JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 2 : 415~424 Feb. 2014

효율적인 ASIC구현을 위한 버퍼 삽입 방식의 HFNS

장석우[1] · 김동욱[2] · 서영호[3*]

Buffer Insertion-based HFNS for Efficient ASIC Implementation

Seok Woo Jang[1] · Dong-Wook Kim[2] · Young-Ho Seo[3*]

[1]Department of Electronic Materials Engineering, Anyang University, Kyonggi-do, 430-714, Korea
[2]Department of Electronic Materials Engineering, Kwangwoon University, Seoul 139-701, Korea
[3]College of Liberal Arts, Kwangwoon University, Seoul 139-701, Korea

요 약

본 논문에서는 HFNS(high fanout net synthesis) 기법들 중에서 실제 현장에서는 주로 사용되고 있는 버퍼 삽입 방법을 이용한 기법에 대해서 제안하였다. 먼저 HFNS를 수행하는데 있어서 고려해야할 사항들과 HFNS수행 방법들에 대해서 세부적인 기술들을 제안하였고, HFNS 수행 이후의 후처리 과정에서 수행해야할 기법들에 대해서도 제안하였다. 버퍼 삽입 기반의 HFNS는 이미 널리 사용되는 방법인데 본 논문은 주로 ASIC 및 SoC 상용 작업 현장에서 사용될 수 있는 실전적인 기법들을 대상으로 하였다.

ABSTRACT

In this paper, we proposed a practical methodology of HFNS (high fanout net synthesis) to use buffer insertion which has been applied in the fields. First, we proposed consideration to execute HFNS and detail techniques for it. Next we proposed post-process method which is necessary to obtain success of HFNS. The buffer-insertion based method for HFNS is a kind of popular technique, but we targeted a practical and commercial aspect of HHNS.

키워드 : HFNS, 버퍼 삽입, 팬아웃, 레이아웃, ASIC

Key word : HFNS, buffer insertion, fanout, layout, ASIC

접수일자 : 2013. 11. 08 심사완료일자 : 2013. 12. 04 게재확정일자 : 2013. 12. 21

*** Corresponding Author** Young-Ho Seo(E-mail:yhseo@kw.ac.kr, Tel:+82-2-940-8362)
College of Liberal Arts, Kwangwoon University, Seoul, 139-701, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.2.415 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

HFNS는 ASIC 과정에서 중요한 요소이지만 이에 대한 연구는 많지 않다. HFN(high fanout net)은 특히 리셋(reset)과 클록 게이팅(gated clock)에서 빈번히 발생하는데 오랜 합성 시간이 소요되고, 좋지 않은 네트리스트 결과를 가져온다. 이는 정적 타이밍 해석기(static timing analyzer)에 의해 만들어진 전-레이아웃 가정과 실제 후-레이아웃 회로 사이의 차이에 의해서 발생한다. 본 논문은 이러한 차이점을 분석하고 이를 극복하면서 HFN에 의한 문제를 최소화하기 위한 방법에 대해 논의하고자 한다. HFNS도 CTS와 매우 유사한 성격을 갖는다. 왜냐하면 버퍼 삽입하는 방법이 비슷하다. 본 논문에서는 CTS의 방법을 통해서 효율적인 HFNS 방법을 제안한다.

최근에 반도체 공정기술의 발달로 배선의 크기가 작아지고 이에 따른 높은 집적도를 사용한 ASIC (application specific integration circuit) 및 시스템 온 칩 (system on a chip, SoC)의 설계가 보편화되고 있다. 이러한 환경 하에서 수백 MHz 이상의 높은 성능과 신뢰성의 확보를 위한 중요한 요소가 클록(clock)의 동기화이다[1-2]. 디지털 회로는 클록 신호에 동기되어 동작하므로 클록 신호의 빠른 전달이 시스템의 성능 향상으로 직결된다. 클록 분배 네트워크(clock distribution network)라고 표현되는 클록 신호는 전체 회로 내에서 수백 수천 개의 플립플롭(flip-flop) 등으로 구성된 동기화 부하(load)들을 구동하며 칩 전체에 걸쳐서 배치된다. 클록 신호를 칩의 전체 플립플롭에 분배시키기 위해 trunk, mesh, symmetric H-tree, 그리고 버퍼 트리 (buffered tree) 등 다양한 클록 네트워크 구조가 제안되었다[3-4]. 이중에서도 버퍼 트리 방식은 칩 크기가 늘어나도 버퍼를 삽입하는 방식은 손쉽게 자동화할 수 있어 ASIC/SoC 설계에 적합하고 가장 널리 사용되는 방식이다[1].

회로의 레이아웃(layout) 시 자동으로 클록 네트워크를 구성하고 적절한 위치에 버퍼를 삽입하는 과정을 클록 트리 합성(clock tree synthesis, CTS)이라 한다. 회로의 레이아웃 수행 시 CTS는 클록 소스와 목적지 간의 버퍼 분배를 통하여 네트워크의 부하 균형(load balance)을 얻는 기술이다. CTS의 제약 조건은 소스 입력에서 특정 목적지까지의 지연시간(insertion delay), 그 값 중 가장 큰 값과 가장 작은 값의 차이 값인 클록 스큐(clock skew) 등이 있다. CTS의 품질을 높이기 위해서는 클록 구조가 복잡한 SoC 설계를 고려하기 위해 다양한 CTS 제약 조건의 설정이 필요하다. 또한 정상동작 클록(function clock)과 테스트 모드의 클록을 동시에 고려하는 경우에 흔히 발생하는 버퍼의 과도 삽입을 막아야 한다[5].

CTS 기법들과 마찬가지로 HFNS도 실제 현장에서는 버퍼 삽입 방식이 주도적으로 사용되고 있다. 그러나 이러한 HFNS 기법에 대해서 좋은 이론적인 개념이나 새로운 방식을 소개하는 연구는 많지만 실제적인 현장에서 접목할 수 있는 세부적인 기법들에 대해서 논의된 연구가 많지 않은 것이 사실이다. 본 논문의 연구팀은 지금까지 ASIC 및 SoC 설계를 수행하면서 얻은 많은 경험적이고 실제적인 HFNS 관련 기술들에 대해서 실제 현장에서 사용할 수 있는 수준으로 상세하게 제안하고 논의하고자 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 ASIC 과정에 대해서 간략히 소개를 하고, 3장에서는 HFNS에 대한 기본 이론을 설명한다. 다음으로 4장에서는 제안하고자 하는 버퍼 삽입 방식의 HFNS 기술에 대해서 논의한다. 5장에서 실험과정 및 결과를 보이고 6장에서 결론을 맺는다.

Ⅱ. ASIC 설계 과정

본 절에서는 ASIC 설계 과정을 전체적으로 이해함으로써 다음 장에서 소개될 제안한 HFNS 기법에 대한 기본적인 이해를 높이도록 한다.

2.1. RTL 단계

가장 먼저 수행되는 단계는 설계 사양을 정하는 것이다. 설계자는 회로를 설계하기 전에 먼저 설계 사양을 결정한다. 설계 사양은 회로의 기능 및 타이밍에 대한 요구조건을 명확하고 상세하게 기술하는 것이다. 설계 사양이 준비되면 설계자는 RTL(register transfer level) 코딩을 수행하고, 기능 시뮬레이션(functional simulation)을 수행하여 기능을 검증한다.

2.2. 설계 검토 및 준비

RTL 코드와 설계 사양을 바탕으로 해서 합성 및

DFT(design for test) 전략을 수립하고, 사용될 툴들을 확정한다. 이들에 필요한 스크립트(script)와 환경을 구축한다. 이를 바탕으로 pre-RTL 합성을 수행하여 RTL 코드의 문제점을 발견하고 수정한다. 다음으로 평면화(floorplan)과 빠른 배치(placement)를 수행하여 CWLM (custom wire load model)에 필요한 배선 기생요소(wire parasitic)를 생성한다. 이 과정은 모든 설계에 대해서 수행할 필요는 없다. 배선 기생요소를 이용하여 CWLM을 생성하고 앞서 결정된 DFT 방식에 따라서 메모리에 대한 BIST(built-in self test) 등의 회로를 삽입한다.

2.3. 합성

RTL 코드가 확정되면 정상 또는 1-패스 스캔 합성을 하여 네트리스트(netlist)를 추출한다. 합성 과정에 따라서 CWLM이 사용되기도 한다. 합성을 완료하면 스캔과 경계 스캔 회로를 네트리스트에 추가한다. Incremental 최적화를 통해 타이밍과 설계규칙위반 문제를 해결한다. DFT가 추가된 네트리스트는 기능의 등가 확인(functional equivalence checking)과 STA(static timing analysis)를 통해서 검증한다. 이 과정이 끝나면 초기 ATPG(automatic test pattern generation)을 통해서 테스트 벡터를 생성한다. 이를 이용해서 네트리스트의 기능 및 타이밍에 대한 시뮬레이션을 수행한다[6].

2.4. 레이아웃

합성을 통해 네트리스트가 준비되면 타이밍과 수행시간 등을 고려하여 타이밍-유도(timing-driven) 혹은 밀집-기반(congestion-based) 배치를 수행한다. 배치 이후에 CTS와 HFNS를 수행하게 된다. 이와 함께 배선을 수행하고, 이를 통해 발생하는 타이밍과 설계규칙위반의 문제를 해결해야 한다. 변경된 네트리스트는 기능적 등가 확인과 배선 기생요소 정보를 이용하여 STA를 다시 수행하여 검증 과정을 진행한다. 검증된 네트리스트에 대해서 최종적인 ATPG를 수행하여 최종 테스트 벡터를 확정한다. 레이아웃 이후에 존재하는 타이밍과 설계 규칙 위반은 후-레이아웃 최적화 과정을 통해서 해결한다. 또한 최적화로 인해 변경된 네트리스트는 기능적 등가 확인을 통해서 검증하고, 최종적인 후-레이아웃 네트리스트를 최종 테스트 벡터를 이용하여 기능 및 타이밍에 대해 시뮬레이션을 수행한다.

Ⅲ. HFNS의 기초

하나의 포트 또는 셀이 자신이 구동할 수 있는 용량을 넘어서서 너무 많은 셀을 구동한다면 신호의 에너지는 약해지며 지연시간이 증가한다. 따라서 너무 많은 팬아웃은 적절히 분배해서 중간 중간에 버퍼를 삽입해 주어야 한다. 이를 HFNS(high fanout net synthesis)라고 한다. 그림 1에서 이를 나타내었다.

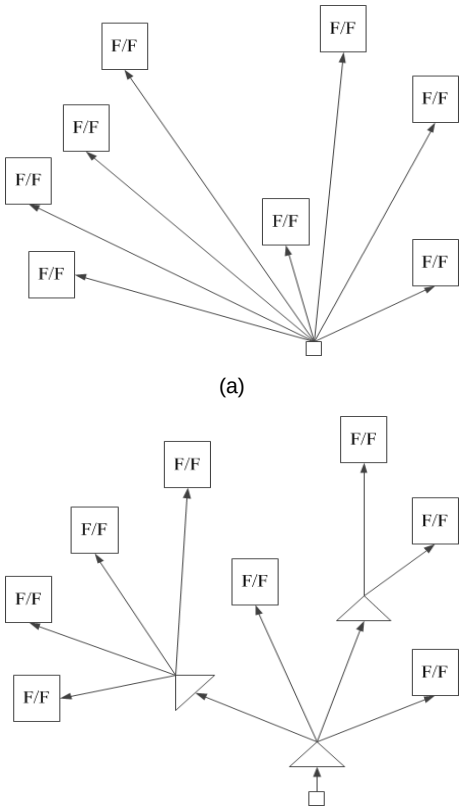


그림 1. HFNS의 적용 (a) 전, (b) 후
Fig. 1 HFNS implementation (a) before, (b) after

CTS도 일종의 HFNS이지만 클록과 리셋, test enable과 같은 네트(net)은 따로 분류해서 HFNS를 수행한다[7]. 클록 네트는 스큐를 최소화시켜야 하지만 일반 네트는 스큐에 크게 민감하지 않기 때문이다. 스큐를 맞추기 위해서는 많은 버퍼가 사용되는데 스큐를 정확하게 맞출 필요가 없는 일반 네트의 HFNS에 많은 버퍼를

사용할 필요가 없기 때문이다.

일반적으로 물리적 합성 시 설계자는 HFN의 임계치를 결정하고 물리적 컴파일러는 이 임계치 이상의 팬아웃에 대해서 HFNS를 수행한다. 이때 리셋, test enable, BIST 모드 등의 네트는 set_ideal_net, set_dont_touch로 선언하여 물리적 합성 시 HFNS을 하지 않는다. 이들은 팬아웃이 많아 물리적 컴퍼일러가 잘 처리하지 못한다. 특히 test enable 네트는 설계의 모든 플립플랍에 대해 연결되어 있으므로 팬아웃 수가 엄청나다. 이런 이유로 CTS후에 Astro를 이용하여 이들 네트만 별도로 1차로 HFNS를 수행하고 물리적 후합성 시 증가적인 HFNS를 수행한다. 이때는 이들 네트의 팬아웃 분배가 어느 정도 되어있는 상태이므로 물리적 컴파일러가 미진한 부분만 다시 최적화를 수행해준다[8].

그림 2에서 HFNS 전후의 천이지연 차이를 나타내었다. 천이지연을 고려하면 버퍼가 많이 사용되지만 이를 고려하지 않으면 많은 버퍼를 사용하지 않는다. HFN중에서도 크게 변화가 없는 BIST 모드 등의 네트는 천이시간을 고려할 필요가 없다. 천이지연를 감소시키려면 경로의 중간중간에 그림 3(b)와 같이 버퍼를 삽입해야 한다. 밀집도가 좋지 않은 설계일수록 HFNS시 주의를 기울여야 한다. 이때 HFN시 삽입되는 버퍼 수는 무시할 수 없을 정도다.

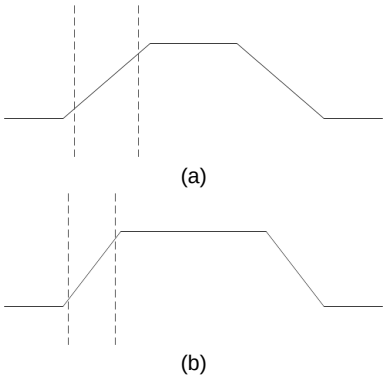


그림 2. 천이지연 시간의 차이 (a) HFNS 전, (b) 후
Fig. 2 Change of transition delay time (a) before, (b) after HFNS

Astro에서도 클록 네트에 대한 CTS 엔진과 일반 네트에 대한 HFNS 엔진이 별도로 존재한다. 물론 일반 네트에 대해 CTS엔진을 이용하여 HFNS를 수행할 수 있지만 이는 비 효율적이다. 일반 네트에 대한 HFNS 엔진도 천이시간을 고려하는 엔진과 그렇지 않은 엔진이 구분되어 있다. 그림 3에서 이들의 결과를 각각 나타내었다.

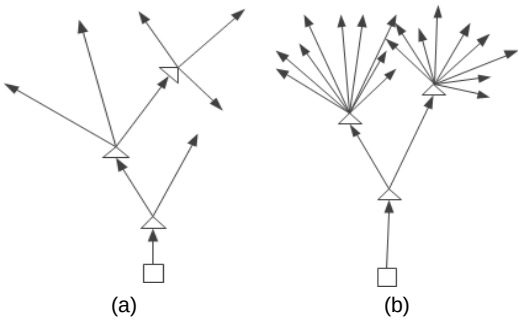


그림 3. HFNS를 위한 버퍼 삽입 (a) 천이 고려, (b) 고려하지 않은 경우
Fig. 3 Buffer insertion for HFNS in case of considering (a) transition, (b) no-trasition

Ⅳ. 효율적인 HFNS 기법

4.1. HFN 덤프(dump)

일반적으로 물리적 합성 시에 팬아웃의 임계치를 정해주면 물리적 컴파일러는 이 범위 내에서 HFNS를 수행하지만 완벽하지 않다. 실제로 임계치 이상의 팬아웃을 갖는 경우가 허다하다. 따라서 CTS전에 설계자는 반드시 일정한 팬아웃 이상을 갖는 네트를 찾아서 이들을 대상으로 HFNS를 수행해야 한다. 만약 클록 네트가 덤프된다면 CTS가 잘못된 것이다. 일반적으로 CTS는 HFNS보다 작은 팬아웃으로 수행하므로 HFN 임계치보다 큰 클록 네트가 존재한다면 반드시 확인해야 한다. 덤프를 통해 검출된 네트들은 다음 단계를 위하여 팬아웃 개수를 제외하고 네트 이름만 파일로 저장한다.

4.2. HFN 병합

일반적으로 논리적 합성 및 물리적 합성 시에 클록 네트와 HFN은 “set_dont_touch” 및 “set_ideal_net”로 선언되어 이들 경로에는 어떠한 버퍼도 삽입되어있지 않아야 한다. 만약 합성 시 삽입된 버퍼가 존재한다면 Astro에서 이들을 제거할 수 있다.

이들뿐만 아니라 앞에서 덤프받은 HFNS 대상이 되는 네트들도 경로상의 버퍼들을 모두 제거해야 깨끗한 경로상에서 HFNS을 수행한다. 그림 4에서 병합 전후를 비교하였다.

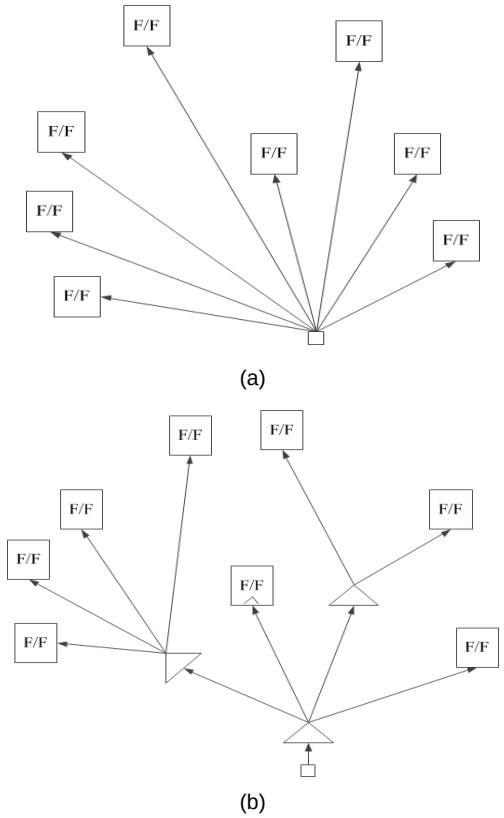


그림 4. HFN 병합 (a) 전, (b) 후
Fig. 4 Merge of HFN (a) before, (b) after

4.3. CTS엔진을 이용한 HFNS

리셋 및 test_enable 등의 네트도 CTS 옵션에서 각각 “Clocks Net”에 입력해서 “astCTS”로 HFNS를 수행할 수 있다. 그러나 이 방법은 비효율적이다. HFN을 클록과 같이 스큐를 맞추기 위해 많은 버퍼를 삽입하기 때문이다. 반면 “astHFCTS”는 스큐 및 천이을 조절하면서 HFNS를 수행할수 있으므로 가장 효율적이다. CTS 엔진을 이용하는 HFNS 시에도 CTS때와 같이 클록 공통 옵션을 설정해 주어야 한다. 그림 5에 HFNS를 위한 공통 클록 옵션을 정리하였다[9].

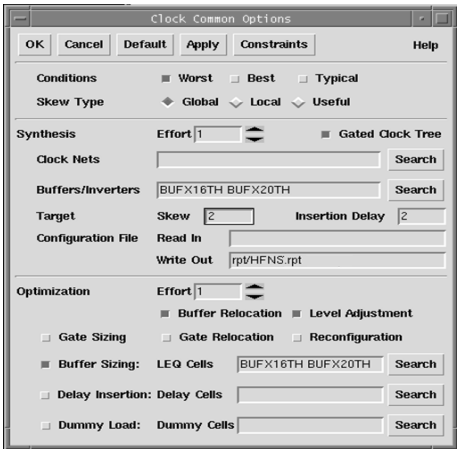
```
astClockOptions
formDefault "Clock Common Options"
formButton "Clock Common Options"
"ConstraintSubForm"
formDefault "Clock Common Options"
setFormField "Clock Common Options" "Maximum
Transition Delay" "0.8"
setFormField "Clock Common Options" "Maximum Load
Capacitance" "1.2"
setFormField "Clock Common Options" "Maximum
Fanout" “128"
setFormField "Clock Common Options" "Maximum
Buffer Level" "5"
subFormHide "Clock Common Options" 3
```

그림 5. HFNS를 위한 클록 옵션
Fig. 5 Clock option for HFNS

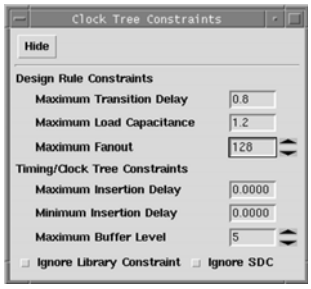
최대 버퍼 레벨 설정에서 되도록 숫자가 크지 않도록 해야 한다. 이는 핀으로부터 셀까지의 경로에 삽입되는 버퍼의 깊이이다. 일반적으로 팬아웃이 많을수록 스큐를 맞추기 위해서 버퍼 레벨이 많아지는데 HFNS시는 스큐를 크게 고려하지 않으므로 버퍼 레벨을 팬아웃에 따라서 3~7정도로 제한한다. 그림 6에 클록 옵션을 예시하였다.

CTS와 비슷하게 HFNS 버퍼는 크기가 큰 X16 및 X20만 사용한다. Astro가 많은 팬아웃의 FHNS시에 천이지연을 잘 맞추지 못하므로 STA시보다 0.1정도 낮은 값을 부여한다.

그림 7의 구성 파일 부분에 대한 설명은 CTS시의 내용과 비슷하다. Read in file은 HFNS를 수행할 때 설계자가 직접관여하기 위하여 버퍼 레벨 및 버퍼당 팬아웃 수 등을 자세하게 설정하는 부분이다. 이를 이용하면 버퍼 레벨과 각 버퍼당 팬아웃 수는 정확히 지켜서 HFNS를 수행하지만 천이지연을 고려하지 않는 단점을 갖는다. 그러므로 HFNS시에는 거의 사용하지 않는다. Read in file의 설정 예시는 그림 7과 같다.



(a)



(b)

그림 6. 클록 옵션의 예시
Fig. 6 Example of clock option

```
begin_clock_tree 4                ;; ①
clock_net TEST_SE                 ;; ②
buffer_level BUFX20TH 1           ;; ③
buffer_level BUFX20TH 30          ;; ④
buffer_level BUFX20TH 100         ;; ⑤
buffer_level BUFX20TH 1000        ;; ⑥
end_clock_tree
```

그림 7. HFNS 구성 파일 예시 (TEST_SE : 90000 fanout)
Fig. 7 Example of HFNS configuration file (TEST_SE : 90000 fanout)

그림 7은 아래와 같이 해석될 수 있다.

① HFNS buffer level=4
② HFHN
③ level 1 : insertion buffer 1
④ level 2 : insertion buffer 29(30-1)
⑤ level 3 : insertion buffer 69(100-30-1)
⑥ level 4 : insertion buffer 931(1000-100-30-1)

위에서 언급한 삽입 버퍼 갯수는 각 레벨의 누적된 개수 이므로 주의해야 한다. 이를 근거로 HFNS된 결과는 그림 8과 같다. 실제로 설계자가 직접 구성해서 HFNS을 수행하는 것은 대단히 어렵다. 최종 레벨에서의 팬아웃은 어느 정도 맞출 수 있으나 이는 대부분 최대 천이 위반을 유발할 수 있을 것이다[10].

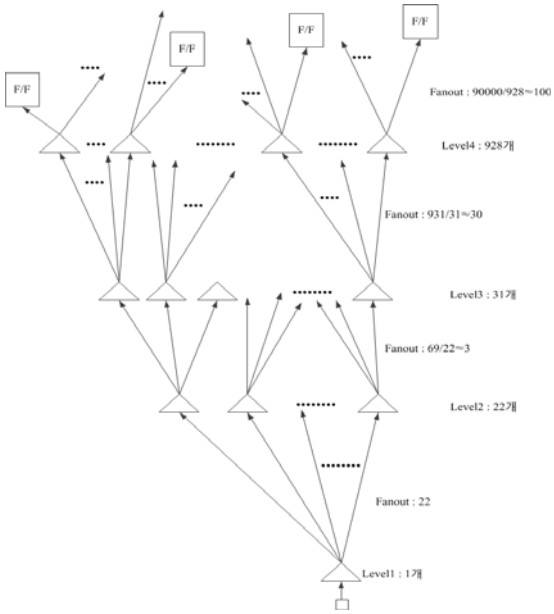


그림 8. 구성에 의한 HFNS
Fig. 8 HFNS by configuration

앞에서 클록 공통 옵션에서 천이 및 캐패시턴스 등을 제약사항으로 주었다. 이는 반드시 지켜야 하는 임계치라고 할 수 있다. 그러나 Astro는 이렇게 입력해준 제약사항을 만족시키지 못한다. 따라서 따로 제약사항를 주어야 효과적으로 HFNS을 수행할 수 있다. 이러한 파라미터는 절대적인 제약사항이 아니다. 지금처럼 제약사항을 보완하는 의미에서 파라미터를 입력해준다. Astro를 이용해서 P&R을 진행할 때 단계별로 입력해주는 이러한 파라미터는 몇 가지가 있다. Synopsys에서는 이

러한 파라미터를 실험적으로 적절히 사용할 것을 추천한다. 그림 9에는 본 논문에서 사용한 파라미터를 예시하고 있다.

```
(axSetRealParam "acts" "target: transition delay rise" 0.8)
(axSetRealParam "acts" "target: worst transition delay rise" 0.8)
(axSetRealParam "acts" "target: best transition delay rise" 0.8)
(axSetRealParam "acts" "target: transition delay fall" 0.8)
(axSetRealParam "acts" "target: worst transition delay fall" 0.8)
(axSetRealParam "acts" "target: best transition delay fall" 0.8)
(axSetRealParam "acts" "target: load capacitance" 0.7)
(axSetIntParam "acts" "CTA" 0)
(axSetIntParam "acts" "target fanout" 128)
```

그림 9. 파라미터 셋팅
Fig. 9 Parameter setting

4.4. CTS엔진을 이용하지 않는 HFNS

일반적으로 “astHFCTS”로 HFNS를 수행한다. 이는 CTS시와 비슷하게 HFNS 대상의 네트에 대해서 스큐 및 천이지연 등을 고려하기 위함이다. 그러나 실제로 0 또는 1로 고착되어 거의 변화가 없어서 천이지연 등을 고려할 필요가 없는 네트에 대해서는 굳이 “astHFCTS”로 HFNS를 수행할 필요가 없다. 이는 삽입되는 버퍼의 수가 증가하기 때문이다. 이때는 단순히 팬아웃의 개수만으로 HFNS를 수행하는 것이 효율적이다. 그림 10에 HFNS를 위해 본 논문에서 사용한 실험적인 클록 옵션을 나타냈다.

4.5. HFNS 이후의 처리

CTS와 마찬가지로 HFNS후에도 몇 가지 확인해야 할 사항이 있다. 일반적으로 HFNS 대상의 네트들은 타이밍 여유가 충분하고, 팬아웃도 많아서 그림 11(a) 같이 첫 번째 버퍼가 핀으로부터 멀리 삽입된다[11][12]. 이는 그림 11(b)와 같이 버퍼를 그 자리에 새로 삽입한 후 핀 가까이로 옮겨놓아야 한다. 한편 그림 11(c)와 같이 핀으로부터 2개 이상의 팬아웃을 갖는 경우가 있다.

이는 그림 11(d)와 같이 핀 앞에 버퍼를 새로 삽입해서 핀에서의 팬아웃은 1개가 되도록 하는 것이 좋다.

```
define HFN_CTS "list/HFN.ls“
astFanoutSetup
setFormField "FanoutSetup" "FanoutLimit" "128"
setFormField "FanoutSetup" "Collapse" "1"
setFormField "FanoutSetup" "CollapseLimit" "128"
setFormField "FanoutSetup" "From File" HFN_CTS
setFormField "FanoutSetup" "Timing Driven" "1"
formOK "FanoutSetup"
```

그림 10. HFNS를 위한 클록 옵션
Fig. 10 Clock option for HFNS

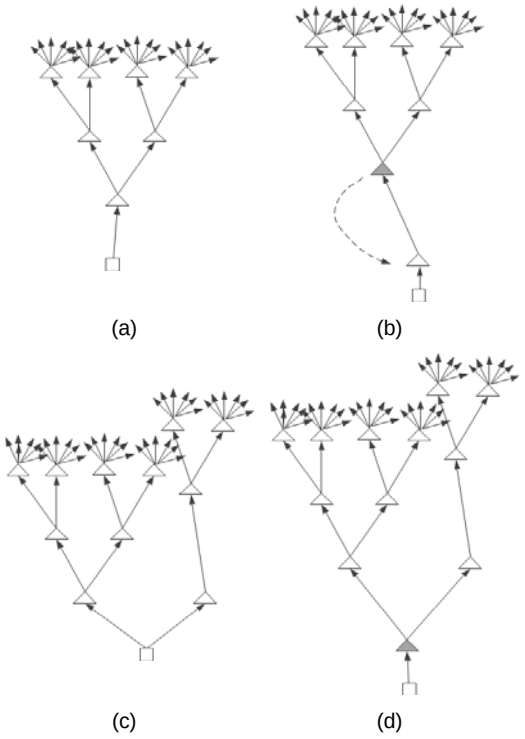


그림 11. HFNS 이후의 과정 (a) 멀리 떨어진 버퍼, (b) 버퍼 재이동 및 삽입, (c) 팬아웃=2, (d) 버퍼 삽입
Fig. 11 Post process of HFNS (a) remote buffer, (b) re-location & insertion, (c) Fanout =2, (d) buffer insertion

```
• • •
#set_ideal_network -no_propagate [get_ports $reset_port
]
#set_dont_touch_network [get_ports $reset_port ]
#set_drive 0 [get_ports $reset_port ]
• • •
define SDC_File “./const/VDP0.sdc“
ataRemoveTC
ataLoadSDC
setFormField "Load SDC File" "SDC File Name"
SDC_File
formOK "Load SDC File"
```

그림 12. 본 논문에서 사용된 .sdc
Fig. 12 .sdc file used in this paper

Ⅴ. 실험 결과

5.1. 실험 환경

이미 소개된 바와 같이 HFNS의 적용 및 실험을 위해 본 논문에서는 Astro를 사용하였다. 이때 표준 셀과 매크로 등의 Astro 라이브러리를 사용하고, 배치가 완료된 Astro의 라이브러리를 사용하였다. 물리적 구현 시 기준이 되는 제약사항는 .sdc이다.

HFNS의 경우와 같이 Astro에도 제약사항를 로드시켜야 한다. 그러나 CTS시 로드해준 .sdc와는 약간의 차이가 있다. CTS에서처럼 HFNS 대상이 되는 리셋, test enable, BIST mode등의 네트들에 적용된 "set_ideal_network", "set_don’t_touch", 및 "set_drive 0" 등의 속성들을 제거해야 한다. 본 논문에서 사용한 예를 그림 12에 정리하였다.

Astro library에 설계자가 인위적으로 제약사항를 줄 경우에는 타이밍 검사를 해야 한다. 이는 Synopsys사에서 권고하는 사항이고, 좋은 HFNS를 위해서는 필수적이라고 할 수 있다. 그림 13에 본 논문에서 사용된 타이밍 검사 방법을 정리하였다.

.sdc를 로드시킨 후에 제대로 제약사항가 적용되었는지 확인해야하다. 사용한 툴의 버전에 따라서 지원하지 않는 속성이 있을 수 있기 때문이다.

```
atTimingProbe
atTimingProbeGoto "Timing Analyzer"
atCmdRunUpdateTimer
atTimingProbeHide
```

그림 13. 타이밍 검사 스크립트
Fig. 13 Script of timing probe

5.2. HFNS 결과

본 논문에서는 많은 팬아웃을 갖는 256비트 AES 블록 암호화 알고리즘을 하드웨어로 구현한 후에 제안한 HFNS 기법을 적용하여 실험을 수행하였다. 그림 14에 HFNS 결과를 캡쳐하여 나타내었다. 그림 14는 전체 레이아웃 결과를 나타내고, 그림 15는 그림 14의 상단 부분을 확대한 결과를 나타낸다.

그림 16은 그림 14의 하단 부분을 확대한 결과를 나타낸다. 물론 HFNS 결과를 게이트 단위로 모두 확인하기는 어렵지만 그림 14에서 그림 16의 결과를 살펴보면 제안한 HFNS를 통해서 팬아웃을 적절히 분배시켰다는 것을 확인할 수 있었다. 또한 그림 17를 살펴보면 CTS 버퍼 어레이와 같이 HFNS 시에 핀 근처에 버퍼를 둘 수 있도록 F/P시에 공간을 확보해 두면 좋다는 것을 알 수 있었다.

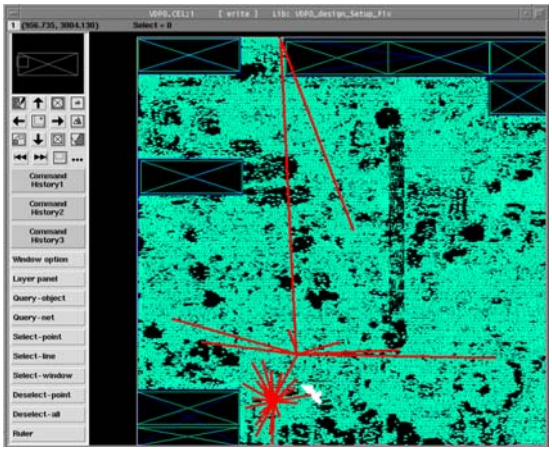


그림 14. HFNS 결과
Fig. 14 HFNS result

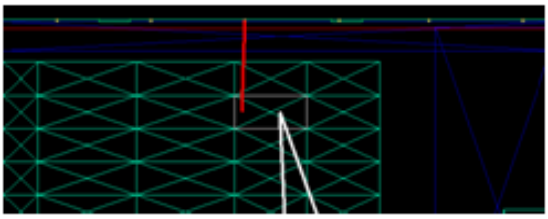


그림 15. 그림 14의 상단 부분 확대 결과
Fig. 15 Magnification of upper part of Fig. 14

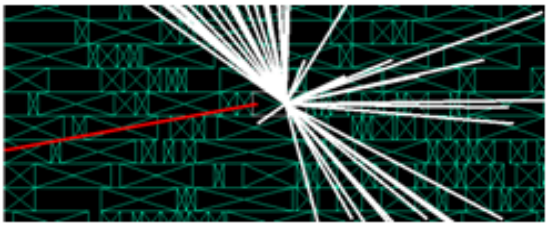


그림 16. 하단 부분 확대 결과
Fig. 16 Magnification of bottom part of Fig. 14

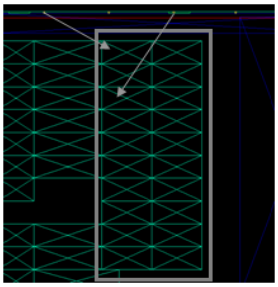


그림 17. 버퍼 어레이 결과
Fig. 17 Buffer array result

Ⅵ. 결 론

본 논문에서는 버퍼 삽입 방법에 기초한 HFNS 기법에 대해서 제안하였다. HFNS를 수행하는데 있어서 고려해야할 사항들과 HFNS 수행 방법들에 대해서 세부적인 기술들을 제안하였고, HFNS 수행 이후의 후처리 과정에서 수행해야할 기법들에 대해서도 제안하였다. 버퍼 삽입 기반의 HFNS는 이미 널리 사용되는 방법인데 본 논문은 주로 ASIC 및 SoC 상용 작업 현장에서 사용될 수 있는 실전적인 기법들을 대상으로 하였다. 그러나 이러한 HFNS 기법에 대해서 좋은 이론적인 개념이나 새로운 방식을 소개하는 연구가 매우 부족하여 실제적인 현장에서 접목할 수 있는 세부적인 기법들에 대한 논의가 거의 없었다. 본 논문의 연구팀은 지금까지 ASIC 및 SoC 설계를 수행하면서 얻은 많은 경험적이고 실제적인 HFNS 관련 기술들에 대해서 실제 현장에서 사용할 수 있는 수준으로 상세하게 설명하였다. HFNS는 사용되는 툴에 매우 의존적인데 본 논문은 Synopsys의 Astro를 대상으로 하였다. 본 논문을 통해 제안된 기법들은 많은 백앤드 설계자들에게 좋은 가이드가 되기를 희망한다.

REFERENCES

[1] Conistis, Furtner, & Islam, "Static Timing Analysis in Design Compiler and PrimeTime"; BSNUG 2000 Tutorial.

[2] J. Minz, X. Zhao, and S. K. Lim. buffered clock tree synthesis for 3d ics under thermal variations. In *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2008.

[3] T.-Y. Kim and T. Kim. Clock tree embedding for 3d ics. In *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2010.

[4] X. Zhao, D. Lewis, H.-H. S. Lee, and S. K. Lim. Pre-bond Testable Low-Power Clock Tree Design for 3D Stacked ICs. In *Proc. Int. Conf. on Computer Aided Design*, Nov 2009.

[5] F. Liu. A General Framwwork for Spatial Correlation Modeling in VLSI Design. In *Proc. Design Automation Conf.*, Jun 2007.

[6] Design Compiler User's Guide.

[7] Synthesis-749.html, "Preventing Buffers on RESET Line", Solv-Net Article.

[8] Synthesis-835.html, "Reducing Runtime Impact of High-fanout Nets in DesignCompiler", Solv-Net Article

[9] Synthesis-780.html; "Commands for high fanout nets", Solv-Net Article.

[10] John Hennessy and David Patterson, Computer Architecture : A Quantitative Approach.

[11] Rick Furtner, "High Fanout Without High Stress: Synthesis and Optimization of High-fanout Nets Using Design Compiler," SNUG Boston 2001, 2000.11.

[12] Teng, Siong Kiong, Chye, Chuan Ning, Lim, Mui Liang, Yeap, Cheong Siak, "Advanced Clock Tree Design Implementation Using IC-Compiler CTS Tool," SNUG Singapore 2009.

장석우(Seok Woo Jang)

2009년 3월 ~ 현재 안양대학교 디지털미디어학과 조교수
2000년 8월 : 숭실대학교 컴퓨터학과 (공학박사)
1997년 2월 : 숭실대학교 컴퓨터학과 (공학석사)
1995년 2월 : 숭실대학교 전자계산학과 (공학사)
※관심분야 : 로봇비전, 증강현실, HCI

김동욱(Dong-Wook Kim)

1983년 2월 : 한양대학교 전자공학과 졸업(공학사)
1985년 2월 : 한양대학교 공학석사
1991년 9월 : Georgia공과대학 전기공학과(공학박사)
1992년 3월 ~ 현재 광운대학교 전자재료공학과 정교수
2009년 3월 ~ 현재 광운대학교 실감미디어 연구소 연구소장
※관심분야 : 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP설계

서영호 (Young-Ho Seo)

1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
2001년 2월 : 광운대학교 일반대학원 졸업(공학석사)
2004년 8월 : 광운대학교 일반대학원 졸업(공학박사)
2005년 9월 ~ 2008년 2월 : 한성대학교 조교수
2008년 3월 ~ 현재 : 광운대학교 교양학부 부교수
※관심분야 : 실감미디어, 2D/3D 영상 신호처리, 디지털 홀로그램, SoC 설계