타이밍이 고려된 배치를 위한 기하적인 제약조건 탐색

이 재 훈, 조 준 동
성균관대학교 정보통신공학부
{ dlwogns, jdcho }@skku.ac.kr

Geometric Constraints Exploration for Timing-Driven Placement

Jaehoon Lee, Jundong Cho
School of Electrical and Computer Eng., Sungkyunkwan University

요 약

고성능 VLSI 설계 시 배치 후를 포함한 전체적인 설계 과정이 완성되기 전까지는 물리적인 정확한 설계의 특성은 배치 단계에서는 알기 어렵다. 따라서 주어진 성능 (시간적 제약조건)을 만족하는, 즉, timing-driven placement (타이밍이 고려된 배치)는 1.0 미크론 이하의 초미세한 설계에서 중요하게 되었다. 타이밍을 고려한 배치는 초기 레이아웃 디자인 단계에서 타이밍 제약조건에 의해 디자인 반복을 줄인다. 하지만 대부분의 배치 단계의 디자인 모델은 배치단계에서 기하학적인 면을 고려하여 최대허용 지연시간 (slack 이라고 부름) 과 같은 물리적인 디자인 효과를 분석하기 어려운데 이것은 물리적으로 정확한 특성이 이 단계에서 알려지지 않기 때문에 당연한 결과이다.

본 논문에서는 기하적인 요소를 고려한 slack의 재분배의 이점을 이용하여 허용 지연시간 처리의 혁신적인 방법을 제안한다. 제안된 접근법은 timing-closed 솔루션을 쉽게 찾는데 도움을 주고 이는 디자인을 반복하는 시간을 절약할 수 있게 한다.

1. 서 론

VLSI 기술의 발전에 따라 칩에서 모듈의 조밀도는 증가하고 이는 결국 회로에서 총 지연이 모듈간의 상호연결에서의 지연에 의해서 지배된다. 물리적인 설계 과정의 첫 번째 단계는 칩의 배치이다. 수년에 걸쳐 다양한 배치 알고리즘이 발전되었다[1]. 1미크론 이하의 초미세 설계에서 상호연결의 타이밍에 대한 고려는 초기 배치 단계 동안 설계의 효율성을 위해 널리 채택되었다. 또한, 과도한 공간 밀집은 향후 라우팅을 할 때 어려움을 유발할 수 있으며[1,3,5], 고속 신호 통신망에서 잠재적인 혼선에 의한 잡음을 증가시킨다. 과도한 공간 밀집은 커패시턴스 커플링으로 인해 전력 손실을 유발한다. 라우팅 설계 전 타이밍 분석에서 라우팅은 보통 최소한의 직선 Steiner 트리라는 것으로 가정한다.[2] 밀집으로 인해 이 라우팅 트리의 커패시턴스는 최소한의 Steiner 트리인 커패시턴스보다 더욱 커진다. 따라서, 밀집된 공간을 방지하기 위해서 충분한 slack을 부여하는 타이밍을 고려한 배치는 많은 연구가 진행되고 있다.[1,3,5] 본 논문에서는 타이밍을 고려한 분할과 배치단계가 효율적으로 고려된 새로운 기하학적인 특성을 분석하고 slack을 최대화 하여 배치단계에서 지연시간 오류가 발생할 확률을 줄일 수 있는 방법을 제시한다. Slack은 배선 프로그램에서 망(net)의 요구된 지연시간 (성능 요구조건)과 실제 지연간의 차이이다. 그리고 양의 slack은 요구된 제약시간 조건을 만족하도록 물리적인 레이아웃이 실현된다는 것을 의미한다. 반면에, 음의 slack은 레이아웃이 타이밍 요구 조건을 만족하지 않은 상태를 말한다. 허용 지연시간은 배치와 라우팅 단계 전에 지연된 slack을 배치하는 것을 말한다. 이런 타이밍을 고려한 레이아웃 방법들은 배치할 때 타이밍과 물리적인 요구사항을 동시에 만족하려고 노력한다. 여기서 주요 어려운 점은 타이밍을 고려한 레이아웃에서 수많은 계산의 복잡성이다. 이 문제는 제약조건 수를 줄이기 위해서 순환적 배치나 2차 방정식 프로그래밍 문제를 라그란지안 문제로 전환하는 선형프로그램 같은 모델이 될 수 있다. 하지만 이 최적화 과정은 1미크론 이하의 초미세 설계에서는 매우 복잡하고 시간이 많이 소요될 수 있다.

이러한 사실들에 의해서 타이밍을 만족하는 배치를 위해서는 상각 부등식 같은 기하학적 정보의 장점을 고려한 허용 지연시간 처리 기술을 적용시켜야 한다

본 논문의 구성은 다음과 같다. 2장에서는 문제에 대한 정의를 명확하게 기술하고, 지연시간 처리 방법의 전체적인 방법을 논한다. 3장에서는 기하학적 제약조건과 선형성 slack 을 재 구성 (rebudgeting이라고 함)하는 알고리즘에 대해서 소개한다. 마직막으로 제 4장과 5장에서 실험 결과와 결론을 제시한다.

2. Problem Formulation: the New Problem

타이밍을 고려한 배치 문제의 입력은 모듈과 망(net)들의 집합이다. 각 모듈들은 고정된 모양과 터미널 위치를 갖는다. 칩에 위치한 각각의 모듈의 최적 위치를 찾는 것이 목표이다. 타이밍을 고려한 배치의 적절한 해결책은 다음의 배치 제한 조건이 따른다. (1) 매크로는 오

버래핑 없이 적절한 위치에 있어야 한다, (2) 상호연결이 가능한 충분한 공간이 있어야 한다, (3) 타이밍 제한 조건이 만족되어야 한다. 일반적 타이밍을 고려한 배치에서는 망의 효율적인 타이밍과 그 밖의 망 길이와 공간 효율성을 최적화하는 것을 목표로 한다.

3. Geometric-constraint Slack Rebudgeting

삼각부등식법칙과 타이밍 제약조건을 따르는 타이밍 허용 지연시간 문제를 계산하기 위해서 삼각부등식을 유지하는 반면 총 타이밍 slack을 최대화 하는 것을 목적으로 삼는다. 문제는 아래와 같이 나타내어질 수 있다.

Geometry-constrained Slack Re-budgeting (GSB)
유도된 회로 그래프 G=(V,E)에서 노드는 r(V):V->Z로 edge는 r(E):E->Z로 표시되는데, r(V)와 r(E)는 각각 V와 E에서의 재조정된 변화이고, Z는 정수의 집합이다. GSB의 목적은 총 타이밍 slack를 최대화 하는 것으로, 그래프에서의 매 사이클에 대해 삼각부등식법칙을 필요로 한다.

3.1 Geometry-constrained Slack Re-budgeting based on Minimum Spanning Tree

우선, Kruskal의 신장 트리(spanning tree)를 만드는 것을 기반으로 한 알고리즘을 소개 할 것이다. Kruskal의 신장 트리는 트리를 설계하는 동안 edge 값들을 삼각부등식에 만족 시키기 위해 체크하고 업데이트한 그래프로 부터 만들어 진다. 기본적으로, 허용 지연시간 재할당의 타이밍에서 Deque*(Q)*는 우선순위 (priority queue) Q로부터 그래프 G의 가장 작은 edge (cheapest edge) *(u,v)*를 출력하는 것이고, *Enque (Q,s(u,w))*는 새로운 값인 *s(u,w)*를 Q에 입력하는 것이다.

Algorithm: Geometry-constrained Slack Re-budgeting based on Minimum Spanning Tree

Input: G=(V,E) with s(e), ∀(e)⊆E

Step 1: maintain a priority Q that initially contains graph edges with weights (timing budgets) in their ascending order.

Step 2: (based on Kruskal's Minimum Spanning Tree)

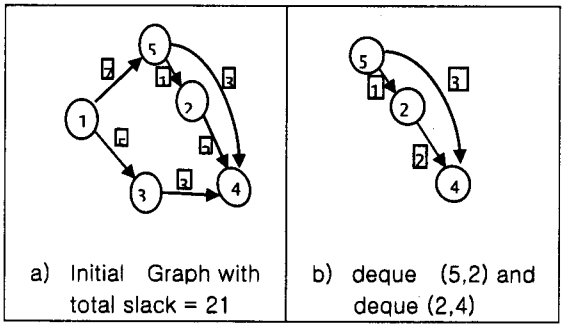
Until Q is empty
s(u,v) = Deque(Q);
If the edge (u,v) is incident to another edge (v,w) that has been already traversed and they form a cycle with another edge (u,w) (that has not yet been traversed) in G.
If (u,w) in E
then s(u,w) = s(v,w) + s(u,v);
else
until (u,w) in E
s(u,w) = s(v,w) + s(u,v);
E = E ∪(u,w); /* triangulation */
if s(u,w) ≥ [s(v,w) + s(u,v)] then
s(u,w) = s(v,w) + s(u,v)

알고리즘은 그림 1.a)~d)에 설명되어있다. GSA 알고리즘 각 단계에서 가장 작은 edge는 Q로부터 선택되어진다. 알고리즘의 시간 복잡도는 O (|E|log|E|+|N|^2)이다.

그림 1에서 보여주는 GSA 실행 후의 예에서 총 slack은 삼각부등식법칙을 충족하기 위해 18에서 15로 감소했다. 이것은 더욱 제한된 허용 지연시간 타이밍의 결과를 초래하여 바람직하지 않으므로 적합한 타이밍이 고려된 배치로의 해결방법을 찾는데 덜 유용하다. 삼각부등식 제한 조건으로 인한 허용 지연시간의 총 타이밍이 줄어드는 것을 피하기 위해서 다음과 같은 re-budgeting 전략이 사용 될 수 있다.

3. 2 Geometry-constrained slack Re-budgeting based on Modified Retiming technique

이제, 위에서 서술 된 Geometry-constrained Slack Re-budgeting 문제에 대한 더욱 좋은 해결책을 찾아보도록 한다. 그러기 위해서는 Leiserson과 Saxe[4]에 의해 주어진 본래의 retiming 공식과 알고리즘을 수정한다. 원래, retiming은 회로의 입/출력 특성에 영향을 주지 않고 회로에서 지연되는 요소들을 바꾸는데 사용된다. Retiming이란 초기 입력부터 초기 출력까지의 주기(cycle)또는 유도된 경로(path)에서 지연 시간이 변경되지 않는다.



a) Initial Graph with total slack = 21

b) deque (5,2) and deque (2,4)

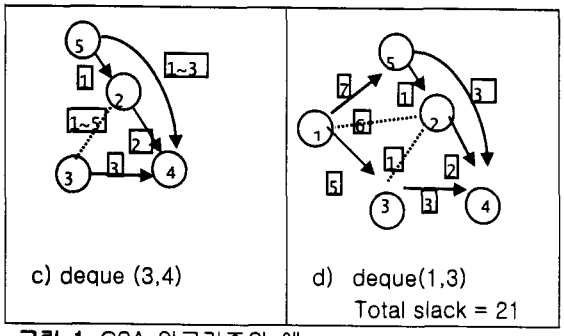


그림 1. GSA 알고리즘의 예

Re-budgeting은 Z가 정수의 집합일 때 노드 r:V→Z의 표시이다. Re-budgeted 노드 표시r(v)는 출력으로부터 입력을 향해 움직인 타이밍 slack의 양을 나타낸다. Re-budgeting 후 edge (u,v)의 타이밍 slack은r(u,v)=s(u,v)+r(v)-r(u)로 나타내어졌다. 그림 2 참조.

즉, *r(v)*는 *v*의 fanout으로부터 *s(u,v)*로 추가 된 타이밍 slack에 해당한다. 유사하게, *r(u)*는 *s(u,v)*에서 감해지고 *u*의 fanin으로 옮겨진 타이밍 slack에 해당한다. 이런식으로, re-budgeting은 주기 또는 경로의 총 slacks를 유지한다.

Re-budgeting 후의 회로의 총 타이밍 slack을 다음과 같이 나타내자.

$$r(G) = \sum(s(e)+r(v)-r(w)) =$$

$$S(G)$$

$$+ \sum(|FO(v)|-|FI(v)|+1)r(v) \ (if \quad |FO(v)| \geq |FI(v)|)$$

$$-\sum(|FI(v)|-|FO(v)|+1)r(v) \quad (if \quad |FI(v)| > |FO(v)|),$$

|FI(v)|와 |FO(v)|는 fanins의 수와 노드 *v*의 fanout이다. 최대의 re-budgeting 문제는 타이밍과 삼각부등식을 조건으로 하여 극한으로 증가하는 r(G)로 표시 될 수 있다.

그러면, 최대 re-budgeting 문제는 Leiserson's Min-Area Retiming[14]을 유사하게 변경하여 다음과 같이공식화 할 수 있다.

Minimize: $\sum(|FO(v)|-|FI(v)|+1)r(v) \ (if \quad |FO(v)| \geq |FI(v)|)$

$$-\sum(|FI(v)|-|FO(v)|+1)r(v) \quad (if \quad |FI(v)| > |FO(v)|)$$

Subject to

Constraint 1 (retiming)

$$s(u,v) + r(v) - r(u) \geq 1;$$

$$r(i) - r(h) \geq 0, \quad \forall i \in PI(G);$$

$$r(h) - r(o) \geq 0, \quad \forall o \in PO(G);$$

Constraint 2 (min-rule)

$$r(u) \leq r_{min}(u),\ r(v) \leq r_{min}(v), r(w) \leq r_{min}(w)$$

타이밍 한계의 양 또는 음의 이득을 포함한 re-budgeting은 다음의 두 과정에 의해 행해 질 수 있다.

If |FI(u)| > |FO(u)| (Figure 6b: budget increasing)

then, r(u) ≤ s(u,v);

$$r(u,v) = s(u,v) - r(u) + r(v);$$

$$r(i_1, u) = s(i_1, u) - r(i_1) + r(u);$$

$$r(i_2, u) = s(i_2, u) - r(i_2) + r(u);$$

If |FO(v)| ≥ |FI(v)| (Figure 6c: budget decreasing)

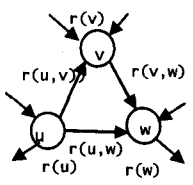
then, $r(v) \leq r_{min}(v)$ (=min(s(e)), $e \subseteq FO(v)$);

$$r(u,v) = s(u,v) + r(v) - r(u);$$

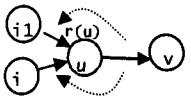
$$r(v, i_1) = s(v,i_1) - r(v) + r(i_1);$$

$$r(v, i_2) = s(v,i_2) - r(v) + r(i_2);$$

그림 2은 삼각부등식을 유지하면서 re-budgeting에 사용되는 변수들을 묘사하는 회로 그래프를 설명한다. 이 변수들은 삼각부등식을 만족하는 문제 해결방법을 찾는데 요구된다.



a)Variables used in Re-budgeting



b) Budget increasing | c) Budget decreasing

그림 2. Re-budgeting 에 대한 임의의 과정

위의 re-budgeting 전략에 따르면, 그래프 G에서 매 주기동안 삼각부등식을 만족하는 Geometry-constrained Re-budgeting 문제는 아래에 제시된 정수 선형 프로그램(integer linear program)과 같이 공식화 될 수 있다. Re-budgeting 동안, 각 노드 i는 r(i)로 표시된 "slack-retiming variable"로 정해졌고 각 edge (i,j)는 r(i,j)로 표시된 "slack-retimed variable"로 정해졌다.

전제 1 Constraint 2 is redundant.

증명: 그림 2a를 보고 ($s(v,i_1) < s(v,i_2)$ and $r(i_1)=0$인 경우라고 고려하자), $r(v) = s(v,i_1)$ 대신 $r(v) = s(v,i_2)$ 을 선택하기 위해 제한조건 $r(v) \le r_{min}(v)$를 따르지 않는다라고 가정하면, $r(v, i_1) < 0$ 이 된다. 왜냐하면 $r(v, i_1) = s(v,i_1) - s(v,i_2) < 0$ 이기 때문이다; 이것은 반대가 된다. 왜냐하면 re-budgeting 후에 각 edge e 에 대한 $s(e)$ 는 음수가 되지 않기 때문이다. 따라서 제한조건 3은 $r(i,j) \ge 0, \forall(u,v) \subseteq E$ of G로 간략화 할 수 있다. Slack은 0이 아니기 때문에 또한 $r(i,j) \ge 1$ 이 필요하다

호스트 노드인 h 는 모든 첫 입력이나 출력과 연결된 가상 edge('0'의 값을 갖는 edge)로써 입력 소스이자 출력 싱크가 된다.

연결된 그래프 G 는 또한 다음과 같은 특성을 갖는다: $r(u,v) \ge 0, \forall(u,v) \in E$ of G.

따라서, 결국 다음을 얻을 수 있다

Algorithm Geometry-constrained Slack Re-budgeting (GSB)

Minimize: $\Sigma(|FO(v)|-|FI(v)|+1)r(v)$ (if $|FO(v)| \ge |FI(v)|$)

$-\Sigma(|FI(v)|-|FO(v)|+1)r(v)$ (if $|FI(v)| > |FO(v)|$)

(exception: r(v)has no weigt if $v \subseteq PI$)

Subject to

Constraint 1 (triangle inequality)

r(u,v)+r(v,w) ≥r(w,u);

r(u,v)+r(w,u) ≥r(v,w);

r(u,w)+r(w,u) ≥r(u,v) ;

Constraint 2 (retiming)

if (i,j), $\forall$(i,j) ∈E, then

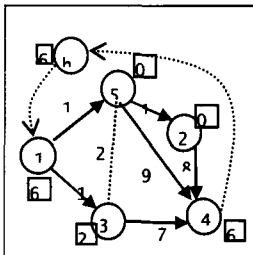
$s(u,v) + r(v) - r(u) \ge 1$;

else $s(u,v) + r(v) - r(u) \ge 0$; /* edges created after triangulation */

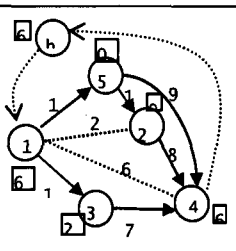
r(i) − r(h) ≥ 0, ∀i∈PI(G); /* h:imaginary host node */

r(h) −r(o) ≥ 0, ∀o∈PO(G);

그림 3은 타이밍 이득에서 양의 이득을 얻는 re-budgeting의 예가 실현되는 것을 나타낸다. 최초 회로 그래프인 그림 1a와 비교하여 GRA가 총 slack(from 21 to 21)이 유지되는 동안 GSB 알고리즘은 6(from 21 to 27)의 slack 이득을 얻는다.



a) Applying GSB (total slack = 27) with triangulation A



b) Applying GSB (total slack= 27) with triangulation B

그림 3. 서로 다른 접근법 비교

4. 실험 결과

알고리즘 GSB의 정확도를 증명하기 위해서 선형 프로그램 솔루션인 lp_solve을 사용하여 표 1에 그 결과를

나타내었다. 모든 예제에서 slack gain을 얻을 수 있었다.

표 1 실험 결과

ex's	# of nodes	# of edges	Slack gain
ex1-1(Fig3a)	5	6	6
ex1-2(Fig3b)	5	6	6
ex1-3	5	6	6
ex1-4	5	6	6
ex3	9	11	7
ex4-1	16	20	3
ex4-2	16	20	3
ex5-1	25	36	31
ex6-1	50	61	16
ex6-2	50	61	16
ex7-1	100	123	44
ex7-2	100	123	44
ex8-1	200	248	90
ex8-2	200	248	90

Slack Gain은 서로 다른 triangulation 그래프(예를 들면 ex4-1 과 ex4-2)에 따라 변하지 않는 것을 알 수 있다. 그리고, 수행시간은 노드가 200개인 경우 2초 정도이다.

5. 결론

이 논문에서는 기하학적인 면과 타이밍 제약조건을 모두 고려한 효율적인 최대허용 지연시간 처리에 대한 알고리즘을 나타내었다. 이 문제를 해결하기 위한 선형 프로그래밍 접근법을 나타내었다. 빠른 계산과 처리 과정의 강화를 위해 배치를 하는 동안 타이밍 문제를 줄이고 더 좋은 예측성을 갖는 솔루션을 찾기 위한 독창적인 그래프 공식을 제안하였다. 확실하건대, 위의 기하학적 특성은 레이아웃 단계(배치와 라우팅)의 복잡성을 줄일 수 있다.

Acknowledgments

본 논문은 과학재단, IT-SoC, IDEC 의 지원을 받았음.

참고 문헌

[1] A. Mathus and C.L.Liu, "Compression-Realization: A New Approach to Timing-Driven Placement for Regular Architectures," IEEE TCAD, Vol. 16, No. 6, June 1997

[2] J.D.Cho and M. Sarrafzadeh, "Four-bend Top-Down Global Routing", *IEEE Trans. on CAD of Integrated Circuits*, Vol. 17, No. 9, pp. 793-802, September 1998

[3] (U.S. *Patent)* US 6480991B1, Jun-Dong Cho, David Kung , "Timing-Driven Multi-Level Partitioning and Placement using Geometry-aware Timing Budgets" Nov. 12, 2002

[4] C. Leiserson and J. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, pp. 5-35, 1991.

[5] S. Ghiasi, E. Bozorgzadeh, P. Huang, and M. Sarrafzadeh, "Unified Theory of Timing Budget Management", in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* Vol. 25, No. 11, pp. 2364-2375, November 2006.