논문 2007-44SD-11-17

크로스톡 회피를 위한 게이트 사이징을 이용한 타이밍 윈도우 이동

(Timing Window Shifting by Gate Sizing for Crosstalk Avoidance)

장 나 은*, 김 주 호**

(Naeun Zang and Juho Kim)

요 약

본 논문은 CMOS 디지털 회로에서 delay에 영향을 미치는 crosstalk을 gate의 downsizing이나 upsizing으로 발생을 회피하기 위한 효율적인 휴리스틱 알고리즘을 제시한다. 제안된 알고리즘은 게이트 사이징을 2가지 step으로 분류하며 avoidance 효과를 극대화하기 위해서 step1에서는 downsizing, step2에서는 upsizing을 순차적으로 적용하여 critical path에 인접하는 aggressor들을 차례로 회피해 나간다. 제시된 알고리즘은 LGSynth91 벤치마크 회로에 대한 테스트 결과 효율성을 검증 하였으며 실험 결과는 평균적으로 8.64%의 Crosstalk Avoidance 효과를 보여줬다. 이 결과로 제시된 새로운 알고리즘의 가능성을 입증하였다.

Abstract

This paper presents an efficient heuristic algorithm to avoid crosstalk which effects to delay of CMOS digital circuit by downsizing and upsizing of Gate. The proposed algorithm divide into two step, step1 performs downsizing of gate, step2 performs upsizing, so that avoid adjacent aggressor to critical path in series. The proposed algorithm has been verified on LGSynth91 benchmark circuits and Experimental results show an average 8.64% Crosstalk Avoidance effect. This result proved new potential of proposed algorithm.

Keywords : crosstalk, optimization, timing, analysis, estimation

I. 서 론

휴대용 계산기기와 모바일 시스템의 급격한 성장으로 고성능 고집적화에 대한 요구가 증가하고 있다. 하지만 고성능 고집적화에 따른 노이즈의 효과는 더욱 증폭되고 있는 실정이다. 현재 누설전류(crosstalk)에 대한 접근 방법은 회로에 대한 최악의 경우 지연시간(worst case delay)을 구하기 위한 노이즈의 분석에 중점을 두고 있다. 이 접근 방법은 누설전류를 어떻게 모델링하며, 내부결선간의 거리는 가까우나 논리적으로 영향을 미치지 않는 어그레서들이나, 시간적으로 서로 영향을 미치지 않는 어그레서들을 찾아서 최악 경우 지연시간의 계산에 참고하지 않는 것이다.

누설전류에 대한 여러 가지 모델링 방법은 [1][2]에 제시되어 있으며, 정적 시간 분석기 (Static Timing Analysis)에서 어떻게 누설전류를 다룰 것이지는 [3]에 제시되어 있다. 또한 어그레서들을 여러 가지 알고리즘을 적용시켜 어그레서로서 작동하지 않는 것을 찾아내고 어그레서에서 제거해 나가는 방법인, 누설전류 프루닝은 타이밍 윈도우(timing window)를 사용하는 방법은 [4]에 이진 결정 다이어그램(bdd)을 사용하는 방법은 [5]에 혼합한 방법은 [6]에 제시되어 있으며 현재 아직도 많은 연구가 활발히 이루어지고 있다. 이 방법은 그 이전에 있었던 전통적인 STA with Signal Integrity

* 학생회원, * 정회원, 서강대학교 컴퓨터공학과
(Department of Computer Science and Engineering, Sogang University)

※ 본 연구결과물은 2007년도 「서울시 산학연 협력사업」의 「나노IP/SoC설계기술혁신사업단」의 지원으로 이루어졌음

접수일자: 2007년4월12일, 수정완료일: 2007년10월24일

(모든 누설전류에 대해서 switching factor나 miller factor를 적용하는)에 비해서 보다 정확한 최악 경우 지연시간을 제공할 수 있다. 하지만 이런 알고리즘들은 어디까지나 회로의 수정이 없는 누설전류 분석 수준의 대응이며 누설전류 자체를 제거하는 적극적인 방법은 아니므로 회로의 근본적인 디자인 한계를 극복할 수는 없다.

본 논문의 알고리즘에 적용된 게이트 사이징(gate sizing)은 일반적으로 CMOS 디지털 회로에서 신호 천이에 의해 발생하는 동적 전력 소모(dynamic power dissipation)에서 기능에 영향을 미치지 않는 불필요한 기능성 천이(functional transition)인 글리치(glitch)를 제거하기 위해서 사용되어 왔다. 이러한 방법들은 [7][8]에서 제시하고 있다. 전통적인 기법의 게이트 사이징과 버퍼 삽입은 회로의 전력과 지연시간 최적화를 위한 효율적인 방법이다.

이제부터 전혀 어울리지 않을 것 같은, 누설전류를 게이트 사이징으로 회피하는 방법을 논문에서 제안할 것이다. 이 방법은 게이트 사이징에 대한 이해와 누설전류의 이해에서 출발하며 아직까지 시도되지 않은 알고리즘이다.

본 논문은 다음과 같이 구성된다. Ⅱ장은 알고리즘에서 사용되는 누설전류 모델링의 한 방법인 타이밍 윈도우 모델링(timing window modeling), 게이트 수준의 지연시간, 사이징 효과에 대해서 논의한다. Ⅲ장은 게이트 사이징으로 누설전류의 회피에 대해서 논의하며 Ⅳ장은 구현된 알고리즘에 의한 실험결과를 보여주며, 마지막으로 Ⅴ장은 앞으로 계획하고 있는 연구에 대한 언급을 하며 결론으로 끝을 맺는다.

Ⅱ. 크로스톡 (Crosstalk)

1. Noise Effect

일반적으로 누설전류 노이즈는 회로 기능에 두 가지 방법으로 영향을 준다. 첫 번째는 신호무결성 (Signal Integrity)이고 두 번째는 지연감소 (Delay Degradation)이다. 전자는 그림 1과 같이 정적인 신호에 노이즈 커플이 발생하여 빅팀에 글리치를 일으키게 되어 틀린 논리 값이 발생하는 것이다. 후자는 그림 2와 같이 천이하는 신호에 노이즈 커플이 발생하여 지연시간(delay)이 증가하는 것이다.

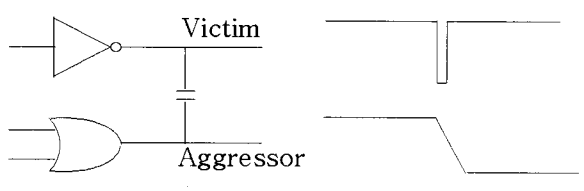


그림 1. 신호 무결성 문제
Fig. 1. Signal Integrity Problem.

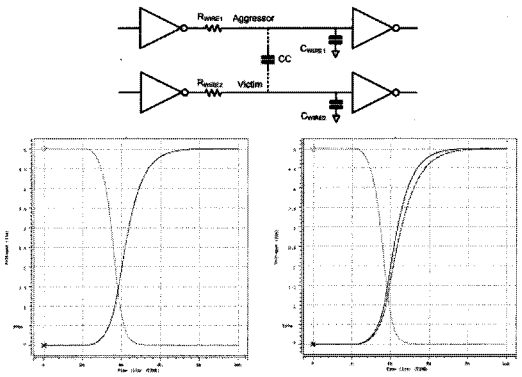


그림 2. 지연시간 감소
Fig. 2. Delay Degradation.

2. Crosstalk modeling

레이아웃에서 두개의 인접한 넷들은 누설전류에 민감하다. simplified lumped RC model을 통해서 우리는 노이즈 수준을 계산할 수 있다.[9] 물리적인 레이아웃에서 빅팀과 어그레서를 고려하면 빅팀 넷에서의 신호파형은 [9]에서 보여준 것과 같이 다음과 같이 계산된다.

$$V_v(t)=\begin{cases} V_{DD}-V_{DD}\dfrac{C_x R_v}{t_f}(1-e^{-t/R_vC_t}), & 0<t\le t_f \\ V_v(t_f)e^{-(t-t_f)/R_vC_t}+V_{DD}(1-e^{-(t-t_f)/R_vC_t}), & t>t_f \end{cases}$$

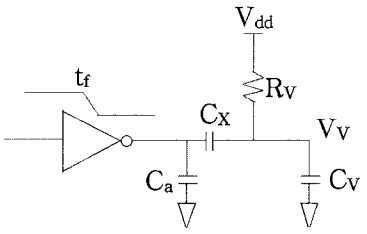


그림 3. 잡음 수준 모델
Fig. 3. Noise Level Model.

그림 3을 보면 Cv, Ca, Cx는 각각 빅팀 넷, 어그레서 넷, 커플링의 캐패시턴스며 Ct = Cv + Cx 이다. Rv는 빅팀 넷의 저항이다. 이 값들은 최악의 경우 모델(worst case model)에 쓰이고 있다.

3. Temporal Pruning

누설전류 노이즈는 넷 각각의 타이밍 윈도우와 관련이 있다. 예를 들어 그림 4에서 A1의 타이밍 윈도우가 V의 타이밍 윈도우와 겹쳐 있는데 이것은 A1이 V에게 영향을 미친다는 것을 의미한다. 반면에 A2는 V와 겹치지 않는데 이것은 넷 V에게 어떠한 누설전류 노이즈도 발생하지 않는다는 것을 의미한다.

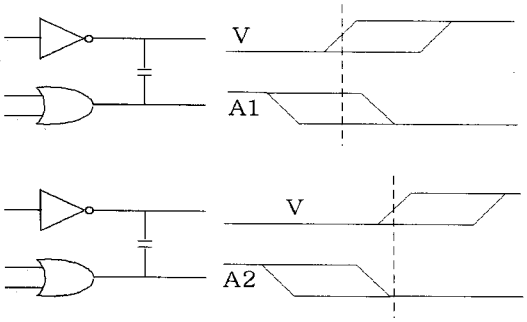


그림 4. 빅팀넷과 어그레서넷 사이의 시간적인 관계
Fig. 4. Temporal Relationship between Victim Net and Aggressor Net.

이러한 접근은 정적 노이즈 분석 모델(static noise analysis model)에 반영되어 있는데 기능적(functional) 정보는 무시한 채 시간(timing) 정보만 포함된 것이다.

4. Functional Pruning

경로의 기능적인(functional) 관계에 의해서 노이즈에 대해서 절대 반응하지 않는 경로(path)들을 어그레서에서 제거하는 것이며 그 알고리즘에 따라 이진 결정 다이어그램(BDD)을 이용하거나 path sensitization을 이용하기도 한다.[5] 이 접근은 zero-delay model을 가정하고 있으며 역시 시간(temporal) 정보는 무시된다.

Ⅲ. 게이트 사이징을 이용한 크로스톡 회피

기존의 방법은 누설전류의 분석에 의미를 두어왔으며 더욱 정확한 분석을 통해서 functional, temporal pruning을 수행했다. 하지만 보다 적극적인 방법으로 전력 감소에만 쓰이던 게이트 사이징(gate sizing)을 사용하여 누설전류가 발생하는 부분을 가능한 회피할 수 있는 휴리스틱 알고리즘을 본 논문에서 제시할 것이다. 본 논문은 누설전류의 기능적인(functional)적인 측면은 고려하지 않는다.

1. 알고리즘 개념도

전체적인 알고리즘 흐름도는 그림 5와 같다. Cross-talk Avoidance 알고리즘의 입력으로는 회로에 대한 도착시간(arrival time)과 요구시간(require time) 그리고 회로의 어그레서와 빅팀 정보이다. 출력은 그림과 같이 게이트 사이징으로 수정된 crosstalk avoided netlist이다.

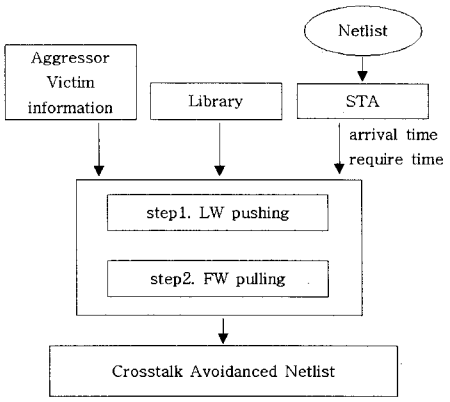


그림 5. 전체 흐름도
Fig. 5. Overall Flow Graph.

2. timing window modeling

기존의 타이밍 윈도우 모델링과는 달리 본 논문에서는 빅팀과 어그레서의 구분을 두지 않는다. 단지 둘 중 어떤 타이밍 윈도우가 우선하느냐에 따라서, 우선하는 노드의 타이밍 윈도우를 FW(Former Window)로 정의하며 우선하지 않는 노드의 타이밍 윈도우를 LW (Latter Window)로 정의한다. FWLT(Former Window Latest Time)는 FW의 스위칭이 끝나는 시간을 가리키며 LWET(Latter Window Earliest Time)는 LW의 스위칭이 시작하는 시간을 가리킨다. Td(timing difference)는 FWLT에서 LWET를 뺀 값으로 정의한다. 그림 6은 본 논문에서 사용하는 타이밍 윈도우 모델링을 표현하고 있다.

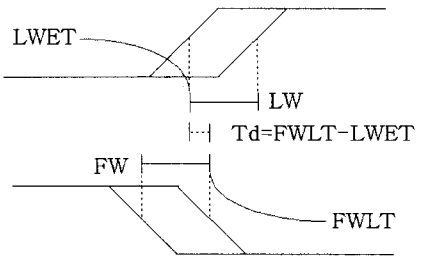


그림 6. 타이밍윈도우 모델링
Fig. 6. Timing Window modeling.

3. 크로스톡 회피 알고리즘

가. step1: LW pushing

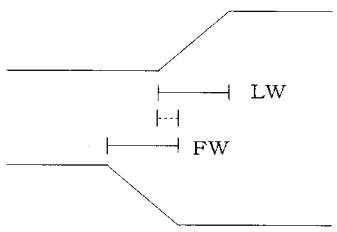


그림 7. 누설전류 발생
Fig. 7. Crosstalk appearance.

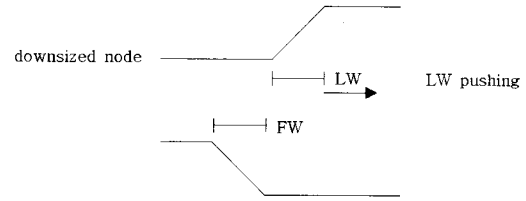


그림 8. LW 밀기 (slack≥Td)
Fig. 8. LW pushing (slack≥Td).

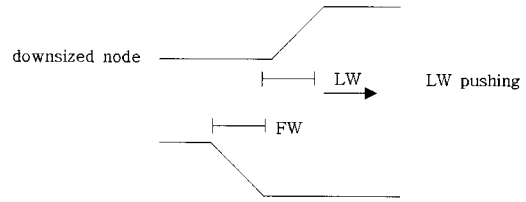


그림 9. LW 밀기 (slack<Td)
Fig. 9. LW pushing (slack<Td).

그림 7과 같이 타이밍 윈도우가 서로 겹치게 되어서 누설전류가 발생하는 경우 LW측의 노드의 게이트의 사이즈를 작게 하여 게이트의 동작 속도를 느리게 한다. 그렇게 하면 그림 8과 같이 서로 겹치는 부분은 사리지게 되며 누설전류를 회피하게 된다. 만약 LW의 슬랙이 Td보다 작다면 슬랙만큼 pushing을 수행한다. 이 경우는 그림 9에 나타나 있으며 완벽한 회피가 되지는 않는다. 하지만 그림 7보다는 겹치는 부분이 적어지므로 그만큼 누설전류의 효과가 약해진다는 것을 의미한다. 여기서 고려해야 할 사항은 해당 노드의 슬랙이다. 슬랙을 넘어서 pushing을 할 경우에 다음노드의 도착시간(arrival time)이 요구시간(required time)을 넘어서는 경우가 발생하므로 절대 슬랙을 넘어서 pushing을 하지는 않는다.

step1, LW pushing 알고리즘은 다음과 같다.

```
if ( slack >= Td ) {
   select biggest gate Gi such that slack >= d(Gi) > Td --(1)
   if ( Gi exist ) {
      mark gate Gi as avoidance step1 resizable
   }
}if else ( slack < Td ) {
   select smallest gate Gi such that slack >= d(Gi)-------(2)
   if ( Gi exist ) {
      mark gate Gi as avoidance step1 resizable
   }
}

else abandon avoidance step1
```

위에 제시된 알고리즘 중

Gi는 pushing에서 다운 사이징될 게이트이며,

d(Gi)는 사이징될 게이트의 기존 게이트에 대한 추가 delay(사이징될 게이트의 delay에서 원래 게이트의 delay를 뺀 값),

LWET는 Latter Window의 Earliest Time,

FWLT는 Former Window의 Latest Time,

Td = FWLT - LWET를 의미한다.

(1)은 Td가 슬랙보다 작거나 같은 경우 (LW pushing으로 충분히 누설전류 회피가 가능)

다운사이징으로 추가되는 delay가 슬랙보다 작거나 같고, Td보다 큰, 가장 큰 새로운 게이트 Gi를 라이브러리에서 선택한다.

(불필요하게 타이밍 윈도우를 띄우지 않는다. 불필요한 LW pushing은 다른 노드에 또 다른 누설전류를 야기 시킬 수 있기 때문이다.)

그림 10에서 보면 그러데이션 부분의 보다 진한 부분으로 라이브러리에서 선택하게 된다. Td보다 큰 지연시간을 선택하므로 게이트의 크기는 줄어들게 되며 게이트 다운사이징이 된다. 그리고 LW가 게이트 다운사이징에 의해서 지연시간이 늘어나므로 LW는 그 지연시간만큼 pushing이 된다. 그림 10은 LW pushing 알고리즘 중 (1)에 대한 묘사이다. (1)의 알고리즘을 수행하고

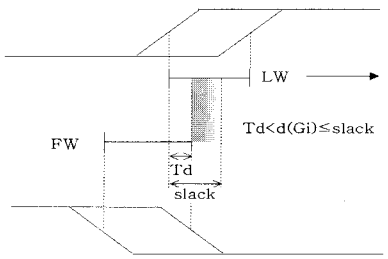


그림 10. LW 밀기의 (1)에 대한 게이트 다운사이징
Fig. 10. Gate downsizing of LW pushing (1).

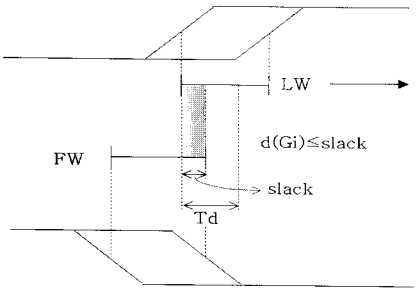


그림 11. LW 밀기의 (2)에 대한 게이트 다운사이징
Fig. 11. Gate downsizing of LW pushing (2).

난 뒤에는 그림 8과 같이 누설전류 회피가 구현된다.

(2)는 Td가 슬랙보다 큰 경우(LW pushing으로 완전한 누설전류 회피가 불가능할 경우)

다운사이징으로 추가되는 지연시간이 슬랙보다 작은 범위 내에서, 사이즈가 가장 작은 새로운 게이트 Gi를 라이브러리에서 선택한다.

(슬랙이 허락하는 범위까지 LW pushing을 수행한다.)

그림 11을 보면 그러데이션 부분의 보다 진한 부분으로 라이브러리에서 선택하게 된다. 슬랙이 허락하는 선까지 pushing을 수행하게 되며 앞에서 설명한 바와 같이 누설전류 회피는 아니지만 누설전류의 부정적인 효과를 최대한 약화시킬 수 있다. 그림 11은 LW pushing 알고리즘 중 (2)에 대한 묘사이다. (2)의 알고리즘을 수행하고 난 뒤에는 그림 9와 같이 타이밍 윈도우의 겹치는 부분을 최소화해 준다.

만약 라이브러리에 이러한 게이트 (1)과 (2)를 만족하는 delay를 가진 Gi가 존재하지 않을 경우 step1에서는 회피를 고려하지 않는다.

나. step2: FW pulling

그림 7과 같이 타이밍 윈도우가 서로 겹치게 되어서 누설전류가 발생하는 경우 FW측의 노드의 게이트의 사이즈를 크게 하여 게이트의 동작 속도를 빠르게 한다. 그렇게 하면 그림 12와 같이 서로 겹치는 부분은 사

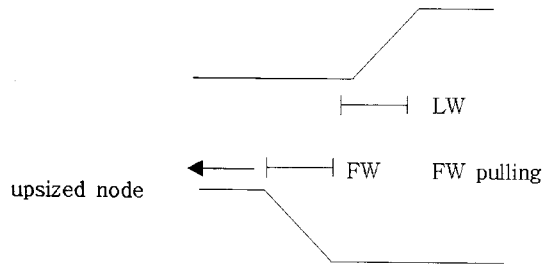


그림 12. FW 당기기 (d(Gj) ≥ Td)
Fig. 12. FW pulling (d(Gj) ≥ Td).

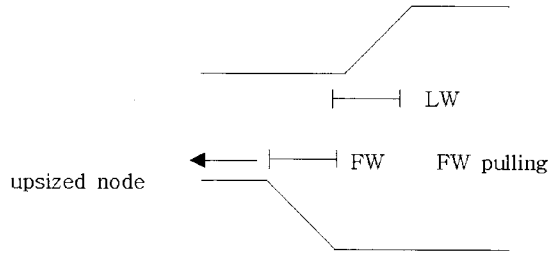


그림 13. FW 당기기 (d(Gj) < Td)
Fig. 13. FW pulling (d(Gj) < Td).

라지게 되며 누설전류를 회피하게 된다. 만약 FW측 노드의 게이트를 가장 큰 게이트를 넣어도 이전 게이트와 사이징된 게이트의 차가 Td를 넘지 못할 경우 가능한 큰 게이트로 사이징함으로써 가능한 만큼만 pulling을 수행한다. 이 경우는 그림 13에 나타나 있으며 pushing의 (2) 경우처럼 완벽한 회피가 되지는 않는다. 하지만 그림 7보다는 겹치는 부분이 적어지므로 그만큼 누설전류의 효과가 약해진다는 것을 의미한다.

step2, FW pulling 알고리즘은 다음과 같다.

```
if ( d(Gj) >= Td ) {
  select smallest gate Gj such that d(Gj) >= Td ------(3)
  if ( Gj exist ) {
     if (overlap with prenode) abandon
     mark gate Gj as avoidance step2 resizable
   }
 }

if else ( d(Gk) < Td ) {
  select biggest gate Gk in library  ----------------(4)
  if (overlap with prenode) abandon
  mark gate Gj as avoidance step2 resizable
 }

else abandon avoidance step2
```

Gj는 pulling에서 업사이징될 게이트이며,

d(Gj)는 기존 게이트에서 사이징될 게이트에 대한 지연시간 차이(원래 게이트의 지연시간에서 사이징될 게이트의 지연시간을 뺀 값)이다.

d(Gk)는 기존 게이트에서 가장 큰 게이트에 대한 지연시간 차이(원래 게이트에 지연시간에서 라이브러리에서 가장 큰 게이트의 지연시간을 뺀 값)이다.

(3)은 Td가 d(Gk)보다 작거나 같은 경우, (FW pulling으로 충분히 누설전류 회피가 가능한 경우)

업사이징으로 제거되는 지연시간이 Td보가 크거나

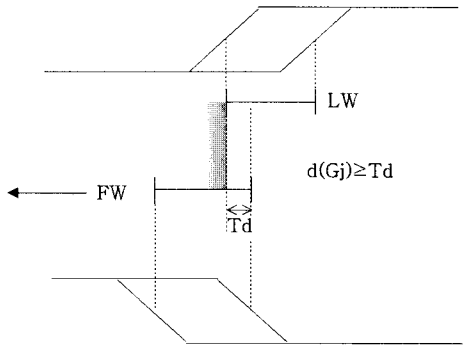


그림 14. FW 당기기의 (3)에 대한 게이트 업사이징
Fig. 14. Gate upsizing of FW pulling (3).

같은, 가장 작은 새로운 게이트 Gj를 라이브러리에서 선택한다.

(불필요하게 타이밍 윈도우를 띄우지 않는다. 불필요한 FW pulling 역시 다른 노드에 또 다른 누설전류를 야기할 수 있기 때문이다.)

그림 14의 그러데이션 부분의 보다 진한 부분으로 라이브러리에서 선택하게 된다. Td보다 더 많은 지연시간을 제거하므로 게이트의 크기는 커지며 게이트 업사이징이 된다. 그리고 FW가 게이트 업사이징에 의해서 지연시간은 줄어들게 되므로 FW는 그 지연시간만큼 pulling이 된다. 그림 14는 FW pulling 알고리즘 중 (3)에 대한 묘사이다. (3)의 알고리즘을 수행하고 난 뒤에는 그림 15와 같이 누설전류 회피가 구현된다.

(4)는 Td가 d(Gk)보다 큰 경우, (가장 큰 게이트 사이징으로 FW pulling을 해도 완전한 누설전류 회피가 불가능할 경우)

사이즈가 가장 큰 새로운 게이트 Gj를 라이브러리에서 선택한다. (가장 큰 게이트 사이징으로 가능한 LW pushing을 수행)

그림 15를 보면 그러데이션 부분의 보다 진한 부분으로 라이브러리에서 선택하게 되며 Td가 허락하는 선까

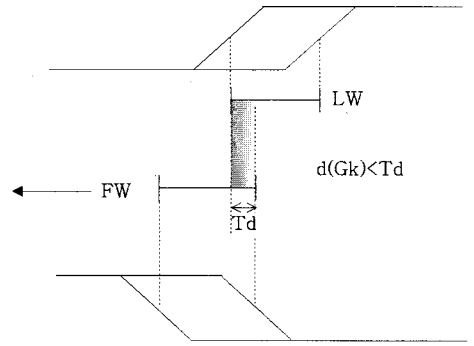


그림 15. FW 당기기의 (4)에 대한 게이트 업사이징
Fig. 15. Gate upsizing of FW pulling (4).

지 pulling을 수행하게 된다. 앞서 말한 바와 같이 누설전류 회피는 아니지만 누설전류의 부정적인 효과를 최대한 약화시킬 수 있다. 그림 15은 FW pulling 알고리즘 중 (4)에 대한 묘사이다. (4)의 알고리즘을 수행하고 난 뒤에는 그림 3.3.8과 같이 타이밍 윈도우의 겹치는 부분을 최소화해 준다.

만약 라이브러리에 이러한 게이트 (3)과 (4)를 만족하는 delay를 가진 Gi가 존재하지 않을 경우 step2에서 역시 회피를 고려하지 않는다.

FW pulling의 경우 FW를 당기다 보면 앞 노드의 타이밍 윈도우와 겹칠 수 있다. 이는 또다른 누설전류를 발생시키는 요인이 된다. 이 경우를 회피하기 위해서 앞의 노드와 타이밍 윈도우가 겹치는 경우는 step2: FW pulling의 대상에서 제외한다.

다. 전체적인 알고리즘

그림 5의 알고리즘 흐름도에서 묘사됐듯이 정적 시간 분석기(Static Timing Analysis)에 의해 시간(delay) 정보를 가진 넷리스트(netlist), 빅팀 어그레서 정보, 그리고 라이브러리를 입력으로 받는다. 주어진 입력으로 임계경로를 탐색하면서 step1: LW pushing, gate downsizing, step2: FW pulling, gate upsizing으로 gate sizing을 하며 사이징이 이루어 질 때마다 전체 지연시간 정보를 갱신하며 제거된 어그레서를 회로에 반영한다. 그리고 누설전류 회피 알고리즘에 의해서 게이트 사이징된 넷리스트를 출력한다.

Ⅳ. 실험 결과

제안된 알고리즘은 Ultra Sparc workstation상에서 C 언어로 구현되었으며 LGSynth91 벤치마크 회로를 사용해 검증되었다. 회로들의 초기 매핑은 SIS[1]를 이용하였고 실제공정에서 사용되는 0.5㎛ 표준 셀 라이브러리를 목표로 하였다. 실험에서 사용된 라이브러리는 각각 5종류의 사이즈를 가지는 AND, OR, NAND, NOR, BUFFER, INVERTER로 구성되어 게이트의 INERTIAL DELAY는 게이트의 전파 지연과 같은 크기를 갖는 것으로 가정하였다. 표 1에 실험결과를 제시하였다. 표 1의 '# of aggressor by critical path' 항목은 입력으로 제공되는 회로의 임계경로에 인접한 어그레서의 개수이며 '# of avoided aggressor' 항목은 논문에서 제공되는 알고리즘을 적용한 프로그램을 거친 뒤 제거된 어그레서의 개수이다. 그리고 'Aggressor avoidance

표 1. Crosstalk Avoidance by Gate Sizing 실험결과
Table 1. Experimental result of Crosstalk Avoidance by Gate Sizing

Circuit Name	#of gates	# of Nodes	# of aggressor by critical path	# of avoided aggressor	Aggressor avoidance ratio (%)
comp	212	244	12	1	8.3
C432	222	258	16	2	12.5
C880	378	438	24	2	8.3
C499	602	643	28	3	10.7
C1908	672	705	39	3	7.7
rot	796	931	45	3	6.7
C1355	978	1019	61	6	9.8
C3540	1321	1371	65	4	6.1
pair	2160	2333	103	8	7.7
Average					8.64

ratio'는 제거된 어그레서에 대한 비율을 나타내고 있다. 제시한 Crosstalk Avoidance by gate sizing 알고리즘을 수행한 후 평균적으로 8.64%의 어그레서 감소가 있었다. 실험결과에서 나타나는 어그레서의 감소는 일반적으로 언급되는 활동하지 않는 어그레서의 제거가 아닌 실제 어그레서의 제거를 의미한다.

V. 결 론

본 논문은 임계경로의 delay에 영향을 미치는 어그레서들을 게이트 사이징을 이용한 누설전류 회피 알고리즘으로 최대한 회피시키는 방법을 제안했다. 논문의 알고리즘은 step1의 LW pushing과 step2의 FW pulling을 거치면서 게이트의 다운사이징 업사이징을 수행한다. 그 결과는 기대한 만큼 누설전류 회피 효과가 있었으며 앞으로 deepsub micron 공정이 점점 진행됨에 따라 더욱 높은 효율이 기대된다. 더군다나 이 알고리즘은 기존에 나왔던 일반적인 분석 중심의 프루닝(pruning)이 아닌 여태까지 시도하지 않았던 다른 방법인 적극적인 프루닝이 가능하다는 것을 입증한 데에 의의를 가진다. 특히 기존의 분석 접근 방법의 프루닝과는 전혀 다른 개념의 알고리즘이므로 이전 프루닝 방법과 병행 사용이 가능하다. 그리고 본 논문에서 제시된 실험으로 게이트 사이징이 글리치 제거에 의한 전력 최적화(power optimization)에만 사용되지 않고 누설전류 제거에까지 사용될 수 있음을 보여준다.

앞으로 연구할 과제는 스케줄링 알고리즘을 도입한 누설전류 회피 알고리즘을 고려하고 있으며 보다 효율적인 알고리즘이 나올 것으로 기대하고 있다.

참 고 문 헌

[1] Pinhong Chen, Kurt Keutzer, "Toward True Crosstalk Noise Analysis" on Dept. of EECS, Univ. of California at Berkeley, in 1999 IEEE

[2] Yasuhiko Sasaki, Giovanni De Micheli, "Crosstalk Delay Analysis using Relative Window Method" on stanford University Computer System Laboratory, in 1999 IEEE

[3] Tong Xiao, Malgorzata Marek-Sadowska "Worst Delay Estimation in Crosstalk Aware Static Timing Analysis" on Department of Electrical and Computer Engineering University of California, in 2000 IEEE

[4] Pinhong Chen, Yuji Kukimoto, Kurt Keutzer, "Refining Switching Window by Time Slots for Crosstalk Noise Calculation", on Dept. of EECS, U.C. Berkeley, Cadence Design System Inc, in 2002 IEEE

[5] Jae-Seok Yang, Jeong-Yeol Kim, Joon-Ho Choi, Moon-Hyun Yoo, Jeong-Taek Kong, "Elimination of false aggressors using the functional relationship for full-chip crosstalk analysis" on CAE team, Memory Division, Dept. of Device solution Network, Samsung Electronics, in 2003 IEEE

[6] Donald Chai, Alex Kondratyev, Yajun Ran, Kenneth H. Tseng, Yosinori Watanabe, Malgorzata Marek-Sadowska, "Temporo functional Crosstalk Noise Analysis", on Cadence Design System, in DAC 2003, June 2-6, Anaheim, California, USA

[7] M. Hashimoto, H. Onodera, and K. Tamaru, "A Power Optimization Method Considering Glitch Reduction by Gate Sizing", in Proceedings of the Inter national Symposium on Low Power Design, pp. 221-226, August 1998.

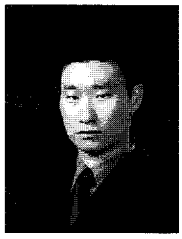
[8] J. Kim, C. Bamji, Y. Jiang, and S. Sapatnekar, "Concurrent Transistor Sizing and Buffer Insertion by Considering Cost-Delay Tradeoffs", in Proceedings of the International Symposium on Physical Design, pp. 130-135, April 1997.

[9] A.Rubio, N.Itazaki, X.Zu, and K.Kinoshita. "An Approach to the Analysis and Detection of Crosstalk Faults in Digital VLSI Circuits". IEEE Trans. on Computer-Aided Design, 13:387-394, Mar. 1994.

[10] Sachin S. Sapatnekar and Weitong Chuang, "Power vs Delay in Gate Sizing: Conflicting Objectives?", in Proceedings of the 1995 IEEE/ACM International Conference of Computer-Aided Design, pp.463-466, 1995.

[11] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A .L. Sangiovanni-Vincentelli. "SIS: A system for sequential circuit synthesis, Technical Report", UCB/ERL M92/41, Electronics Research Lab, University of California at Berkeley, 1992.

저 자 소 개



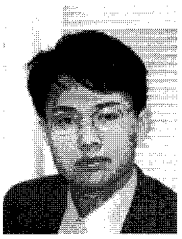
장 나 은(학생회원)

2002년 서강대학교 컴퓨터공학과 (공학사).

2004년 서강대학교 컴퓨터공학과 (공학석사).

2004년~현재 서강대학교 컴퓨터 공학과 박사과정 재학중.

<주관심분야 : Low Power and High Performance Design Methodology, Statistical Timing Analysis, Cell Characterization>



김 주 호(정회원)

· University of Minnesota at Minneapolis B.S degree in Computer and Information Science (1987).

· University of Minnesota at Minneapolis Ph.D degree in Computer and Information Science(1995).

· Senior Member of Technical staff at Cadence Design System (1995~1997).

2007년 현재 서강대학교 컴퓨터공학과 교수.

<주관심분야 : Low Power and High Performance Design Methodology, Statistical Timing Analysis, parallel spice based simulator>