지연 제약 하에서 면적의 최적화를 위한 트랜지스터 사이징과 버퍼 삽입 알고리즘

(Transistor Sizing and Buffer Insertion Algorithms for Optimum Area under Delay Constraint)

이 성 건 † 김 주 호 ††
(Sungkun Lee) (Juho Kim)

요 약 저 전력회로의 설계를 위해서, 전체 회로의 면적을 줄임으로써 용량성 부하(capacitance)값을 줄이는 방법으로 적절한 트랜지스터를 선택하여 사이징하는 방법을 이용할 수 있는데, 이 때 트랜지스터 사이징을 수행하면서 적당한 위치에 버퍼를 삽입하여주면 더 좋은 결과를 가져올 수 있다. 본 논문은 TILOS 알고리즘을 이용하여 트랜지스터 사이징(sizing)을 수행하는 동시에 버퍼의 삽입을 수행하는 알고리즘 두 가지를 소개하고 이 두 방법을 비교한다. 그 첫 번째 방법은 Template Window를 이용하여 직접 시뮬레이션하는 방법이고 다른 하나는 보외법(Extrapolation)을 이용하는 방법이다. 이와 같이 버퍼를 삽입하면서 트랜지스터 사이징을 수행한 결과, 버퍼를 삽입하지 않을 때 보다 10-20%의 면적감소를 얻었을 수 있었으며 보외법을 이용한 방법 보다 Template Window를 이용했을 때 더 좋은 결과를 얻을 수 있었다.

Abstract For designing circuits for low power systems, the capacitance is an important factor for the power dissipation. Since the capacitance of a gate is proportional to the area of the gate, we can reduce the total power consumption of a circuit by reducing the total area of gates, where total area is a simple sum of all gate areas in the circuit. To reduce the total area, transistor resizing can be used. While resizing transistors, inserting buffer in the proper position can help reduce the total area. In this paper we propose two methods for concurrent transistor sizing and buffer insertion. One method uses template window simulation and the other uses extrapolation. Experimental results show that concurrent transistor sizing with buffer insertion achieved 10-20% more reduction of the total area than when it was done without buffer insertion and template window simulation is more efficient than extrapolation.

1. 서 론

휴대용 전화기나 카세트와 같은 휴대용 전자제품들의 수요가 점점 증가하며, 칩의 크기가 점점 작아지는데 비례하여 저 전력 회로설계가 중요시되고 있다. 즉, 휴대용 전자제품의 경우 가장 중요한 것은 한번의 충전으로 얼마만큼 오래 사용할 수 있는가가 중요한 문제이며, 또한 칩의 크기는 작아지는데 반해 그 속도는 빨라지고 있기 때문에 칩에서 발생하는 열로 인해 칩의 수명이 단축되는 것을 방지하기 위해서, 칩을 생산함에 있어서 보다 많은 비용이 들게 된다. 이로 인해 최근 몇 년 전부터 저 전력에 대한 문제가 대두되었다. 식 (1)에서 보여지듯이 회로의 전력 소모 중 대부분을 차지하는 동적 전력 소모(Dynamic Power Dissipation)는 용량성 부하 값에 비례하므로[1] 회로의 면적을 줄임으로써 회로의 용량성 부하를 줄여서 전력 소모를 줄일 수 있다. 이와 같이 본 논문에서는 회로를 설계함에 있어서 주어진 지연 제약 하에서 최소의 면적을 갖도록 하는데 초점을 맞추고 있다.

$$P = \frac{1}{2} V_{dd}^2 \sum_i C_i f N_i \qquad (1)$$

여기서 회로의 면적은 회로 내의 전체 트랜지스터 면적의 합으로 계산되며, 제안된 방법에서는 래치

† 비 회 원 : SiliconCraft R&D Engineer
sklee@siliconcraft.com
†† 종신회원 : 서강대학교 컴퓨터학과 교수
jhkim@ccs.sogang.ac.kr
논문접수 : 1999년 7월 26일
심사완료 : 2000년 5월 17일

(latch) 사이의 하나의 조합회로에 대해서만 고려하는데, 각각의 조합회로 블록들의 지연이 클럭주기와 맞는 범위 내에서 회로의 면적이 최소가 되도록 한다. 이와 같은 방법으로 큰 회로를 사이징하는 문제도 이처럼 각각의 조합회로 블록으로 나누어서 생각할 수 있다. CMOS 조합 회로에 있어서 회로내의 모든 트랜지스터들의 크기들을 가장 작게 하였을 때, 그 회로는 가장 적은 면적을 가지게 되지만, 이 경우 시간제약을 만족할 수 없게 되므로 특정 트랜지스터를 사이징해야 할 필요가 있다. 그림 1은 트랜지스터의 면적과 지연과의 관계를 나타내었는데, 보여지는 바와 같이 트랜지스터의 크기가 클수록 속도는 더 빨라지는 것을 알 수 있다[2][3]. 또한 지연 제약이 심하지 않을 경우 적은 트랜지스터 사이징을 통해 만족할만한 결과를 얻을 수 있으나 지연 제약이 심해질 수록 사이징을 많이 해야 한다[2][3]. 트랜지스터 사이징을 하는 방법으로 잘 알려진 알고리즘으로 TILOS 알고리즘이 제안되었다[2]. TILOS 알고리즘은 회로내의 모든 트랜지스터들을 최소 크기로 만들어 놓고, 각 트랜지스터마다 센서티비티(sensitivity)를 계산하여, 적절한 트랜지스터를 선택하여 사이징하는 방법으로 모든 경로가 지연 제약에 맞을 때까지 반복한다. 이 때 게이트에 걸린 용량성 부하가 클수록 그 게이트의 크기를 더 크게 사이징 해 주어야 한다[4][5][6]. 이러한 TILOS 알고리즘을 이용하여 트랜지스터를 사이징하면서 동시에 버퍼를 삽입하여 회로의 면적을 더 줄일 수 있다[7][9]. 이와 같이 본 논문에서는 적당한 위치에 버퍼를 자동적으로 삽입해 주는 두 가지 방법을 소개하고 비교하는데 하나는 보외법을 이용하여 버퍼를 삽입하는 방법이고[7][8] 다른 하나는 Template Window 시뮬레이션을 이용한 방법이다[9]. 이 두 가지 방법은 임계경로(critical path)가 길고 팬아웃이 많은 경우에 좋은 결과를 나타낸다. 이는 어떤 게이트의 출력단의 용량성 부하가 큰 경우에 버퍼를 삽입하면, 그 부근의 지연을 줄일 수 있고, 그 앞의 게이트가 보다 작게 사이징됨으로써 같은 시간에 동작하면서 회로의 면적을 줄일 수 있기 때문이다[7][9]. 이는 2장에 자세히 설명하도록 하겠다. 여기서 말하는 회로의 면적은 그 회로에 있는 트랜지스터들의 총 면적으로 계산되는데, 면적을 계산함에 있어서, 길이(length)는 고정되어 있고 너비(width)만 사이징하며 이 때 지연을 계산하기 위해 RC-지연 모델[10][11]을 사용하였다.

본 논문은 다음과 같이 구성되어 있다. II장에서는 사이징을 위한 지연시간 분석과 버퍼 삽입에 대한 기본적인 이론적 배경에 대해 설명하고, III장에서는 버퍼 삽입을 위한 두 가지 알고리즘을 소개한다. IV장에서는 트랜지스터 사이징을 수행하면서 버퍼를 삽입했을 때의 실험결과를 비교하고, V장에서는 버퍼삽입의 효과에 대한 요약과 향후 과제를 결론으로 끝을 맺는다.

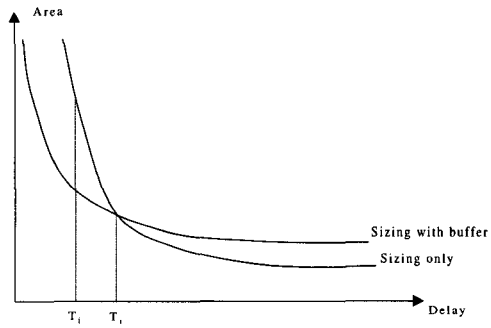


그림 1 면적-지연 곡선 ; 버퍼 삽입과 사이징

2. 버퍼 삽입의 이론적 배경

버퍼 삽입을 이용한 트랜지스터 사이징은 어떤 게이트의 팬아웃에 걸린 용량성 부하가 매우 클 경우 그 게이트가 너무 많이 커져서 면적이 더 증가되거나 사이징할 수 있는 범위를 벗어나는 경우를 방지하기 위해 사용된다.

그림 1에서 보는 바와 같이 TILOS 알고리즘을 이용해서 버퍼를 삽입하지 않고 사이징을 수행했을 때, 버퍼를 삽입하면서 사이징을 수행할 때의 면적과 같아지는 지연 제약 T_i보다 지연 제약이 작은 경우에는 트랜지스터의 면적을 조금만 크게 해 주어도 지연이 많이 향상되지만, 시간제약이 점점 심해질수록 지연을 줄이기 위해서 트랜지스터의 면적을 더 많은 단위로 크게 해주어야 한다[2][3]. 이와 같이 사이징을 하면서 버퍼를 삽입해 주면 똑같은 지연 제한에 맞추면서 회로의 면적을 더 줄일 수 있는데, T_i 까지는 버퍼를 넣지 않고 사이징만 해 주는 것이 면적을 더 줄일 수 있었으며, 시간제약이 더 심하게 주어질 때(T_j, $T_j < T_i$ 일 때), 버퍼를 넣어주면, 더 좋은 결과를 얻을 수 있었다[7][9].

2.1 기본개념

트랜지스터는 기본적으로 너비(width)와 길이(length)를 가지고 있다. 여기서 실제로 사이징되는

것은 트랜지스터의 너비이다. 그리고 각각의 트랜지스터의 면적은 "너비 × 길이"로 계산되어 지며 전체 회로의 면적은 모든 트랜지스터의 면적의 합으로 계산된다. 다음은 본 논문에서 자주 사용되는 용어들에 대한 정의들이다[7][9].

정의 2 어떤 경로 P 가 위반경로(violating path)라는 것은 회로의 출력이 래치에 연결되어 있을 때 시간 제약을 맞추지 못한 것을 말한다. 이 때 상대적으로 많이 초과하는 경로를 크게 위반되는 경로(highly violating Path)라고 하고 조금 초과하는 것을 작게 위반되는 경로(mildly violating path)라고 한다.

정의 2.2 어떤 경로가 임계경로(Critical Path)라는 것은 가장 많이 위반된 경로를 말한다.

정의 2.3 트랜지스터의 센서티비티(sensitivity)는 트랜지스터의 면적을 증가시켰을 때 얻어지는 지연의 향상을 말한다. ($\frac{\partial T}{\partial A}$)

2.2 TILOS 알고리즘

게이트 사이징은 Technology Mapping을 적용한 후 지연, 전력, 면적에 대한 최적화를 위하여 사용되는 기법이다. 과거의 사이징 기법은 주로 시간 제약이 만족되지 않는 회로에 대하여 면적 증가를 최소로 하면서 주어진 시간 제약을 만족시키는 것이 목적이었다[12][13]. 최근에는 저 전력에 대한 관심이 높아지면서 시간 제약을 만족시키면서 전력 소모를 최소화하는 것이 주목적으로 대두되고 있다[14][15][16][17]. 게이트 사이징 알고리즘은 크게 두 가지 접근 방식으로 나누어진다. 첫째는 TILOS 알고리즘을 바탕으로 하여 모든 게이트를 가능한 최소 크기로 놓고 시간 분석을 통해 시간 제약을 만족할 때까지 선택적으로 게이트의 크기를 증가시키는 방법이다. 둘째는 먼저 시간 분석을 통하여 각 게이트의 slack을 구하고, 양의 slack을 갖는 게이트 중에서 선택적으로 게이트의 크기를 감소시키는 기법이다[14][15].

본 논문에서 사용한 방법은 TILOS 알고리즘으로, 이는 회로를 지연 제약에 맞춰서 면적을 최소화시킨다. TILOS 알고리즘은 먼저 회로의 트랜지스터들의 크기를 최소 크기로 초기화한다. 그리고 각각의 트랜지스터마다 센서티비티를 계산하고 센서티비티가 가장 좋은 트랜지스터를 선택하여 그 트랜지스터를 적당한 크기만큼 사이징 해 준다. 이러한 과정들을 회로 내의 모든 경로가 지연제한에 맞출 수 있을 때까지 반복한다.

이 알고리즘을 정리해 보면 다음 그림 2와 같다.

```
Downsize transistors to minimum size or cost
  while( slack(WorstConstraint) < 0 )
    for each transistor on critical path
    find sensitivity of transistor
    increase size of transistor with best sensitivity
    update timing and critical path information
```

그림 2 *TILOS* 알고리즘

그림 2의 best sensitivity란 센서티비티의 절대값이 가장 큰 것을 의미한다.

2.3 형식 A 버퍼 삽입

만일 어떤 게이트의 팬아웃들이 모두 크게 위반되는 경로인 경우 버퍼를 삽입하면 이 경로의 지연이 줄어들게 된다. 즉, 적당한 크기의 버퍼를 잘 선택하면 게이트 G의 팬아웃에 걸리는 용량성 부하 값이 줄어들게 되어 지연을 줄일 수 있다. 이 때, 똑같은 지연으로 맞춰서 사이징을 수행했을 경우 전체 면적이 버퍼를 삽입해 주었을 때보다 버퍼를 삽입하지 않은 경우가 더 크다면 버퍼를 삽입한다[7][9].

이를 그림 3에 나타내었다.

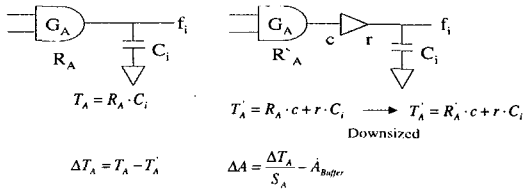


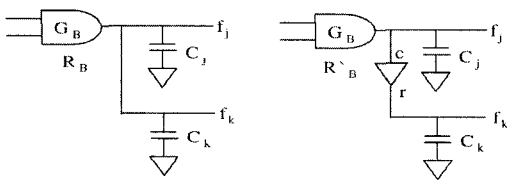
그림 3 형식 A 버퍼 삽입

그림 3에서 S_A는 센서티비티를 나타내고, A_{Buffer}는 삽입된 버퍼의 면적을 나타낸다.

2.4 형식 B 버퍼 삽입

만일 어떤 게이트의 팬아웃들 중에 몇 개는 크게 위반되는 경로이고 몇 개는 작게 위반되는 경로와 비위반경로들로 구성되어 있는 경우 버퍼를 삽입함으로써 비위반경로들의 용량성 부하를 G의 팬아웃으로부터 분리시켜 내어 위반되는 경로의 지연을 줄일 수 있다. 이 때, 형식 A 버퍼 삽입과 마찬가지로 똑같은 지연제약 하에서 버퍼를 삽입해 주었을 경우 면적이 줄어들면 버퍼를 삽입해 준다[7][9].

이를 그림 4에 나타내었다.



$$\Delta T_B^j = R_B \cdot (C_j + C_k) - R_B' \cdot (c + C_j)$$

$$\Delta T_B^k = R_B \cdot (C_j + C_k) - R_B' \cdot (c + C_j) - r \cdot C_k$$

그림 4 형식 B 버퍼 삽입

2.5 버퍼 삽입 효과

간단한 예를 들어 버퍼 삽입의 효과를 살펴보기로 하겠다[7][9].

그림 5와 같이 게이트들 $G_a, G_b, G_c, \ldots, G_f$ 를 구동하는 게이트 G가 있다고 하자. 만일 모든 팬아웃들이 크게 위반되는 경로인 경우에는 G 바로 뒤에 형식 A 버퍼를 삽입하면 된다. 이렇게 버퍼를 삽입하면, G의 팬아웃에 걸리는 용량성 부하값이 변하게 되어, 지연은 T_G^{old} 에서 T_G^{new} 로 변하게 된다. 이 때 버퍼의 지연을 T_{buf} 라고 한다면 임계경로의 출력에서의 지연 향상은 $T_G^{new} + T_{buf} - T_G^{old}$ 가 된다. 이 값이 음수가 되는 경우, 버퍼의 삽입이 지연의 향상에 도움을 준다.

다음으로 형식 B 버퍼 삽입에 대해서 살펴보도록 하자.

G_a, G_b 가 크게 위반되는 경로라 가정하고 G_e, G_f는 비위반경로, G_c, G_d는 위반경로 이지만 크게 위반되는 경로는 아니라고 가정하자. 그러면 다음과 같이 각각의 위치에 버퍼가 삽입될지 여부에 대해 생각해 볼 수 있다.

1) G 와 G_a, G_b 사이에는 버퍼가 삽입될 수 없다.

2) G 와 G_e, G_f 사이에 버퍼가 삽입되어야 할 것이다. 그 이유는 버퍼가 삽입된다 할지라도 이 경로는 위반경로가 될 가능성이 가장 적으며, 만일 최악의 경우에 위반경로가 되더라도, 작게 위반되는 경로일 것이기 때문이다. 따라서 버퍼를 삽입하여 G 의 팬아웃으로부터 분리해 내어 용량성 부하를 줄일 수 있다.

3) G 와 G_c, G_d 사이에 버퍼를 삽입할 수 있을지 없을지를 알아내는 것이 중요한 문제이다. 이 경우 만일 버퍼를 삽입하면 이 경로가 크게 위반되는 경로가 될 수도 있으며 버퍼를 삽입하지 않는 경우에는 G 의 팬아웃 용량성 부하를 충분히 줄이지 못하게 될 수 있다. 따라서 이 경우 버퍼를 삽입하지 않든지, 둘 중 하나만 선택해서 버퍼를 삽입하든지, 아니면 둘 모두 버퍼를 삽입하든지 선택하는 알고리즘이 요구되어지며 여기에 대한 휴리스틱(heuristic)을 3장에 소개하였다.

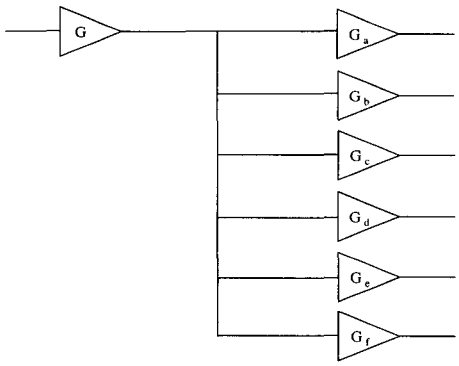


그림 5 버퍼 삽입 효과

3. 버퍼 삽입 알고리즘

버퍼 삽입의 경우 버퍼가 삽입된 후 실제로 같은 지연에 맞추어 게이트를 사이징했을 때 면적이 줄어드는지 확인해 보려면 버퍼를 삽입하고 처음부터 다시 TILOS 알고리즘을 이용하여 사이징을 수행해야 한다. 이 경우 버퍼가 삽입되어질 수 있는 위치가 많으면 수행 시간이 많이 걸린다. 이런 경우 보외법을 이용하거나 Template Window 시뮬레이션을 통해서 버퍼가 삽입되었을 때 실제 면적 감소를 얻을 수 있는지를 빠른 속도로 확인한다.

3.1 보외법을 이용한 버퍼 삽입

이 방법은 먼저 버퍼가 삽입될 만한 위치들(candidate)을 생성하고, 이 위치에 버퍼가 실제로 들어갔을 때 면적 감소를 얻을 수 있는지를 보외법을 이용하여 확인하여 실제로 버퍼 삽입으로 면적 감소를 얻을 수 있는 경우에만 실제로 버퍼를 삽입하는 방법이다[7]. 이러한 보외법을 이용한 방법은 버퍼가 들어갈 만한 위치에 먼저 최소 크기의 버퍼를 삽입한 후 전체 회로의 면적과 지연의 향상을 측정한 후 버퍼를 삽입했을 때와 같은 지연조건을 만족하도록 사이징만 수행했을 때의 면적을 보외법을 이용하여 예측하여 버퍼를 삽입했을 때 면적이 줄어들 경우 실제로 버퍼를 삽입한다.

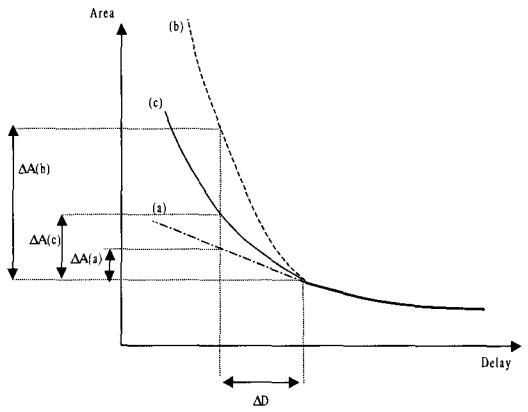


그림 6 면적-지연곡선(2)

먼저 보외법을 이용한 형식 B 버퍼 삽입에 대해 소개하면, 2장에서 설명했던 바와 같이 형식 B 버퍼의 삽입의 목적은 비위반경로를 크게 위반되는 경로로부터 분리해내어 전체 회로의 지연을 줄이는 데 있다. 최소 크기의 형식 B 버퍼 삽입은 크게 위반되는 경로의 지연을 감소시키며 이것은 면적의 증가를 초래한다. 즉, ΔT만큼의 지연을 감소시키기 위해서 ΔA 만큼의 면적 증가가 필요하다. 이제 같은 지연으로 맞추기 위해 사이징을 했을 때 늘어나는 면적 ΔA_T를 계산해야 하며 만일 $\Delta A < \Delta A_T$ 인 경우에 형식 B 버퍼 삽입을 수행한다. 사이징의 효과로 나타나는 면적 증가와 형식 B 버퍼 삽입으로 인한 면적 증가를 비교하기 위해서는, 같은 양만큼의 지연 감소(ΔT)를 위해서 사이징과 형식 B 버퍼 삽입 각각에 대한 면적의 증가가 얼마나 될 것인가를 고려해야 한다. 그림 6의 곡선 (a)는 ΔA_T 를 $-\frac{\Delta T}{S_T}$로써 계산한 경우를 나타낸다. 여기서 S_T는 트랜지스터의 센서티비티를 나타낸다. 그러나 대응하는 면적에 대한 변화 $\Delta A_{(a)}$는 ΔA_T의 단지 하계일 뿐이지 실제 범위가 아니다. 이는 ΔT가 상대적으로 큰 데 반해, 면적이 적게 늘어나기 때문이다. ΔA_T 의 계산을 위해 선형적인 측정을 할 수 있는데, 그 식은 다음과 같다.

$$S_T = K_1 - \frac{K_2}{x^2} \qquad (2)$$

여기서 K_1, K_2 는 x에 독립적이다[2]. 위의 식은 임계경로의 지연에 대해서는 정확하지만, 전체 회로의 지연에 대해서는 그렇지 못하다. 그 이유는 트랜지스터의 크기를 변화시킬 때마다 임계경로가 변하기 때문이다. 두 번째 방법은 위의 그림 6의 곡선(b)에서 보여지는 바와 같이 식(2)을 이용하여 ΔA_T를 계산하는 것이다. 그러나 이 방법은 시간 지연을 ΔT만큼 줄이기 위해 같은 트랜지스터를 계속 사이징하는 경우에만 정확하다. 이것은 옳지 않으며 면적 증가에 대한 상계를 나타낸다. 대부분의 경우 실제 면적-지연 곡선은 위의 그림 6에서의 곡선(c)와 같이 나타내어 질 것이며, 이 곡선의 모양을 결정하고 ΔA_T를 구하는 것이 중요하다.

TILOS 알고리즘을 사용하여 트랜지스터 사이징을 수행할 때 현재 수행단계에서의 전체 회로의 지연을 T 라고 하자. 이 때 각각의 트랜지스터의 크기를 크게 했을 때 지연의 감소를 δT라고 하고, 이 때 생기는 임계경로의 면적 증가를 δA라고 했을 때 전체 회로의 지연은 δT만큼 줄지 못할 수 있다. 그 이유는 만일 어떤 트랜지스터의 크기를 변화시켰을 때 다른 경로가 위반경로가 될 수 있기 때문이다. 따라서 모든 경로의 지연을 $T-\delta T$에 맞추도록 제약을 주고, 여기 맞추어 면적의 증가를 고려해야 할 것이다. 다시 말하면, 모든 경로들이 적절히 사이징되어져야 하며, 단지 임계경로에서의 면적 증가뿐만 아니라 모든 경로들에 대해서 ΔA를 구해야 한다. 이런 문제를 고려하기 위해 모든 PO(primary output)에 대해 면적증가를 고려해야 한다. 다시 말해서, 모든 출력에 대해 최대 지연이 $T-\delta T$보다 같거나 작게 하도록 하는 면적증가를 구해야 한다.

센서티비티 $\frac{\partial T}{\partial A}$ 는 게이트의 크기에 따라 비선형적으로 변하고, ΔT 만큼 지연을 줄이기 위해 사이징하는 경우 하나의 게이트만을 사이징하는 것이 아니라 여러 개의 게이트들을 사이징 해야 하기 때문에 ΔA_T를 구하는데 있어서 $\Delta T / \frac{\partial T}{\partial A}$ 의 방법을 쓰면 정확하지 않다. 따라서 주어진 버퍼 삽입을 하기 위한 지점에 대해 ΔA_T 를 구하기 위하여, 먼저 최소 크기의 버퍼를 그 지점에 삽입한 후, 지연의 감소를 계산한다. 그리고 회로의 지연을 똑같이 맞추어 주었을 때의 면적증가(Δa_i)를 각각의 primary output i에 대해, 보외법을 사용하여 계산한다. 그리고 식(3)을 이용하여 사이징을 했을 때 늘어나는 총 면적을 구할 수 있다.

$$\Delta A_T = \sum_{i \in PO} \Delta a_i \qquad (3)$$

본 논문에서는 Lagrangian extrapolation[8]을 사용하여 ΔT에 대한 ΔA_T를 구하였고, 4차 다항식을 이용한 근사치를 이용하였다.

만일 지연을 줄이기 위해 사이징만을 수행했을 때의 면적증가(ΔA_T)가 형식 B 버퍼를 삽입했을 때의 면적증가(ΔA_B) 보다 크다면 그 지점에 버퍼를 삽입한다. 만일 그렇지 않다면 형식 A 버퍼 삽입을 고려하거나 사이징만을 수행한다. 이 때 위반정도를 계산해 내어 버퍼가 삽입될만한 위치를 찾는 알고리즘이 필요한데, 제안된 알고리즘은 게이트의 팬아웃을 위반경로 집합과 비위반경로 집합을 분리해 내기 위해 식 (4)을 사용한다.

$$X_i = \min_{j \in fanouts(i)}[X_j + slack_j] + |\sigma_i| \quad (4)$$

$$\text{where } \sigma_i = \min(0, sensitivity \cdot \Delta x_i)$$

여기서 $slack_j$는 게이트 i에서의 slack을 의미하며, Δx_i는 게이트 i 가 사이징 되었을 때의 면적의 증가를 나타낸다. 또한 primary output의 X값은 그 지점에서의 최대 지연과 실제 그 지점에서의 지연의 차이로 나타낸다. 만일 게이트 i의 팬아웃 j에서의 X_i의 값이 크다면, i로부터 j 를 통해 PO까지 가는 경로는 비위반경로라는 것을 의미한다. 버퍼가 들어갈 만한 위치를 찾기 위해 위반경로 상의 가장 큰 용량성 부하를 팬아웃으로 갖는 게이트 i를 찾고 그 게이트 i의 모든 팬아웃들의 가장 큰 X_j들을 구한다. 이 때, X_{max}를 X_j들 중에서 가장 큰 값으로 놓으면, 모든 $X_j \geq c_1 \cdot X_{max}$인 팬아웃 j들을 비위반경로 집합으로 놓는다. 여기서 c_1은 1보다 작은 값으로 tuned number이다. 버퍼가 삽입 될 때, 게이트 i를 통한 위반경로의 지연이 ΔT_{dec}만큼 감소되었다고 하면, 비위반경로의 팬아웃 j를 통한 지연은 $\Delta T_{inc} - \Delta T_{dec}$ 만큼 감소된다. 여기서 ΔT_{inc}는 버퍼를 삽입함으로써 증가되는 지연이다. 따라서 j로부터 PO까지의 지연은 $X_j - (\Delta T_{inc} - \Delta T_{dec})$만큼 증가한다. 이 때, 모든 팬아웃 j 에 대해, 만일 $X_j - (\Delta T_{inc} - \Delta T_{dec}) < \beta$ 이면 게이트를 비위반경로 집합으로부터 위반경로 집합으로 이동시킨다. 여기서 β는 $c_2 \cdot d_{min\,size}$ 로 구하며, $d_{min\,size}$는 최소 크기의 인버터의 지연을 나타낸다. 여기서 알고리즘에서 c_1, c_2 는 실험적으로 결정된 값이며, $c_1 = 0.8, c_2 = -0.5$ 일 때 좋은 결과를 나타낸다[7].

형식 A 버퍼 삽입의 경우, 삽입되는 버퍼 크기를 최소 크기로 하면, 항상 그 지연은 늘어난다[7]. 그러나 적절한 크기의 버퍼를 삽입하면, 지연을 줄일 수 있다. 형식 A 버퍼 삽입의 알고리즘은 다음과 같다.

1) 임계경로의 절대값이 가장 큰 음수 값의 센서티비티를 찾고 그 값을 $\frac{\partial T}{\partial A}|_{best}$라고 한다.

2) 위반경로의 각각의 게이트의 출력에 한번에 하나씩 적절한 크기의 버퍼를 삽입하고 버퍼 의 센서티비티 $\frac{\partial T}{\partial A}|_{buffer}$를 구한다.

3) 만일 $\frac{\partial T}{\partial A}|_{best} > \frac{\partial T}{\partial A}|_{buffer}$ 이면 버퍼를 삽입한다.

4) 버퍼를 삽입한 그 앞의 게이트와 버퍼의 크기를 최소크기로 만든다. 이렇게 하면, TILOS 알고리즘에 의해 다시 사이징되어서 적절한 크기로 만들어진다.

이제까지 보외법을 이용하여 버퍼를 삽입하는 알고리즘에 대해 살펴보았고 전체 알고리즘을 그림 7에 나타내었다.

```
minimum_delay = minimum_sized_circuit_delay
All Transistor size are set to minimum size
  While ( delays at all primary output ≥ T_spec ) {
Compare path delays with T_spec and find the most violating path
  For all gates on the violating path {
    Estimate figure of merit of sizing up a transistor
    Estimate figure of merit for inserting a Type A buffer
    Estimate figure of merit for inserting a Type B buffer
  }
  If (inserting a Type A buffer has the best figure of merit)
    Insert a Type A buffer
  If (inserting a Type B buffer has the best figure of merit)
    Insert a Type B buffer
  else (sizing up a transistor has the best figure of merit)
    Increase the size of a selected transistor
Recompute circuit delays
If (circuit_delay < minimum_delay) minimum_delay = circuit_delay
If (circuit_delay > 1 × minimum_delay) exit // failed to meet specification
}
```

그림 7 보외법을 이용한 버퍼 삽입 알고리즘

3.2절에서는 보외법을 이용한 방법보다 더 정확한 위치에 버퍼를 삽입하여 더 많은 면적감소를 얻을 수 있는 Template Window 시뮬레이션을 이용하여 버퍼를 삽입하는 방법을 소개한다.

3.2 Template Window를 이용한 버퍼 삽입

이 방법은 버퍼를 삽입하는데 있어서 Template

Window를 사용하는 방법[9]이다. 이 방법은 TILOS 알고리즘을 이용하여 사이징을 하는 동시에 Template Window 시뮬레이션을 이용하여 버퍼를 삽입한다. Template Window는 어떤 경로상의 필요한 일부분만을 떼어내서 매핑(mapping)시킨 후 Window 내에서 매핑된 부분만 사이징할 수 있도록 만든 것이다.

다음 그림 8은 Template Window를 이용한 버퍼 삽입의 전반적인 알고리즘을 기술한 것으로 Cost는 원래 회로에서 그 부분의 면적을 나타내고, Cost*는 Template Window에서 사이징되었을 때 최종 면적을 나타낸다.

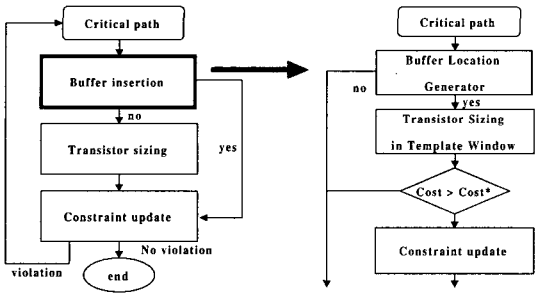


그림 8 Template Window를 이용한 버퍼삽입 알고리즘

그림 8에서 보여지듯이 버퍼 삽입에서 중요한 부분 중의 하나는 버퍼가 들어갈 위치를 찾아내는 것이다(Buffer Location Generator). 본 논문에서는 branch and bound 알고리즘을 이용하였는데, 이 방법은 먼저 모든 팬아웃들을 buffered set으로 초기화하고 unbuffered set으로 하나씩 옮겨보면서 면적을 줄일 수 있다고 예측되는 경우에만 실제로 그 팬아웃을 unbuffered set으로 옮기는 알고리즘이다. 그림 3.4에 이 알고리즘을 정리하였다.

```
Put all fan-outs in S_buffered
  S_config = ∅
  Explore_Possibility( S_buffered, ∅, ΔC(S_buffered, ∅))
  return( S_config)
  Explore_Possibility( S_buffered, S_unbuffer, currentΔC)
    if ( currentΔC > 0 )
       S_config = S_config + ( S_buffered , S_unbuffer)
    for each fan-out F_i in S_buffered
       newΔC = ΔC( S_buffered - F_i , S_unbuffer + F_i)
       if ( currentΔC < newΔC)
          Explore_Possibility( S_buffered - F_i, S_unbuffer + F_i
           newΔC,)
```

그림 9 버퍼 위치를 찾는 알고리즘

그림 9의 알고리즘에서 사용되는 Cost function은 다음과 같다.

$$\Delta T(S_{unbuffer}) = R_G(C_{total} - interconnectCap(S_{unbuffer}) - C_{buffer} - \sum GateCaps(S_{unbuffer}))$$

$$\Delta T(S_{buffered}) = R_G(C_{total} - interconectCap(S_{buffered}) - C_{buffer} - \sum GateCaps(S_{unbuffer})) - bufferDelay$$

$$\Delta Cost_{unbuffer} = \sum_i((\Delta T(S_{unbuffer}) - slack_i)1/sensitivity_i) + CostOf(buffer)$$

$$\Delta Cost_{buffered} = \sum_j((\Delta T(S_{buffer}) - slack_j)1/sensitivity_j)$$

$$\Delta Cost(S_{unbuffer}, S_{buffered}) = \Delta Cost_{unbuffer} + \Delta Cost_{buffered}$$

여기서 R_G는 게이트의 유효 저항(Effective Resistance)을 나타내고, C_{total}, C_{buffer}는 각각 게이트에 걸린 전체 용량성 부하와 버퍼의 게이트의 용량성 부하를 나타내며 $\Delta Cost(S_{unbuffer}, S_{buffered})$는 그림 9의 알고리즘에서 ΔC를 나타낸다. 이 Cost Function들에서는 버퍼가 삽입되었을 경우와 버퍼가 삽입되지 않았을 경우의 지연의 향상(ΔT)을 계산하여 반영했다. 이 알고리즘을 이용하여 버퍼가 삽입되면 회로의 면적을 줄일 수 있을만한 위치를 찾아내어 그 위치를 저장한다. 그 후 실제로 그 위치에 버퍼가 삽입될 수 있는지를 Template Window를 이용하여 확인한다.

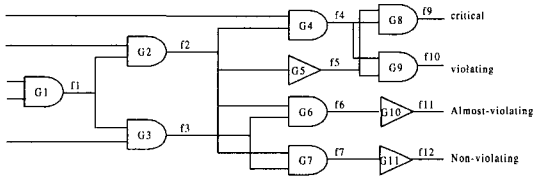


그림 10 예제 회로

Template Window에 매핑하는 과정을 설명하기 위해 그림 10과 같은 회로가 있다고 가정한다. 이와 같은 회로의 경우 게이트 G2에 팬아웃이 많기 때문에 G2의 팬아웃쪽에 버퍼를 삽입함으로써 팬아웃을 분리해 내서 임계경로 쪽의 지연을 줄일 수 있고, G2가 과다하게 커지는 경우를 방지할 수 있다.(형식 B 버퍼 삽입) 이 때, 게이트 G2 와 게이트 G6 사이에 버퍼가 삽입되는 것이 적당하다고 가정한다면, 이 부분을 Template Window로 매핑 시킨다. 이를 그림 11에 제시하였다.

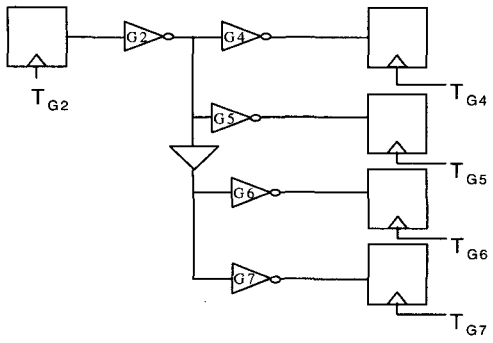


그림 11 Template Window 매핑

그림 11에서 보는 바와 같이, 그림 10에서 버퍼가 삽입될 위치의 바로 전 게이트(버퍼 입력)를 인버터로 매핑시키고 그 게이트의 입력에 래치를 연결시킨다. 그리고 이 래치에 G2의 입력이 들어오는 시간 정보를 클럭으로 준다. 또, G2의 팬아웃에 걸린 모든 게이트들을 인버터로 매핑시키고 그 팬아웃에 각각 래치를 연결시키고 각각의 게이트들이 출력을 내는 시간 정보를 클럭으로 준다. 이와 같이 게이트 G2 이후에 하나의 레벨만 Template Window에 매핑하면 Template Window 내에서의 면적-지연 곡선이 급격히 상승한다. 따라서 Template Window에 매핑 시킬 때 인버터의 팬아웃 노드에 원래 회로의 팬아웃에 걸린 용량성 부하들을 계산해 넣어 주고, 래치에 지연을 조금 더 걸어 주면 면적-지연 곡선을 보다 완만하게 만들 수 있으며 더 정확한 결과를 얻을 수 있다.

인버터는 일반적인 게이트와는 달리 각각 하나씩의 pull up과 pull down 트랜지스터만을 가지고 있다. 따라서 본 논문에서는 Template Window를 이용하여 지연을 계산할 때, 이를 고려하기 위해 Multiplier Factor를 사용했다. Multiplier Factor는 각각의 게이트 G_i 의 pull up 과 pull down 각각에 대해 직렬로 연결된 트랜지스터의 개수를 나타낸다. 그림 12와 같이 3개의 입력을 가지는 NAND 게이트가 인버터로 매핑 되는 과정을 보자.

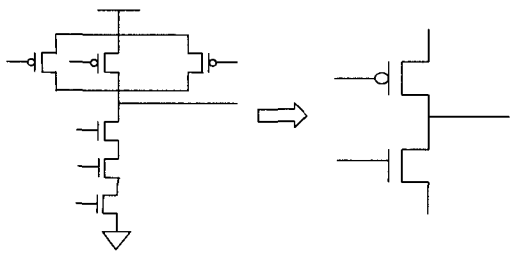


그림 12 NAND 게이트를 인버터로 매핑

3개의 입력을 가지는 NAND 게이트에서는 n-트랜지스터들이 직렬로 3개가 연결되어 있으므로 인버터의 pull down 트랜지스터에 Multiplier Factor로 3을 저장하고, Template Window 내에서 사이징할 때마다 이 factor를 이용하여 지연을 계산한다. 마찬가지로 p-트랜지스터의 경우에는 직렬로 연결된 트랜지스터들이 없고 모두 병렬로 연결되어 있으므로 인버터의 pull up의 경우 multiplier factor는 1이 된다. 이 경우 간단한 RC-지연 모델을 사용하였을 때, pull up 지연은 $t_r = R_p C_L$과 같이 계산되어지고, pull down 지연은 $t_f = mR_n C_L$ ($m=3$) 과 같이 계산된다. 여기서 R_p는 인버터의 pull up 트랜지스터의 저항 값을 나타내고, R_n은 pull down 트랜지스터의 저항 값을 나타낸다. 또, C_L 은 팬아웃에 걸린 모든 용량성 부하 값을 나타낸다. 다시 말하면, $j \in fanout_i$ 일 때 $C_L = \sum(C_{gn_j} + C_{gp_j}) + C_{interconnect}$ 과 같이 나타내어진다.

이제까지 트랜지스터 사이징과 동시에 버퍼를 삽입하는 알고리즘 2가지를 소개하였으며, 다음 장에서 이 두 가지 알고리즘을 이용하여 버퍼를 삽입하였을 때의 실험결과를 비교하고 분석한다.

4. 실험결과

이제까지 Template Window와 보외법을 이용한 버퍼 삽입을 소개하였다. 소개된 Template Window를 이용한 버퍼 삽입 실험은 C언어로 구현되었으며 ULTRA SPARC-I WorkStation 하에서 benchmark85 회로와 benchmark89 회로를 SPICE Netlist로 변환하여 실험하였다. 표에서는 Template Window과 보외법을 이용하여 버퍼를 삽입했을 때의 결과를 보여주고 있는데 TW는 Template Window를 사용하여 버퍼를 삽입했을 때의 결과를 나타내며, EP는 보외법을 사용하여 버퍼를 삽입했을 때의 결과를 나타낸다. 각각의 수치는 전체 회로의 총 면적을 나타내고 있다. 여기서 보여지는 바와 같이 보외법을 이용하여 버퍼를 삽입한 경우 사이징만 수행했을 때 보다 평균 10% 정도의 면적 감소를 얻을 수 있었고, Template Window를 이용하여 버퍼 삽입을 하였을 경우 면적은 평균 16% 정도 감소하였다. 대부분의 경우 적절한 지연제약 하에서는 트랜지스터 사이징을 하면서 동시에 버퍼를 삽입하여 주었을 때, 사이징만 수행한 것 보다 더 좋은 결과를 얻을 수 있었다. C499, S298과 같이 버퍼

가 삽입되지 않았던 회로는 그 회로 내의 게이트들의 팬아웃에 걸린 용량성 부하들이 비교적 작았기 때문이었다.

Template Window를 사용한 경우와 보외법을 사용했을 때의 차이점은 Template Window를 사용한 경우는 버퍼를 먼저 삽입하지 않고, Template Window에서 시뮬레이션을 통해 버퍼 삽입의 적합성을 확인한 후 버퍼를 삽입한다는 것이며, 이에 반해 보외법을 이용한 경우는 먼저 버퍼를 삽입하고 지연을 계산하고, 똑같은 지연으로 맞추었을 때, 면적 계산을 보외법을 이용하여 계산한다는 점이다.

결과적으로 보외법을 이용한 방법은 더 많은 수의 버퍼를 삽입해 줌에도 불구하고 더 적은 면적 감소를 나타냈다. 다시말하면, 보외법을 이용한 방법 보다 회로의 일부분을 직접 시뮬레이션 하여 버퍼 삽입 여부를 판단한 Template Window를 이용한 방법이 더 적절한 위치에 버퍼를 삽입하여 더 많은 면적 감소를 얻을 수 있었으며, 보외법의 경우 전체적으로 사이징만 수행했을 때보다 좋은 결과를 나타냈으나 버퍼가 들어가지 않을 위치에도 버퍼를 삽입하여 Template Window를 이용한 방법보다 더 많은 버퍼를 삽입함에도 불구하고 좋지 않은 결과를 가져왔다.

이 두 가지 방법 모두 부분적으로 최적의 지점(Local Optimal Point)에 버퍼를 삽입하는 방법으로 트랜지스터들이 계속 사이징 되어 가는 것을 고려한 전체적인 최적의 지점(Global Optimal Point)을 정확히 찾아주지는 못하기 때문에, 만일 지연의 제약이 너무 심한 경우에는 오히려 전체 면적이 늘어날 수도 있다.

5. 결론 및 향후 과제

트랜지스터 사이징을 하면서 동시에 버퍼를 삽입하면 사이징만 수행했을 때보다 더 좋은 결과를 얻을 수 있었다. 제안된 알고리즘들은 TILOS 알고리즘을 이용하여 트랜지스터 사이징을 하는 동시에 버퍼가 들어갈 만한 위치를 찾아 그 곳에 버퍼를 삽입하는 것이다. 이 때, 버퍼의 위치의 적합성을 판단하기 위해서는 버퍼를 삽입하고 다시 사이징을 해 보아야 하는데, 이런 경우 시간이 매우 많이 걸리게 된다. 특히 회로의 크기가 크고 버퍼가 들어갈 만한 위치가 많은

표 1 버퍼 삽입 결과

circuit	cycle time	Sizing only (Area)	Sizing with buffer(EP) (Area)	#of buffer (EP)	improve--ment(EP) (%)	sizing with buffer TW (Area)	#of buffer (TW)	improve--ment(TW) (%)
C499	15ns	5.104e-03	5.104e-03	0	0	5.104e-03	0	0
	20ns	2.868e-03	2.868e-03	0	0	2.868e-03	0	0
C880	12ns	3.093e-03	2.991e-03	4	3	2.779e-03	3	10
	15ns	2.372e-03	2.289e-03	3	3	292e-03	1	7
C1355	15ns	4.894e-02	4.502e-03	5	8	4.479e-02	5	9
	18ns	3.326e-03	3.225e-03	5	3	3.101e-03	4	6
C1908	18ns	7.591e-02	6.085e-02	4	19	5.747e-02	1	24
	20ns	6.205e-03	5.044e-03	3	18	4.753e-03	1	23
C2670	22ns	7.561e-03	5.899e-03	4	21	5.235e-03	3	31
	25ns	6.554e-03	5.383e-03	3	17	5.203e-03	1	21
C3540	35ns	1.003e-02	9.186e-03	5	8	8.309e-03	5	17
	40ns	8.935e-03	7.952e-03	3	11	7.254e-03	2	19
C5315	25ns	1.239e-02	1.087e-02	5	12	1.043e-02	2	15
	30ns	145e-02	1.005e-02	3	12	9.144e-03	1	20
S298	5ns	7.152e-04	7.152e-04	0	0	7.152e-04	0	0
	10ns	6.302e-04	7.152e-04	0	0	6.302e-04	0	0
S1196	15ns	4.388e-03	3.944e-03	4	10	3.799e-03	2	13
	20ns	3.688e-03	3.371e-03	4	9	3.039e-03	2	18

경우에 각각의 위치마다 버퍼를 넣어보고 처음부터 다시 사이징해야 한다면, 사이징하는데 시간이 상당히 오래 걸린다. 이를 방지하기 위해 Template Window 시뮬레이션과 보외법을 이용한 버퍼삽입을 소개하고 비교했다. Template Window를 이용하는 방법은 버퍼가 들어갈 부근의 게이트들을 인버터로 매핑시키고 래치를 연결하여 그 부분에서만 사이징을 하는 방법으로써 단순히 버퍼 삽입에만 적용되는 것이 아니라 회로의 어느 한 부분만 사이징을 하는 경우나 한 경로 혹은 부분만 시뮬레이션하는 경우에서 이용될 수 있다. 보외법을 이용하는 방법은 버퍼가 들어갈만한 위치에서 버퍼를 삽입하여 줄어드는 지연을 구하고 그와 똑같이 맞추어 사이징할 때 늘어나는 면적을 보외법을 통해 구하는 방법이다. 적절한 시간 제약 하에서 두 가지 방법 모두 버퍼를 삽입하지 않고 사이징만 수행했을 때 보다 전체 회로의 면적을 줄일 수 있었는데, Template Window를 이용한 방법이 보외법을 이용한 방법보다 더 작은 개수의 버퍼를 삽입하면서도 더 많은 면적 감소를 얻을 수 있었다. 이는 보외법을 이용한 방법은 버퍼가 삽입되지 않아도 될 위치에 부적절하게 버퍼를 삽입했기 때문이다.

본 논문에서는 지연 시간을 계산함에 있어서 트랜지스터의 지연 시간만 고려하였으며, 회로의 면적도 트랜지스터 게이트의 면적으로만 계산하였다. 그러나 요즈음에는 트랜지스터 게이트의 크기가 점점 작아지는 추세이기 때문에, 내부 결선(interconnection)의 지연 시간과 회로의 면적에서 내부 결선이 미치는 영향이 상대적으로 증가하고 있다. 따라서 내부 결선의 지연시간과 회로의 면적에 미치는 영향을 고려하여 트랜지스터를 사이징함과 동시에 자동적으로 버퍼를 삽입해 주는 알고리즘의 개발이 필요하다.

참 고 문 헌

[1] Jose Monteiro and Srinvas Devadas, Computer-Aided Design Techniques For Low Power Sequential Logic Circuits, Kluwer Academic Publishers, 1997

[2] J. Fishburn and A. Dunlop, "TILOS : A posynomial programming approach to transistor sizing," in Proceeding of the IEEE International Conference on Computer-Aided Design, pp. 326-328, 1985

[3] S. S. Sapatnekar, V. B. Rao, P. M. Vaildya, and S. M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," IEEE Transactions on Computer-Aided Design, vol. 12, pp. 1621-1634, Nov. 1993

[4] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor sizing for low power CMOS circuits," IEEE Transactions on Computer-Aided Design. vol. 15. pp 665-671, June 1996

[5] K. J. Singh and A. Samgiovanni-Vincentelli, "A heuristic algorithm for the fanout problem," in Proceedings of the ACM/IEEE Design Automation Conference, pp. 357-360, 1988.

[6] C. L. Berman, J. I,. Carter, and K. L. Day, "The fanout problem : From theory to practice," in Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conference, pp. 69-99, 1989.

[7] Yanbin Jiang, Sachin Sapatnekar, Cyrus Bamji and Juho Kim, "Interleaving Buffer Insertion and Transistor Sizing into a Single Optimization," IEEE Transactions VLSI systems. pp625-633, 1998

[8] A. Raiston and P. Ravinowitz, A First Course in Numerical Analysis. New York: McGraw-Hill, 1978.

[9] Juho Kim, Cyrus Bamji, Yanbin Jiang and Sachin Sapatnekar, "Concurrent Transistor Sizing and Buffer Insertion by Considering Cost-Delay Tradeoffs," Proceedings of 1997 International Symposium on Physical Design, pp. 130-135.

[10] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delayin RC tree networks," IEEE Transactions on Computer-Aided Design, vol. CAD-2, pp. 202-211, July 1983.

[11] W. C. Elmore. "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," Journal of Applied Physic, vol 19(1), pp. 55-63, 1948

[12] C. L. Fang and W. B. Jone, "Timing Optimization by Gate Resizing and Critical Path Identification," IEEE Transactions on Computer Aided Design, Vol.14, No 2, pp.201-217, February 1995.

[13] J. Kim, Y. Hsu, and D. Du, "A New Gate Selection Method for Resizing to Circuit Performance Optimization," in Proc. ISCAS, Vol. 4, pp.461-464, May 1996.

[14] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor Sizing for Minimizing Power Consumption of CMOS Circuits under Delay Constraint," in Proc. Int'l Symp. on Low Power Design, pp.167-172, April 1995.

[15] M. Sarrafzadeh and D. S. Chen, "An Exact Algorithm for Low Power Library-Specific Gate Resizing," in Proc. 33rd Design Automation Conference, pp.783-788,June 1996.

[16] P. Girard, C. Landrault, S. Pravossoudovich, and D. Severac, "A Gate Resizig Technique for High

Reduction in Power Consumption," in Proc. Int'l Symp. on Low Power Design, pp.281-286, August 1997.

[17] C. H. Tan and J. Allen, "Minimiztion of Power in VLSI Circuits Using Transistor Sizing, Input Ordering, and Statistical Power Estimation," in Proc. Int'l Symp. on Low Power Design, pp.75-80, August 1994.

이 성 건

1997년 8월 서강대학교 수학과 학사. 1999년 2월 SiliconCraft, Inc. 입사. 2000년 2월 서강대학교 컴퓨터학과 석사. 현재 SiliconCraft, Inc. R&D Engineer

김 주 호

1987년 University of Minnesota at Minneapolis 전산과 박사. 1995 ~ 1996년 Cadence Design System Inc. San Jose, California, Senior Member of Technical Staff. 1997년 ~ 현재 서강대학교 컴퓨터학과 교수. 관심분야는 CAD 및 하드웨어 시스템 설계