1999年度 大韓電子工學會 秋季綜合學術大會 論文集 第 22 卷 第2號 99/11

면적 최적화를 위한 셀 교체 알고리듬

김 탁 영, 김 영 환
포항공과대학교 전자전기공학과 VLSI & CAD Lab.
경북 포항시 남구 효자동 산 31 번지
E-mail : tykim@postech.edu, youngk@postech.edu

Cell Replacement Algorithm for Area Optimization

Tak-Yung Kim and Young Hwan Kim
Dept. of Electronic and Electrical Engineering
Pohang University of Science and Technology
San 31 Hyojadong, Pohang, Kyungbuk, Rep. of Korea
E-mail : tykim@postech.edu, youngk@postech.edu

Abstract

This paper presents an efficient algorithm that minimizes the area of the combinational system through cell replacement. During the minimization, it maintains the circuit speed same. For the minimization, the proposed algorithm defines the *criticality* of each cell, based on the critical delay and the number of paths passing through the cell. Then, it visits the cells of the system, one by one, from the one with the lowest criticality, and replaces it with the minimum area cell that satisfies the delay constraint. Experimental results, using the *LGsynth91* benchmark circuits synthesized by *misII*, show that the proposed algorithm reduces the circuit area further by 17.54% on the average without sacrificing the circuit speed.

I. 서론

일반적으로 초집적회로설계는 physical layout으로 가기 전 회로 소자의 크기를 조정함으로써 회로의 딜레이, 면적, 전력 소모 등을 최적화하는 과정을 거치게 되는데 이를 sizing problem이라 한다. 그러나, standard cell library를 사용하는 셀 기반의 초집적회로설계에서는 회로 소자마다 사용할 수 있는 라이브러리 셀의 수가 한정되어 있고, 크기 또한 discrete size를 가진다. 이러한 discrete cell library를 기반으로 소자의 셀을 선택하는 작업을 셀 선택(cell selection)[1,2,3,4,5], 셀 교체(cell replacement), 또는 회로 구현 문제(circuit implementation problem)[6]라 하며, 최적의 셀을 선택하는 것은 NP complete 문제로 알려져 있다[2,3,6].

조합논리회로에서 셀 선택을 이용해 면적을 최적화하는 알고리듬들은 크게 heuristic 방법[1,2,3,5,6]과 linear programming 기법을 이용한 방법[4]으로 나눌 수 있다. Lin[1]은 *sensitivity*와 *criticality*의 weighted sum을 통해 면적을 최적화하는 알고리듬을 제시하였고, Li[6]는 series-parallel circuit에 대해 면적을 최적화하는 6가지 heuristic 알고리듬을 제시하였다. Chan[2]은 tree network에 대해서 면적을 최적화하며, tree network이 아닌 network에 대해서는 tree cloning과 backtracking에 기반한 알고리듬을 제시하였다. Chung[3]은 *slack*, *heaviness*, *sensitivity*를 이용하여 maximal series-parallel graph에 대해 바인딩을 반복함으로써 면적을 최적화하는 알고리듬을 제시하였고, Kim[5]은 회로 소자의 라이브러리 셀 조합으로 만들어지는 모든 회로들을 쉽게 표현할 수 있는 *candidate web*을 기반으로 하여 면적을 최적화하는 branch-and-bound 알고리듬을 제시하였다. Chuang[4]은 linear programming을 이용하여 찾은 solution을 사이에 두는 2개의 라이브러리 셀을 선택하여 enumeration과 *local*

*delay difference*를 이용해 면적을 최적화한다.

셀 선택을 이용한 많은 알고리듬들이 소개되었지만, 이들은 fanout에 의한 딜레이 변화를 무시하거나[2,6], 회로 합성 당시의 셀 바인딩 상태를 무시[1,2,3,4,5,6]하고 셀 선택을 수행한다. 이것은 회로 합성으로 인해 optimal solution에 가까운 셀 바인딩 상태를 이용하지 못하고, 처음부터 다시 optimal solution을 찾게 되어 *redundant operation*을 수행하는 단점이 있다. 또한, 셀 라이브러리의 discrete한 특성으로 인하여, linear programming을 이용하는 알고리듬에는 제약 조건을 만족시키기 위한 별도의 heuristic phase가 존재한다[4].

본 논문에서는 셀 기반의 초집적회로설계에서, 회로 합성 당시의 셀 바인딩을 회로 소자들의 초기 셀 선택으로 하고, 회로 소자의 *criticality*가 작은 소자부터 방문하여 각 소자의 라이브러리 셀들 중에서 회로의 딜레이 제약 조건을 만족하는 가장 작은 면적의 셀로 소자의 셀을 교체하는 greedy 방식의 면적 최적화 알고리듬을 제시하도록 한다.

논문의 구성은 다음과 같다. 2장에서는 회로소자모델을 설명하고, 3장에서는 타이밍 해석을 설명한다. 그리고, 4장에서는 면적 최적화 방법을 소개하고, 5장에서는 실험 결과를 분석하며, 6장에서는 결론을 기술한다.

II. 회로소자모델

회로 소자는 n개의 입력과 1개의 출력을 가지는 라이브러리 셀로 표현하며, 셀의 상승/하강 딜레이 D_r/D_f는 각각 식 (2.1), (2.2)와 같이 선형 딜레이 모델로 가정한다[7]. 여기에서 I_r/I_f는 무부하시 셀의 순수 상승/하강 딜레이, R_r/R_f는 상승/하강 딜레이에 대한 캐패시턴스 구동 능력을 의미하고 C_L은 셀이 바인딩된 회로 소자 출력단의 부하 캐패시턴스를 의미한다.

$$D_r = I_r + R_r \times C_L \tag{2.1}$$

$$D_f = I_f + R_f \times C_L \tag{2.2}$$

III. 타이밍 해석

타이밍 해석이란 경로 해석과 함께 회로의 임계 경로와 임계 경로의 딜레이를 구하는 과정이다. 경로 해석 방식에서는 일반적으로 회로소자를 버텍스(vertex, V), 회로소자 사이의 연결선을 방향성 에지(directed edge, E)로 표현하여 회로를 그래프($G=<V,E>$)로 변환한다. 제안하는 알고리듬에서는 $G=<V,E>$의 버텍스에 라이브러리 셀에 대한 정보를 포함시킨 그래프($G=<V,E,C>$)로 회로를 변환한다. [그림 3.1]은 회로를 그래프로 변환한 간단한 예로, 회로 (a)가 (b)와 같은 셀 라이브러리를 가질 경우 (c)의 그래프로 나타낼 수 있다. 주입력 PI, 회로 소자 A와 B, 주출력 PO는 버텍스로 나타내고, 각 버텍스들은 방향성 에지로 연결된다. 그리고, 버텍스 내부에 나타나 있는 실선을 버텍스 A가 셀 C1, 버텍스 B가 셀 C2로 바인딩된 경우를 나타내며, 점선은 그 반대를 나타낸다.

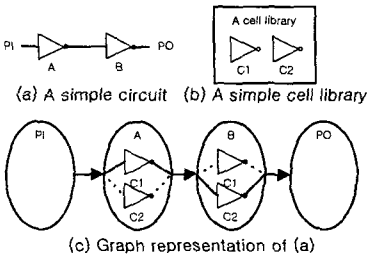


그림 3.1 회로의 그래프 표현

제안하는 알고리듬은 [그림 3.1]과 같이 회로를 그래프로 변환하고, [그림 3.2]의 *MBFS*[8]를 incremental하게 수행하여 타이밍 해석을 효율적으로 수행한다. 여기에서 *level*의 정의는 식 (3.1)과 같다. *LAT*는 소자의 출력단에서의 latest arrival time을 나타내고, $D(fi,v)$는 소자 fi와 연결된 소자 v의 에지를 나타낸다.

$$level(G) = \max_{g \in fanin} \{level(g)\} + 1, \quad where\ level(PI) = 0 \tag{3.1}$$

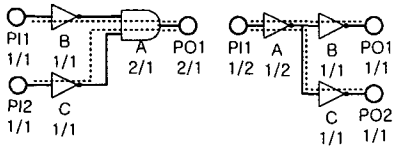
```
procedure MBFS()
    foreach ( vertex v in order of increasing level ) {
        foreach ( fanin of v, fi ) {
            oldValue = v->LAT;
            newValue = fi->LAT + D(fi,v);
            if(newValue > oldValue) v->LAT = newValue;
        }
    }
end MBFS;
```

그림 3.2 *MBFS* 알고리듬의 PSEUDO CODE

IV. 면적최적화

일반적으로 greedy 알고리듬에 기초하여 면적을 최적화하는 경우, 소자의 딜레이 변화가 회로의 딜레이 변화에 영향을 작게 주는 소자들의 셀을 먼저 교체하는 것이 면적 최적화에 유리하다. 즉, [그림 4.1]과 같은 회로가 있을 경우, 소자 A의 셀을 교체하는 것보다 B, C 두 개의 셀을 교체하는 것이 좋다. [그림 4.1]을 보면, 소자 A를 통과하는 경로의 수는 2개이고, B, C

를 통과하는 경로의 수는 각각 1개로, 소자를 통과하는 경로의 수가 많을수록 소자의 딜레이 변화가 전체 회로의 딜레이에 큰 영향을 미치는 것을 알 수 있다. 이에 따라 식 (4.1), (4.2)로 fanin에 의해 생성된 경로의 수 $PathNum_{fanin}$과 fanout에 의해 생성된 경로의 수 $PathNum_{fanout}$을 구하고, 식 (4.3)으로 회로에 존재하는 경로의 수 $PathNum_{circuit}$을 구하여, 식 (4.4)와 같이 *path criticality*를 정의한다. 여기에서 G, g는 각각 회로 소자를 나타내고, g는 G의 fanin 소자이다.



(a) A path generation with fanin (b) A path generation with fanout

$(PathNum_{fanin}, PathNum_{fanout})$

그림 4.1 경로 생성

$$PathNum_{fanin}(G) = \sum_{g \in fanin(G)} PathNum_{fanin}(g), \quad where\ PathNum_{fanin}(PI) = 1 \tag{4.1}$$

$$PathNum_{fanout}(g) = \sum_{G \in fanout(g)} PathNum_{fanout}(G), \quad where\ PathNum_{fanout}(PO) = 1 \tag{4.2}$$

$$PathNum_{circuit} = \sum_{PO \in primary\ outputs} PathNum_{fanin}(PO) \tag{4.3}$$

$$PathCriticality(G) = \frac{Number\ of\ paths\ through\ G}{Number\ of\ paths\ in\ the\ circuit} = \frac{PathNum_{fanin}(G) \times PathNum_{fanout}(G)}{PathNum_{circuit}} \tag{4.4}$$

일반적으로 회로의 임계 경로 딜레이(CPD)와 소자를 통과하는 임계 경로 딜레이의 차이인 슬랙이 큰 소자들은 회로의 임계 경로 딜레이에 주는 영향이 작다. 이러한 자유도로 인해, 슬랙이 큰 소자들의 셀을 먼저 교체하는 것이 회로의 다른 소자들의 셀 교체에 영향을 작게 미친다. 이에 따라 슬랙에 따른 소자의 자유도를 식 (4.5)와 같이 회로 소자 G의 *delay criticality*로 정의한다. 식 (4.5)에서 회로의 임계 경로 딜레이는 $MBFS$를 이용하여 구하고, 회로 소자의 임계 경로 딜레이는 LAT 와 MDS(max-delay-to-sink)[9]을 이용하여 쉽게 구할 수 있다.

$$DelayCriticality(G) = \frac{CPD\ through\ G}{CPD\ of\ the\ circuit} \tag{4.5}$$

제안하는 알고리듬은 *path criticality*와 *delay criticality*를 함께 고려하기 위해 식 (4.6)과 같이 두 가지 criticality의 weighted sum으로 회로 소자 G의 *criticality*를 정의한다

$$Criticality(G) = k \times PathCriticality + (1-k) \times DelayCriticality \tag{4.6}$$

제안하는 알고리듬은 [그림 4.2]와 같이 소자의 *criticality*를 계산한 후, heap sort를 이용하여 회로 소자들을 *criticality*가 작은 순서대로 정렬한다. 그리고나서, *criticality*가 작은 소자부터 방문하여 딜레이 제약조건을 만족하는 가장 작은 면적의 라이브러리 셀로 greedy하게 교체한다. [그림 4.2]에서 딜레이 제약 조건 검사는 LAT를 update 해 줄 필요가 있는 부분만 $MBFS$를 incremental하게 수행한다.

```
procedure Optimize_CircuitArea
    Calculate criticality with (4.1), (4.2), (4.3), (4.4), (4.5), and (4.6);
    Sort combinational vertices with criticality;
    foreach ( vertex v in order of sorted list ) Optimize_VertexArea(v);
end Optimize_CircuitArea;

procedure Optimize_VertexArea ( vertex, v )
    foreach ( candidate cell c of v in order of increasing area ) {
        if ( c is the current cell of v ) return;
        foreach ( fanin of v, fi ) update fi's ouput load;
        if ( the delay constraint is satisfied with incremental MBFS ) {
            update the information of vertices; // cf. LAT
            return;
        }
        restore the information of vertices; // cf. LAT
    }
end Optimize_VertexArea;
```

그림 4.2 면적 최적화 알고리듬

회로에 소자가 V개, 소자마다 라이브러리 셀이 C개 있을 경우, *criticality*를 계산하는데 $O(V)$의 시간이 필요하고, 소자들을 정렬하는데 $O(V\ln V)$의 시간이 필요하다. 타이밍 해석을 한 번 수행하는데 $O(V)$의 시간이 필요하고, 각 소자의 면적을 최적화하는 데에는 최대 $(C-1)$번의 타이밍 해석을 수행하여야 하므로, V개 소자의 면적을 최적화하는 데에는 $O(CV^2)$만큼의 시간이 필요하다. 따라서, 제안하는 알고리듬은 $O(V\ln V + CV^2)$의 복잡도를 가진다. 여기에서 $O(CV^2) > O(V\ln V)$이고, 보통 수 개로 한정되어 있는 라이브러리 셀의 수 C를 상수로 취급하면 제안하는 알고리듬은 $O(V^2)$의 복잡도를 가지고 면적을 최적화한다.

V. 실험 결과

제안하는 알고리듬을 C++ 언어로 작성하여 *ACE*를 구현하였으며, SUN SPARCstation20에서 LGSynth91 benchmark circuits[10]에 0.8μm standard cell library [11]와 *misII*[12]로 테크놀로지 매핑을 수행하여 초기 회로를 얻었다(map -m 0.75 -F)[1,3]. [표 5.1]에 *misII*로 충분히 최적화 한 결과와 이것을 초기 회로로 이용한 *ACE*의 결과를 비교하였으며, *ACE*는 *criticality*의 weighting factor, *k*,를 실험적으로 구한 값인 0.9를 사용하였다. *misII*의 결과에 비해 평균 7.1초의 실행 시간으로 평균 17.54%의 면적을 추가로 줄였다.

표 5.1 면적 최적화 성능 비교

Name	*misII*			*ACE*	
	Gates	*CPD*	Area	Imp.(%)	sec.
alu2	524	19.28	229539.84	13.60	0.6
alu4	1106	22.02	479477.76	15.22	1.9
apex6	997	8.23	503685.12	30.86	1.2
c1355	614	13.45	233472.00	1.95	0.6
c1908	112	17.15	325877.76	12.29	0.9
c3540	1564	23.21	710369.28	14.76	3.6
c432	222	16.07	104693.76	12.79	0.3
c499	623	11.99	265666.56	12.12	0.8
c6288	3205	59.60	1186652.16	0.56	2.9
c880	423	11.48	185548.80	13.31	0.3
cordic	143	5.67	71147.52	31.09	0.1
count	145	13.88	69304.32	14.18	0.1
i10	2997	23.72	1378467.84	14.00	9.8
i6	548	2.60	356229.12	37.15	1.7
i7	717	2.52	446668.80	33.34	2.8
i8	2160	9.87	1241579.52	17.28	12.2
i9	599	8.59	328581.12	5.20	1.6
t481	4031	8.61	2636290.40	36.15	87.2
Ave.	1151	15.44	597402.87	17.54	7.1

VI. 결론

본 논문은 셀 기반의 초집적회로설계에서 셀 교체를 통한 면적 최적화 알고리듬으로, 회로합성 당시의 셀 바인딩을 회로 소자들의 초기 셀 선택으로 하고, 회로 소자의 *criticality*가 작은 소자부터 방문하면서 딜레이 제약 조건을 만족하는 면적이 가장 작은 라이브러리 셀로 소자의 셀을 교체하는 greedy 방식의 알고리듬을 제안하였다. SUN SPARCstation20에서 LGSynth91 benchmark circuits을 *misII*로 충분히 최적화 한 회로를 실험을 통해 평균 7.1초의 실행 시간을 가지고 평균 17.54%의 면적을 추가로 줄였다. 본 논문에서 제안한 면적 최적화 알고리듬은 빠른 실행 시간과 충분한 면적 최적화 결과로 인해 초집적회로설계에 효율적으로 사용할 수 있으리라 본다.

참고문헌

[1] Shen Lin, M. Marek-Sadowska, and Ernest S. Kuh, "Delay and Area Optimization in Standard-Cell Design," *Proc. 27th ACM/IEEE Design Automation Conference*, pp. 349-352, 1990.

[2] Pak K. Chan, "Algorithms for Library-Specific Sizing of Combinational Logic," *Proc. 27th ACM/IEEE Design Automation Conference*, pp. 353-356, 1990.

[3] Moon Jung Chung and Sangchul Kim, "A Path-Oriented Algorithm for the Cell Selection Problem," *IEEE Trans. on CAD.*, Vol. 14, No. 3, pp. 296-307, March 1995.

[4] Weitong Chuang, Sachin S. Sapatnekar, and Ibrahim N. Hajj, "Timing and Area Optimization for Standard-Cell VLSI Design," *IEEE Trans. on CAD.*, Vol 14, No. 3, pp. 308-320, March 1995.

[5] Tae Hoon Kim and Young Hwan Kim, "Area Optimization Algorithm for Cell Selection Problems," *IEE Electronics Letters*, Accepted.

[6] WingNing Li, Andrew Lim, Prathima Agrawal, and Sartaj Sahni, "On the Circuit Implementation Problem," *Proc. 29th ACM/IEEE Design Automation Conference*, pp. 478-483, 1992.

[7] R. A. Saleh, *Iterated Timing Analysis and SPLICE1*, Ucb/erl m84/2, University of California, Berkeley, Jan. 1984.

[8] N. P. Jouppi, *Timing Verification and Performance Improvement of MOS VLSI Design*, Ph. D. thesis, Stanford University, CA 94305-2192, Oct. 1984.

[9] Steve H. C. Yen, David H. C. Du, and S. Ghanta, "Efficient Algorithms for Extracting the K Most Critical Paths in Timing Analysis," *Proc. 26th ACM/IEEE Design Automation Conference*, pp. 649-653, 1989.

[10] S. Y. Yang, "Logic Synthesis and Optimization Benchmarks User Guide," Ver. 3.0, Jan. 1991.

[11] LG Semicon Corp., *0.8μm Standard Cell GVSC450 Library Users Guide.*

[12] Robert K. Brayton, Richard Rudell, Algeto Sangiovanni-Vincentelli, and Albert R. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Trans. on CAD.*, Vol. CAD-6, No. 6, pp. 1062-1081, Nov. 1987.