GWW 휴리스틱을 이용한 회로 분할

박경문,[0] 오은경, 허성우
동아대학교 컴퓨터 공학과
interfaces@hanmail.net, ekoh@mail.donga.ac.kr, swhur@daunet.donga.ac.kr

Circuit Partitioning Using "Go With the Winners" Heuristic

Kyung-Moon Park[0] Eun-Kyung Oh Sung-Woo Hur
Dept. of Computer Engineering, Dong-A University

요 약

회로분할 기법은 VLSI 설계뿐만 아니라 많은 분야에서 응용될 수 있어 오랫동안 연구가 행해졌다. 대부분의 회로분할 휴리스틱에서 Fiduccia-Mattheyses(FM) 방법을 핵심 기술로 사용하고 있다. 회로 분할 문제는 또한 다른 컴비네토리얼 문제에서처럼 해 공간에서 최적해를 찾는 문제로 볼 수 있는데, GWW(Go With the Winners) 방법은 해 공간을 검색하는 성공적인 패러다임 중의 하나이다.

본 논문에서는 "GWW" 패러다임을 FM 방법에 접목시켜 회로를 분할하기 위한 휴리스틱을 제안한다. MCNC 벤치마크 회로를 이용하여 전형적인 FM 방법에 의한 결과와 "GWW" 패러다임을 접목하여 얻은 결과를 비교하였다. 실험결과는 매우 고무적이다.

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. KL 알고리즘은 일반 그래프 모델에서만 적용가능하며, 노드-쌍 교체에 의해서만 컷이 개선되는 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 어니링(simulated annealing) 기법, 클러스터링 기법, 최대-흐름 최소-컷 알고리즘, 수치해석적 방법, 확률적 방법 등 수많은 기법들이 회로분할 문제를 위해 사용되었다. Alpert등은 회로분할에 관한 문제를 심도있게 연구하였고[5], Hauck 등[6]은 여러 기법들을 실험적으로 비교 평가하였다.

IDEC에서 지원한 HW and/or SW를 사용하였음

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 $V=\{v_1,v_2,...,v_n\}$ 는 노드의 집합(회로에서 소자에 대응함), $E=\{e_1,e_2,...,e_m\}$ 는 하이퍼에지의 집합을 나타낸다. 하이퍼그래프에서 각 에지 e는 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 휴리스틱을 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. $W(P)=\sum_{e\subset E(P)} w(e)$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, $p = \sum_{e \subset E} |e|$ 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. 분할 V_1과V_2 각각을 위해 이득버켓을 유지하며, 이들을 이용해 최대 이득을 갖는 자유노드를 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 수행시간 절약에 크게 기여한다.

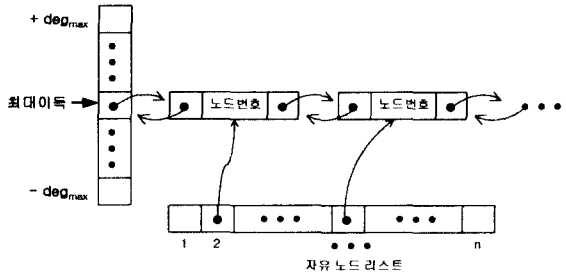

그림 1. 이득버켓의 자료구조

FM 알고리즘의 개략을 그림 2에서 보였다. 최대이득을 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 nodes
While(moves of nodes are possible)
    (1) Slect a 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. "Go With the Winners" 알고리즘

GWW(Go With the Winners) 알고리즘은 트리 구조를 갖는 공간을 검색하기 위한 것으로, Aldous와 Vaziarni[7]에 의해 처음 소개되었다. 그후 Dimitriou와 Impagliazzo[8]가 그래프 구조를 갖는 공간도 검색할 수 있도록 "GWW" 휴리스틱의 변형을 제안하였다. 이 휴리스틱의 동작 패러다임은 다음과 같다.

휴리스틱을 동시에(simutaneously) 수행할 수 있도록 여러 개의 인스턴스 혹은 runs를 생성하여 실행한다. 즉, 다수의 run을 만들어 해 공간을 검색한다. 각각의 run은 실행 도중 가끔씩 국부최적(local optimum)에 도달하는데, 이 때 마다 그 중간 결과가 "우수한"것과 "나쁜" 것으로 각 run을 분류한다. 그리고 중간 결과가 나쁜 것은 해 공간 검색을 중단(즉, 좋은 결과를 찾는 작업을 중단)하고 대신 중간 결과가 좋은 것으로 이동하여 그 지점에서 검색을 계속한다. 그림 3에서 이런 동작 원리의 개요를 보여준다.

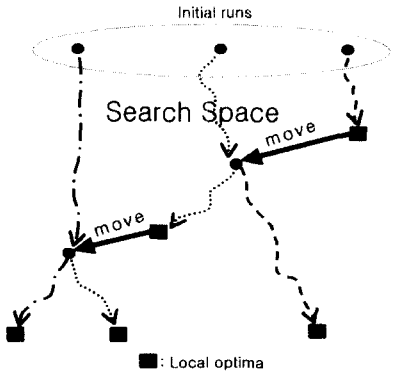

그림 3. "GWW" 휴리스틱 동작원리

본 연구에서는 "GWW" 패러다임을 FM 알고리즘에 다음과 같이 적용하였다.

(1) k 개의 초기 분할을 구한 후 각각에 대해 FM 알고리즘을 적용하여 수행하라. 즉, k 개의 run에 대해 **Single-Pass-of-FM** 함수를 병행 수행하라.

(2) 만약 한 개 이상의 run이 국부최적에 수렴하면 그 때의 결과를 기록하고, 아직 active한 다른 run으로 이동하여 수행을 계속하라.

(3) 모든 run이 국부최적에 수렴할 때까지 과정(2)를 실행하라.

단계 (2)에서 run R_i에서 다른 run Rj로 옮긴다는 말은 다음과 같다. run R_i는 이미 국부최적에 수렴하였으므로 더 이상 개선될 여지가 없다. 그때의 컷 사이즈 및 관련된 정보를 기억한 후, R_j의 현재 분할 상태를 R_i로 복사한 후, 이득 버켓을 서로 다르게 생성한다. 그런 다음 두 run R_i, R_j가 계속 병행 수행된다.

k개의 run을 실행할 때 각각의 이득버켓에 연결된 리스트 관리를 다르게 함으로써 k 개의 run 동작 방법이 실제적으로 다르게 되도록 하였다.

또한 실행 시간 절약을 위하여 어느 정도의 시간이

흐른 후부터는 한 개의 run이라도 중간결과가 상대적으로 좋지 않으면 그 run은 수행을 중단시키고 그 때 최고로 좋은 run으로 이동시킨다. 그림 4에서 "GWW" 알고리즘을 보였다.

"GWW" 알고리즘

```
Make k independent initial partitions, P1, P2,..., Pk
While(TRUE)
   PARBEGIN //run k runs simultaneously
      gaini ← Single-Pass-of-FM(Pi)
   PAREND

   if(All runs converge) then
      Report the best result
      Terminate
   endif

   if(gaini ≤ 0) then
      Find the best active run Rk
      Pi ← Pk // copy Pk to Pi
      if(∃Pj which is relatively too worse) then
         Pj ← Pk // copy Pk to Pj
      endif
   endif
End while
```

그림 4. "GWW" 알고리즘

4. 실험 및 고찰

"GWW" 알고리즘은 C 언어로 구현되어 Linux Pentimu III 733MHz 플랫폼 상에서 실험하였다. 실험에 사용된 회로는 MCNC 표준 벤치마크로서 각 회로의 사양은 표 1에서 보인 바와 같이 노드의 개수가 작은 것은 1952개, 큰 것은 29347개이다. 각 run을 병행 수행하는 대신 우리는 순차적으로 k 개의 run을 실행하였다.

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 사양

제안한 알고리즘의 효율성을 보여 주기 위해 우리는 각각의 회로에 대해 FM알고리즘을 100회 실행하고, 각 회로에 대해 사용된 총 CPU 시간을 구한다. 각 회로에 대해 같은 양의 CPU 시간을 "GWW" 알고리즘을 위해 사용한 다음 최고로 좋은 결과를 서로 비교하였다. 즉, 거의 동일한 CPU 시간을 사용하여 FM 알고리즘과 GWW 알고리즘을 비교함으로 공정한 비교가 되도록 하였다.

Run의 수는 다양하게 하여 실험해 보았으나 동일한 CPU 시간을 사용할 경우 결과는 큰 차이가 없었음을 실험적으로 알 수 있었다. 표 2에서 우리는 k=3인 경우의 실험 결과를 보였다. 총 CPU 시간단위는 sec.이다.

실험 결과에서 보듯이 제안한 GWW 알고리즘이 모든 회로에 대해 FM알고리즘의 결과보다 우수하였다.

회로이름	FM 알고리즘		GWW 알고리즘	
	best	total CPU	best	total CPU
struct	43	5.55	39	5.65
biomed	109	41.32	84	42.73
industry2	436	118.01	355	115.16
industry3	259	199.16	189	191.19
avq small	354	189.18	274	188.78
avq large	366	229.54	319	239.62
ibm05	2282	973.11	1999	1007.19

표 2. 실험 결과 (k=3)

5. 결 론

본 논문에서는 "GWW" 패러다임을 FM 알고리즘에 접목시킨 회로 분할 알고리즘을 제안하였고, 실험 결과도 제안한 알고리즘이 매우 효과적임을 보여 준다.

본 논문에서 다른 기법(예를 들어 클러스터링, 초기 분할을 위한 기법 등)을 사용하지 않았으나, 이런 다른 기법들을 같이 사용한다면 훨씬 더 좋은 결과를 얻을 수 있을 것으로 예측된다. 또한 "GWW" 패러다임은 CAD 관련 다른 문제에도 매우 유용하게 사용될 수 있을 것으로 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] D. Aldous and U. Vazirani, "Go With the Winners' Algorithms," in 35th Annual Symposium on Foundations of Computer Science, pp. 492-501, IEEE, 20-22 Nov. 1994.

[8] T. Dimitriou and R. Impagliazzo, "Towards an Analysis of Local Optimization Algorithms, "in Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory