즘
1. Choose a node v_i and set $\pi(1)$=i. Set *index*, the current size of S, to 1. For each $v_j \in V-S$, compute *Attract*(j). 2. If $V-S \neq \emptyset$, choose $v_i \in V-S$ with optimal *Attract*(i), otherwise exit. 3. Increment *index* and set $\pi(index)$=i. Update *Attract*(j) for each $v_i \in V-S$ and go to Step 2.

그림 4. 순서 알고리즘

S는 이미 순서가 결정된 노드의 집합, 즉 $S=\{v_j \in V \mid \exists\, \text{index}, \pi(\text{index})=j\}$이다. 순서알고리즘 단계 1에서 임의의 한 노드를 선택하여 S에 넣은 다음, 나머지 모든 노드들에 대해 v_i에 대해 $Attract(i)$를 구한다. 그리고 단계 2에서 최적의 $Attract$값을 갖는 노드를 선택하여 S에 추가한다. 단계 3에서는 나머지 노드들에 대해 $Attract$을 갱신한 후 단계 2, 3을 반복한다.

$\text{Nets}(i)=\{e \in E \mid v_i \in e\}$는 v_i에 인접한 넷 집합을, $Adj(i)=\{v_j \in e \cap S \mid e \in \text{Nets}(i)\}$란 순서가 결정된 노드들의 집합 S에 속한 원소 중 노드 i 에 인접한 노드들의 집합을 나타낸다.

$Attract(i)$를 어떻게 결정하는가에 따라 노드 순서가 달라진다. 본 논문에서는 $Attract(i)$를 계산하기 위해 아래 5가지 서로 다른 함수를 구현하였다.

- **DFS(Depth First Search) Ordering**:
  $Attract(i)=\max\{j \mid v_{\pi_j} \in Adj(i) \cap S\}$
- **BFS(Breadth First Search) Ordering**:
  $Attract(i)=\min\{j \mid v_{\pi_j} \in Adj(i) \cap S\}$
- **Max-Adjacency Ordering**:
  $Attract(i)=|\{e \in \text{Nets}(i) \mid e \cap S \neq \emptyset\}|$
- **Absorption Objective**:
  $$Attract(i)=\sum_{(e \in \text{Nets}(i) \mid e \cap S \neq \emptyset)} \frac{1}{|e|-1}$$
- **Scaled Cost Objective**:
  $$Attract(i)=\sum_{e \in Adj(i)} \frac{|S \cap e|}{|e|-1}$$

그림 3의 단계 2에서 여러 노드를 클러스터링하여 슈퍼노드를 생성하기 위해 본 논문에서는 Hur와 Lillis[10]가 제안한 방법을 이용하였다. 즉, 주어진 파라메터 L과 U를 이용하여, 이미 구한 선형순서에서 인접한 노드를 클러스터링하는데 한 슈퍼 노드에 포함되는 원 노드의 개수는 최소 L, 최대 U개가 되며, 경계의 결정은 넷 밀도가 가장 적은 곳이 된다. 자세한 정보는 [10]을 참조하기 바란다. 노드 순서가 동일하다 하더라도 파라메터 L과 U의 값에 따라 서로 다른 압축 하이퍼그래프를 얻게 된다.

4. 실험 및 고찰

본 논문에서 제안한 클러스터링 기법을 이용한 회로 분할 알고리즘은 C 언어로 구현되어 Linux Pentium III 733MHz 플랫폼 상에서 실행하였다. 실험에 사용된 회로는 MCNC 표준 벤치마크로서 각 회로의 사양은 표 1에서 보인 바와 같이 노드의 개수가 작은 것은 1952, 큰 것은 29347이다.

회로이름	# nodes	# nets	# pins
struct	1952	1920	5471
biomed	6514	5742	21040
industry2	12637	13419	48158
industry3	15433	21939	65919
avq small	21918	22124	76230
avq large	25178	25383	82748
ibm05	29347	28446	126380

표 1. 회로 정보

제안한 알고리즘의 효율은 크게 두 파라메터에 의해 결정된다. 첫째, 노드의 순서를 결정하기 위해서는 필수적으로 구해야 하는 $Attract$ 값을 구하기 위해 어떤 함수를 사용할 것인가 하는 것이고, 둘째로 압축 하이퍼그래프의 압축률을 결정하는 L과 U의 값을 어떻게 결정하는가 하는 점이다. 이 값들이 크면 압축 하이퍼그래프의 복잡도는 줄어들게 되어 실행시간이 개선되나 너무 크면 한 슈퍼 노드에 포함된 원 노드들의 연관도가 떨어지는 단점이 있다. 실험에서 L은 1과 10사이의 값 중 하나로 선택되고, $U \approx 2 \cdot L$이 되도록 두었다.

각각의 회로에 대해 FM알고리즘을 100회 실행한 후 최고로 좋은 결과(즉, 최소의 컷 값을 갖는 분할)와 평균값을 구하여 보였고, 제안한 알고리즘에서 사용하는 서로 다른 $Attract$ 함수 각각에 대해, 세 가지 다른 L, U의 값 쌍에 대해 각각 100회 실험을 하여 최고값과 평균값 FM의 결과와 비교함으로써 제안한 알고리즘이 효율적임을 보였다.

L, U 값은 (2,5), (5,9) 그리고 (10,20)로 두어 각각 실험했는데 지면 관계상 모든 결과를 다 보일 수 없어 그 중 결과가 상대적으로 비교적 양호한 $L=10, U=20$인 경우에 대해 표 2에서 보였다. 표

	FM		Attract 함수									
			Absorption		BFS		DFS		Max-Adj		Scaled Cost	
	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg
struct	43	58.9	36	51.3	40	55.6	37	46.9	**35**	55.1	**35**	52.8
biomed	109	202.5	**83**	115.0	88	165.2	86	122.4	**83**	131.9	**83**	144.3
industry2	436	883.7	308	436.8	241	370.3	**186**	293.6	290	449.5	255	399.6
industry3	259	543.4	200	461.5	192	393.5	**188**	256.1	189	460.1	190	402.4
avq_small	354	619.6	347	477.0	233	690.6	**193**	408.4	567	733.0	232	528.9
avq_large	366	811.7	398	533.2	253	639.1	**227**	439.3	411	715.6	365	584.7
ibm05	2282	3446.9	1857	2427.4	1791	2295.0	1748	1895.2	1722	2351.0	**1712**	2373.8

표 2. 실험 결과: (L,U)=(10,20)일 때 최소 및 평균 컷 사이즈

에서 보다시피 *Attract* 함수에 따라 결과가 크게 영향을 받는 회로(예를 들어 avq 회로)도 있음을 알 수 있다. *Attract* 함수가 DFS일 경우 모든 결과가 FM보다 우수함을 알 수 있다. $(L, U) = (2, 5)$ 또는 (5,9)인 경우에도 회로에 따라서, 또 *Attract* 함수에 따라 (10,20)인 경우보다 결과가 좋은 것도 나타났다. 실행 시간은 제안한 알고리즘이 FM 알고리즘보다 평균적으로 10%~20% 개선된 것으로 나타났다.

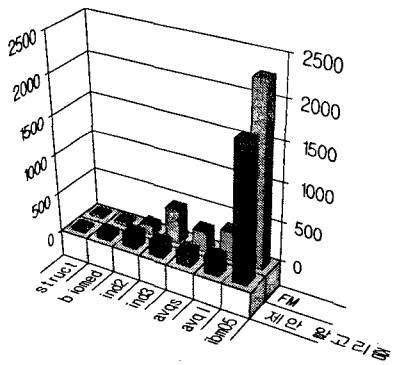

그림 5. Attract 함수 DFS와 FM의 결과비교

그림 5에서는 표 2의 자료 가운데 *Attract* 함수가 DFS인 경우의 best 결과와 FM의 best 결과를 가시적으로 쉽게 비교할 수 있도록 그래프로 나타내 보였다.

5. 결 론

본 논문에서는 선형순서 기법과 클러스터링 기법을 FM 알고리즘에 접목시킨 2 레벨 회로 분할 알고리즘을 제안하였고, 실험 결과도 제안한 알고리즘이 매우 효과적임을 보여 준다.

좀 더 정교하게 고안된 클러스터링 기법과 [8]에서처럼 m-레벨 ($m \geq 2$) 알고리즘을 사용한 v-cycle 기법을 사용한다면 결과는 더 개선될 것으로 기대된다. 또한 [10]에서처럼 넷트웍 플로우 기반 서브그래프(subgraph) 추출 기법을 클러스터링 기법과 같이 이용하는 것도 분할문제를 더 효율적으로 해결할 수 있는 한 방법으로 사료된다. 이런 문제들은 앞으로의 연구 과제로 남겨 둔다.

참고문헌

[1] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," Bell Syst. Tech. J., vol. 49 no. 2, pp. 291-307, 1970.

[2] C. M. Fiduccia and R. M. Matteyses, "A Linear Time Heuristic for Improving Network Partitions," in DAC, pp. 175-181, 1982.

[3] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Network," IEEE Trans. on Computers, vol. 33, no 5, pp.438-446, 1984.

[4] L. Hagen *et al*, "On Implementation Choices for Iterative Improvement Partitioning Algorithm, " in European Design Automation Conference, pp. 144-149, 1995.

[5] C. J. Alpert and A. B. Khang, "Recent Directions in Netlist Partitioning: A Survey," Integration: the VLSI J., pp. 1-18, 1995.

[6] S. Hauck and G. Borriello, "An Evaluation of Bipartitioning Techques," IEEE Transactions on CAD, vol.16, no. 8, pp. 849-866, 1997.

[7] C. J. Alpert, J. Huang and A. B. Kahng, "Multilevel Circuit Partitioning," in Proc. DAC, pp. 530-533, 1997.

[8] G. Karypis *et al*, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain," in Proc. DAC, pp. 526-529, 1997.

[9] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, With Applications to Circuit Clustering," in Proc. IEEE Trans. on CAD, pp. 63-67, 1994.

[10] Sung-Woo Hur and John Lillis, "Relaxation and Clustering in a Local Search Framework: Application to Linear Placement", Proc. of DAC, pp.360-366, 1999.