다중 클록 영역의 SoC를 위한 효율적인 버퍼삽입 방식의 CTS에 대한 고려

서영호* · 최의선** · 김동욱*

Consideration of CTS using Efficient Buffer Insertion for SoC in Multiple Clock Domain

Yong-Ho Seo* · Eui-Sun Choi** · Dong-Wook Kim*

이 논문은 2010년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2008-331-D00405).

요 약

본 논문에서는 버퍼 삽입 방법에 기반한 다중 클록 영역에서의 클록 트리 합성(clock tree synthesis, CTS) 기법에 대해서 논의한다. CTS를 수행하는데 있어서 준비해야하는 사항들과 실제적인 CTS 수행 방법들에 대해서 세부적인 기술들을 제안한다. 또한 CTS 수행 이후의 후처리 과정에 대해서도 제안한다. 버퍼 삽입 기반의 CTS는 기존에도 사용되는 방법인데 본 논문은 ASIC 및 SoC 상용 작업 현장에서 사용될 수 있는 실전적인 기법들에 대해서 논의하고자 한다. CTS는 사용되는 툴에 매우 의존적인데 본 논문은 Synopsys의 Astro를 대상으로 하였고, 이 툴을 이용하여 CTS를 수행하기 위한 세부적인 기술들에 대해서 이론을 바탕으로 경험적이고 고급적인 기법들을 제안한다. 본 논문을 통해 제안된 기법들은 많은 백앤드(backend) 설계자들에게 좋은 가이드가 될 것으로 기대한다.

ABSTRACT

In this paper, we consider a clock tree synthesis technique (CTS) based on buffer insertion method in the multiple clock domain. We propose some detail techniques about the preparing items and the practical method for implementing CTS. We also propose a post processing after CTS implementation. Until now, the buffer insertion-based CTS technique has been widely used, and this paper discusses especially it's practical technique to be applied in the commercial fields to develop ASIC and SoC. CTS is very dependent on the used tool. We use Astro of Synopsys and propose the empirical and theoretical information of the detail techniques for implementing CTS using this tool. We expect that the proposed technique becomes to be good guidelines to backend designers.

키워드

클록 트리 합성, CTS, SoC, 다중 클록 영역, 버퍼

Key word

clock tree synthesis, CTS, SoC, multiple clock domain, buffer

* 종신회원 : 광운대학교 (김동욱, dwkim@kw.ac.kr)
** 정회원 : (주)소노비젼

접수일자 : 2011. 12. 13
심사완료일자 : 2012. 02. 03

Open Access http://dx.doi.org/10.6109/jkiice.2012.16.4.643

Ⅰ. 서 론

최근에 반도체 공정기술의 발달로 배선의 크기가 작아지고 이에 따른 높은 집적도를 사용한 ASIC(application specific integration circcuit) 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.

스큐가 최소화된 버퍼 클록 트리 합성은 고성능의 VLSI(very large scaled integrated circuit) 설계에서 동기화된 회로를 위해 매우 중요한 역할을 한다. 칩 설계를 위한 타이밍 모델의 정확도가 정밀하기 않기 때문에 클록 트리 합성에 시뮬레이션 과정은 반드시 필요하다[5]. CTS를 수행하는 시간은 칩 설계의 복잡도가 증가함에 따라서 함께 증가하게 된다. 그러므로 CTS를 위한 효율적인 메카니즘을 개발하는 것이 바람직하다. 잘 설계된 클록 트리는 클록 스큐를 최소화하면서 클록을 전달하는 속도를 향상시킨다. 초기의 CTS는 선형 지연 모델 혹은 Elmore 지연 모델[6]과 같은 단순한 타이밍 모델[7-10]에 기반하여 스큐를 최소화하도록 작업이 이루어졌다. 이후에 제로-스큐의 개념[10]이 확장되었고, 동적-프로그래밍이 DME(deferred-merge embedding) 알고리즘에서 배선 길이를 줄이기 위해 채용되었다[11-13]. 반면에, 최근의 칩에서는 슬루율(slew rate)의 최적화를 위해서 클록 트리에 버퍼를 삽입하는 것을 기본으로 한다. 슬루율은 회로에서 신호의 최대 변화율을 뜻한다. 즉, 버퍼를 삽입하는 것은 슬루율과 클록 트리의 신호-위상간의 대기지연을 동시에 최적화하는데 적용되는 것이다[14-16].

대부분의 이전 연구에서 타이밍 예측의 정확도는 어떤 타이밍 모델을 사용하였는가에 달려있었다. 그러나 타이밍 모델은 스큐를 정확하게 해석하는데 어려움을 갖고 있다. 그러므로 Shin[17]은 스큐의 최소화를 수행하기 위해서 버퍼 삽입 방식의 CTS 과정에 SPICE 시뮬레이션을 내장하였다. 그러나 이러한 시뮬레이션의 속도가 보여주는 결과에 비해서 너무 느린 단점을 갖는다.

속도를 향상시킬 수 있는 방법은 구조적인 최적화에 의해 클록의 구성을 수행하는 것이다. [18]에서와 같이 매쉬 구조(mesh structure)는 본질적으로 균일한 토폴로지 때문에 작은 클록 스큐를 갖는다. 트리 구조를 비교해보면 매쉬 구조는 불가피하게 VLSI 설계에서 바람직하지 않은 높은 출력의 소스가 필요하다. 구조 최적화에 기반한 또 다른 트리 구조인 타입-정합(type-matching)은 게이트 타입을 정합하는 것으로 스큐를 최소화하였다[15]. 즉, 동일한 트리 레벨에 있는 게이트들이 유사한 지연시간을 갖는 동일한 타입이라면, 클록 스큐는 최적화될 수 있다. 그러나 최적화는 정합하는 타입을 어떻게 적용하느냐에 제한을 받게 된다.

CTS 기법들 중에서 실제 현장에서는 버퍼 삽입 방식이 주도적으로 사용되고 있다. 그러나 이러한 CTS 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 지금까

지 ASIC 및 SoC 설계를 수행하면서 얻은 많은 경험적이고 실제적인 CTS 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, 3장에서 제안하고자 하는 버퍼 삽입 방식의 CTS 기술에 대해서 논의한다. 4장에서 실험과정 및 결과를 보이고 5장에서 결론을 맺는다.

Ⅱ. ASIC 설계 과정

본 절에서는 ASIC 설계 과정을 전체적으로 이해함으로써 다음 장에서 소개될 제안한 CTS의 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(quick 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(normal) 또는 1-패스 스캔 합성을 하여 네트리스트(netlist)를 추출한다. 합성 과정에 따라서 CWLM이 사용되기도 한다. 합성을 완료하면 스캔(scan)과 경계(boundary) 스캔 회로를 네트리스트에 추가한다. Incremental 최적화를 통해 타이밍과 설계 규칙위반 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.

2.4 레이아웃

합성을 통해 네트리스트가 준비되면 타이밍과 수행시간 등을 고려하여 타이밍-유도(timing-driven) 혹은 밀집-기반 (congestion-based) 배치를 수행한다. 배치 이후에 CTS를 수행하게 된다. 이와 함께 배선(routing)을 수행하고, 이를 통해 발생하는 타이밍과 설계규칙위반의 문제를 해결해야 한다. 변경된 네트리스트는 기능적 등가 확인과 배선 기생요소 정보를 이용하여 STA를 다시 수행하여 검증(verification) 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. CTS 기법

본 장에서는 제안하고자 하는 CTS 기법에 대해서 설명한다. 최근 ASIC 및 SoC 회로에는 다양한 클록이 사용된다. 따라서 본 논문에서는 동일 클록 영역 및 이종 클록 영역에서의 CTS를 모두 논의하고자 한다.

3.1. CTS의 필요성

CTS는 앞서 설명한 바와 같이 클록 트리 합성의 약자이다. 디지털 회로를 설계할 경우 클록에 플립플롭 연결을 상당히 많이 사용하게 된다. 클록 라인에 너무 많은 플립플롭들이 연결될 경우에 팬아웃(fanout)이 너무 커져서 문제를 야기한다. 이를 해결하는 방법으로는 CTS를 하거나 구동용량이 상당히 큰 버퍼를 클록 라인에 연결한 후에 플립플롭을 구동한다. 버퍼의 구동력은 한정이 되어 있기 때문에 보통의 경우에는 CTS를 많이 사용한다.

기본적인 CTS의 방법은 플립플롭 몇 개마다 일반 버퍼를 삽입한다. 만약에 플립플롭 5개 마다 버퍼를 삽입한다고 할 경우에, 플립플롭의 총수가 200개이고, 2개 모듈마다 100개의 플립플롭이 있다고 할 경우에, 플립플롭이 100개인 모듈에 있어서 5개의 플립플롭마다 버퍼를 삽입하면 일단 최종단의 20개의 버퍼가 만들어지고 5개의 버퍼마다 1개의 버퍼를 추가하면, 최종단 앞단은 4개의 버퍼가 된다. 그리고 그 앞단에 버퍼를 1개 더 추가한다. 그리고 플립플롭이 100개 더 있는 모듈도 같은 방식으로 버퍼를 삽입한다. 이런 식으로 버퍼를 삽입하게 되면, 모양이 삼각형 모양의 트리 모양이 된다. 그래서 클록 트리라는 이름을 사용한다.

이렇게 하는 것은 결국, 팬아웃 문제를 해결하면서, 최종단의 플립플롭에 도달하는 클록의 시간차이를 최대한 같게 하여 클록스큐를 줄이기 위한 것이다. 기본적으로 CTS는 사람이 일일이 수동으로 하는 것이 아니고 레이아웃 단계에서 툴을 이용한다. 그러나 여기에는 많은 한계가 있고, 실제로 많은 문제점을 발생한다. 따라서 레이아웃 단계에서 자동으로 CTS를 수행하는 과정에서부터 섬세하게 관여를 해야 하고, CTS 결과에 대해서도 설계자가 많은 관여를 하여 보정하는 과정을 수행하야만 한다.

3.2. 동일 클록 영역에서의 CTS

일반적으로 설계에는 많은 클록 영역(domain)이 존재한다. 또한 각각의 클록 영역에 많은 플립플롭과 매크로(macro) 회로가 존재한다. 동일 클록 영역의 플립플롭과 매크로 회로는 시간적으로 동일한 시점에서 트리거 되어야 한다. 그림 1을 이용하여 이러한 이유를 설명한다. 최악의 조건(worst case)시 트리거 시점이 1주기의 차이가 나면 동작에 오류가 발생한다. 따라서 동일 클록 영역에 있는 플립플롭과 매크로 회로의 트리거 시점을 맞춰주어야 하는데, 이를 위해서는 클록 핀에서부터 각각의 플립플롭까지 소요되는 지연시간을 동일하게 조절해주어야 한다. 이러한 과정을 수행하는 것이 CTS이다.

그림 1에서의 A, B, 및 C 플립플롭까지 도달하는 클록 신호의 시간 차이를 그림 2에 나타내었다. 위상 지연은 동일 클록 영역에서 클록 핀으로부터 각각의 플립플롭과 매크로의 클록 핀까지의 지연시간으로 정의한다.

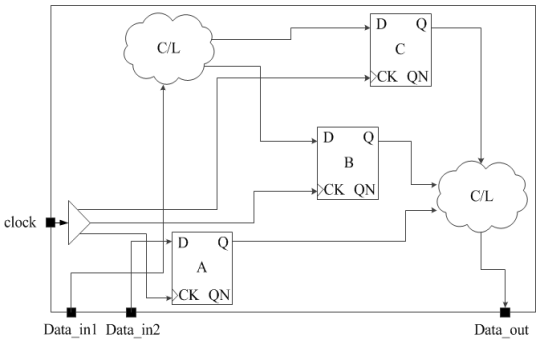


그림 1. 클록 트리의 예제
Fig. 1 Example of clock tree

A, B, 및 C 플립플롭의 각각의 위상 지연시간은 그림 2의 tA, tB, 및 tC이고 클록 스큐는 (tC - tA)이다. 여기에서 스큐가 1주기 이상 차이 나고 있다고 가정한다.

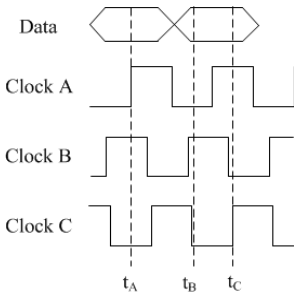


그림 2. 위상 지연
Fig. 2 phase delay

그림 2에서 클록 스큐가 1주기 이상 발생하므로 논리 동작에 오류가 발생할 것이다. 따라서 A, B, C의 스

큐를 조절하여 맞춰 주어야 한다. CTS의 방법에는 몇 가지가 있지만 그림 1에서 생각할 수 있는 방법은 다음과 같다. 첫 번째 방법은 A, B 플립플롭을 C 플립플롭과 같은 거리로 이동해서 클록 신호의 도달 시점을 조절하는 것이다. 그러나 이미 이들 주위의 표준 논리 회로들이 자리 잡고 있으므로 이들 플립플롭만 옮기게 될 경우에 물리-합성(physical-synthesis)시 고려된 타이밍과 밀집(congestion)이 모두 흐트러지게 된다. 두 번째 방법은 클록 핀 자체를 옮기는 것이다. 그러나 이 또한 수만에서 수십 만개되는 플립플롭을 모두 고려하기에는 현실적으로 쉽지 않은 방법이다. 그림 3에서 일반적인 버퍼 삽입 기반의 CTS방법을 나타내었고, 그림 4에서 그에 따른 효과를 나타내었다.

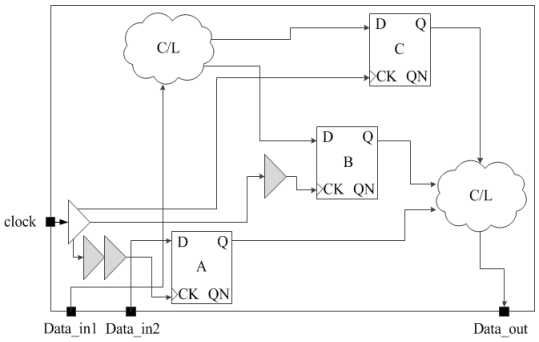


그림 3. CTS 방법
Fig. 3 CTS method

그림 3에서의 A, B, 및 C 플립플롭까지 도달하는 클록 신호의 시간 차이를 그림 4에 나타내었다. 일반적으로 가장 긴 위상 지연을 갖는 플립플롭까지의 지연을 맞추기 위해서 다른 플립플롭까지의 클록 경로에 버퍼를 삽입한다. 버퍼를 삽입함으로써 셀 지연의 영향을 받아 스큐가 감소하고 중간의 클록 천이가 좋아지는 효과도 나타난다. 그림 2와 비교할 때 그림 4의 (tC - tA)는 1주기 이하로 감소하였다. 실제 물리적 구현(physical implementation) 시 스큐를 0으로 만드는 것은 불가능하다. 따라서 최근의 0.25μm 이하의 공정에서 실제는 0.15ns 이하의 스큐는 허용하는 추세다.

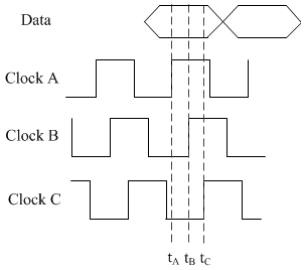


그림 4. 타이밍 다이어그램
Fig. 4 Timing diagram

3.3. 이종 클록 영역에서의 클록 트리 합성

그림 5에서 클록 A는 A 플립플롭에 영향을 미칠 뿐 B와 C 플립플롭에는 영향을 미치지 않는다. 따라서 클록 A는 클록 B 및 클록 C와 서로 false 경로이다.

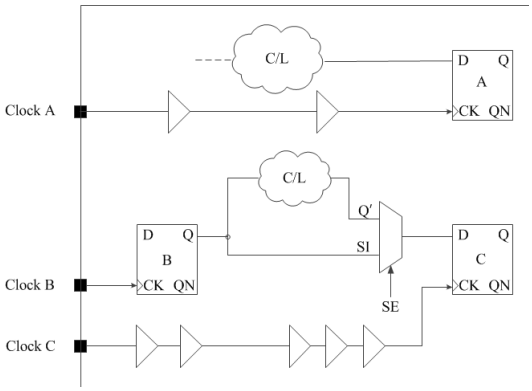


그림 5. 이종 클록 영역
Fig. 5 Different clock domain

그러나 클록 B와 클록 C 영역은 서로 다중화기(multiplexer, MUX)를 공유한다. MUX의 선택(SE) 신호에 따라서 클록 B와 클록 C는 false 경로일수도 있고 아닐 수도 있다. 우선 false 경로가 아니고 SE 신호가 클록 B에 동기 한다고 가정하자.

그림 6에서 클록 B와 클록 C 영역 간 논리 오류가 발생함을 확인할 수 있다. 이를 방지하기 위해서 록업 래치(lockup-latch)를 사용한다.

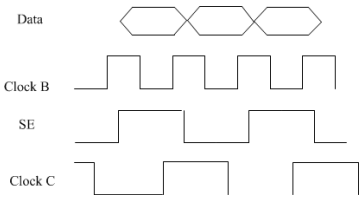


그림 6. 타이밍 다이어그램
Fig. 6 Timing diagram

그림 7에서 록업 래치를 사용함으로써 논리 보상을 수행하였다. 그러나 클록 C의 주기가 클록 B의 2배 이상이 될 경우에는 록업 래치를 한 개 더 써야 한다. 그렇지 않으면 그림 8에서와 같이 중간 데이터를 논 패치(non patch)하게 된다. 따라서 정상 데이터 경로에 록업 래치가 쓰이는 일은 전무하다. 따라서 이종 클록 영역에서의 CTS는 데이터 경로를 공유하는 클록 영역끼리만 고려해야 한다. 통상 록업 래치가 사용되었으면 최소 위상 지연은 록업 래치의 GN 핀까지의 지연이다. 따라서 CTS시 록업 래치의 GN 핀까지의 위상 지연은 무시하고 스큐를 계산한다. 스큐 계산 시 록업 래치를 포함시켜서 클록의 1주기 이상 차이가 나면 문제가 발생한다. 그러나 실제로는 클록 B와 클록 C처럼 클록의 주기가 2배 이상 되는 일이 없다.

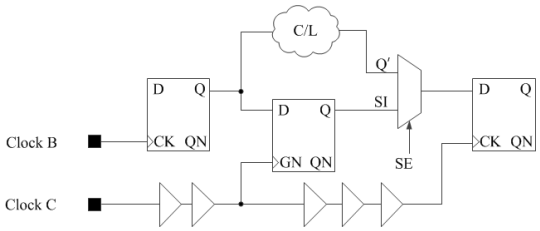


그림 7. 록업 래치를 이용한 논리 보상
Fig. 7 Logic compensation using lockup-latch

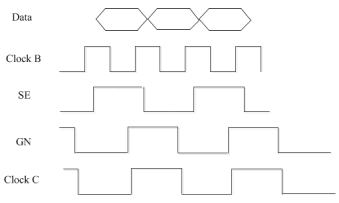


그림 8. 타이밍 다이어그램
Fig. 8 Timing diagram

3.4. 이종 클록간의 위상 지연

일반적으로 계층적 설계 시 각 블록에는 여러 개의 클록 영역이 존재한다. 이들을 최상위 블록에서 보면 그림 9와 같이 같은 클록이 분주해서 서로 다른 이름으로 각 블록에 전파되는 경우도 있고 서로 false 경로인 클록이 각 블록으로 전파되는 경우도 있다. 블록 CTS 시 동일한 클록 영역의 스큐는 0.15ns 이하로 맞추되 같은 계열의 클록 간 위상 지연은 최대한 같게 맞춰야 한다. 그림 9는 동일 클록 영역 내에서 스큐를 고정시킨 것이다.

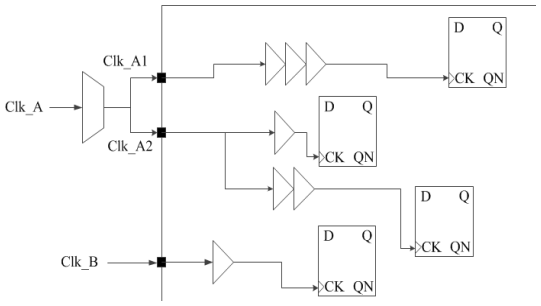


그림 9. 스큐 고정
Fig. 9 Skew fixing

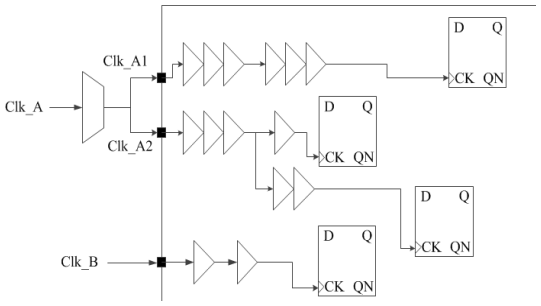
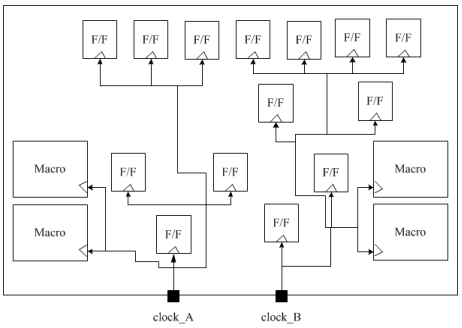


그림 10. 위상 고정
Fig. 10 Phase fixing

그림 10은 서로 다른 클록에 대한 위상을 고정시킨 것이다. clock_A1과 clock_A2는 clock_A가 파생되어 전파된 클록 들이다. 따라서 이들의 위상은 같아야 한다. clock_B가 clock_A와 false 경로라고 가정할 때 clock_B의 위상 지연은 clock_A1, clock_A2의 그것과 별개로 취급한다.

3.5. CTS 후처리

좋은 CTS 결과를 얻기 위해서는 CTS 전에 구동 셀의 클록 핀의 팬아웃 수를 반드시 확인해야 한다. 또한 CTS 후에는 구동 셀의 클록 핀 과 각 셀의 클록 핀 사이에 클록 버퍼가 삽입되어 있어야 한다. 그러므로 CTS 전처럼 구동 셀의 클록 핀의 팬아웃 수는 1개가 되어야 한다. 이러한 과정을 그림 11에 나타냈다.

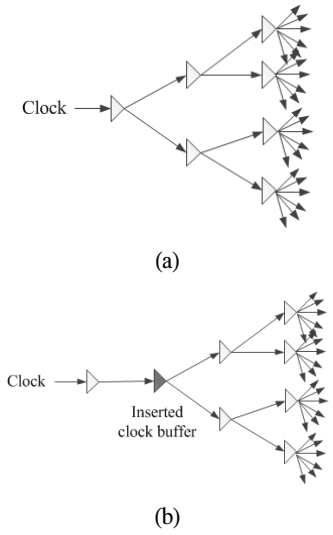


(a)

(b)

그림 11. (a) CTS 전, (b) CTS 후 팬아웃 = 1
Fig. 11 (a) before CTS, (b) after CTS, fanout=1

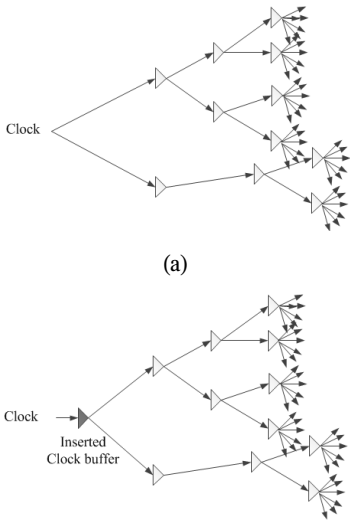
그림 12(a)와 같이 첫 번째 클록 버퍼가 구동 셀의 클록 핀에서 너무 멀리 떨어져서 CTS 되는 경우가 있다. 계층적 블록의 물리적 구현(physical implementation)시 구동 셀의 모든 입력 핀으로 유입되는 신호는 버퍼를 통해서 내부로 전파되어야 천이에 유리하다. 따라서 그림 12(b)의 경우는 멀리 떨어져 있는 버퍼의 자리에 새로운 버퍼를 삽입해서 기존의 스큐가 틀어지는 것을 막고 기존의 버퍼를 클록 핀 쪽으로 당겨서 전진 배치해야 한다.



(a)

(b)

그림 12. CTS 수행 (a) 원래의 회로(멀리 떨어진 버퍼) (b) 버퍼 위치 변경 및 삽입 이후의 회로
Fig. 12 CTS implementation (a) original logic (away from the buffer) (b) changed logic after buffer re-location and insertion



(a)

(b)

그림 13. CTS 수행 (a) 원래 회로(팬아웃=2), (b) 버퍼 삽입 이후의 회로
Fig. 13 CTS implementation (a) original logic(fanout=2) (b) changed logic after buffer insertion

또한 그림 13(a)와 같이 클록 핀이 2개 이상의 팬아웃이 되도록 CTS가 되는 경우도 있다. 이때는 앞서와 같이 첫 번째 두 버퍼 모두 멀리 떨어져 있는 경우가 대부분이다. 따라서 그림 13(b)와 같이 앞의 경우와는 달리 기존의 위치에 버퍼 삽입 없이 클록 핀 앞에 새로운 클록 버퍼를 삽입한다. 새로 삽입해 준 클록 버퍼도 CTS이므로 표시해야 한다.

Ⅳ. 구현 및 실험

본 절에서는 실제로 CTS를 수행하기 위해 먼저 준비해야 되는 실제적인 사항들에 대해서 고려하고 CTS를 수행한 이후의 후처리 방법들에 대해서 설명한다. 또한 테스트 조건하에서의 CTS 결과를 보인다.

4.1. CTS의 적용

CTS 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 constraint는 .sdc이다.

CTS전에 반드시 환경을 잘 갖추어 주어야 큰 효과를 볼 수 있었다. CTS가 잘 되지 않아 스큐가 개선이 되지 않는 클록에 대해서는 다시 클록 트리 경로의 버퍼를 지우고 다른 옵션을 주어 CTS를 다시 수행하였다. 처음에는 가장 기본적인 옵션만 주고 CTS를 수행하였다. CTS는 배치 과정 이후에 수행하였다. 배치 및 배선 등의 물리적 요소를 정확히 적용시켜서 스큐를 계산하기 위함이었다. 따라서 논리 합성 및 물리적 합성 시 클록 핀 및 네트(net)들에 대해서는 set_dont_touch 및 set_ideal_net의 속성이 적용하였다. 따라서 합성 후에는 특별한 경우를 제외하고 그 셀의 클록 핀은 구동 셀의 클록 핀에 직접 연결하였다. 따라서 합성 후에는 report_net의 클록 핀의 팬아웃 수를, 배치 후에는 구동 셀의 클록 핀의 팬아웃 수를 반드시 확인하였다.

앞서서 록업 래치의 GN 핀까지의 위상 지연은 스큐 계산에 고려하지 않아도 된다고 제안하였다. Astro는 록업 래치를 무시하지 않으면 CTS시 록업 래치의 GN 핀도 CTS 대상으로 보고 다른 플립플롭과 스큐 차이를 줄이기 위해서 노력하게 된다. 이런 헛된 노력을 하지 않기 위해서 CTS 수행 전에 모든 록업 래치의 GN 핀은 CTS 대상에서 제외시켰다.

이러한 CTS를 위한 환경설정을 수행하고 CTS를 수행하였다. CTS를 수행한 후에는 각 클록의 스큐를 먼저 관찰하였다. Astro 자체적으로 가상적 라우팅을 수행해서 스큐 값을 계산해서 보고할 수 있다. 그러나 이는 정확한 스큐 값이 아닌 것으로 확인되었다. 정확하게 스큐를 계산하려면 CTS를 배선해서 각 네트에 대한 RC 값을 추출하고 여기에 근거해서 지연 값을 계산해야 한다. 그러나 Astro는 가상적 배선 이후 자체 내장된 RC 추출 및 지연 계산 엔진을 사용하므로 실제 수치와는 다소 차이가 있었다. 그러나 그 오차가 0.07ns 이하로 크지 않고 빠르게 스큐 값을 확인한다는 점에서 의미 있는 결과라 할 수 있다.

초기 시도에서 CTS를 한 번에 맞출 수는 없었다. 스큐를 조절할 때는 설계자가 스큐가 좋지 않은 클록의 CTS 경로를 보며 직접 조절할 수도 있겠지만, 이 방법은 매우 어렵기 때문에 관련된 CTS를 전체 지워서 그 클록에 대한 CTS를 다시 수행하였다. 기존의 CTS는 CTS 표시가 되어있으므로 CTS를 다시 수행하기 위해서는 문제가 되는 클록에 대한 CTS를 un-표시 해주었다. un-표시 순서는 CTS를 표시할 때의 역순이다.

최초 CTS시에는 .sdc에 이종 클록 간 위상을 위한 constraint를 고려하지 않았다. 실험에서 팬아웃이 많은 클록이 위상 지연이 제일 컸다. 이는 스큐를 맞춰야 할 플립플롭의 개수가 많아지면서 클록 버퍼 레벨이 깊어지기 때문으로 보인다. 위상 지연이 가장 큰 클록의 위상 지연에 나머지 클록의 위상을 맞추었다. 처음부터 이 값을 알 수는 없으므로 CTS를 한번 수행한 이후에 제일 큰 위상 지연의 크기를 찾아서 이를 .sdc에 set_clock_latency 속성으로 주었다. .sdc의 내용이 바뀌었으므로 다시 .sdc를 load 한 후 타이밍을 확인하였다. set_clock_latency를 너무 작게 주면 스큐를 맞추기가 어렵다. 이종 클록 간 최대 위상 지연은 1.5ns 이하로 최대한 작게 맞추었다. 위상 지연을 위해 다른 클록에 삽입되는 클록 버퍼의 개수는 적을수록 좋기 때문으로 보인다.

CTS를 다시 해도 스큐가 좋아지지 않는 부분에 대해서는 다음과 같은 방법을 적용하였다. 먼저보다 작은 크기의 클록 버퍼를 사용하였다. 작은 클록 버퍼를 사용하면 삽입되는 클록 버퍼의 개수는 다소 증가할 수 있으나 스큐를 섬세하게 조절할 수 있는 이점이 있을 것이다. 다음으로 합성 및 최적화 노력도(effort)를 증가시켰다. 일

반적으로 노력도를 증가시키면 위상 지연은 증가하나 스큐는 개선되었다. 다음으로 최대 위상 지연을 갖는 경로에 대하여 팬아웃 한계를 낮추어 보았다. 클록 버퍼가 구동하는 셀의 개수가 감소하면 천이는 좋아지므로 최대 경로에 대한 지연이 감소함으로써 스큐가 개선되는 것을 확인할 수 있었다.

4.2. CTS 결과

그림 14와 15는 CTS 결과를 트레이(tray)한 결과이다. 각 클록 버퍼에서 스큐를 맞추기 위해서 각각의 균일한 각도로 팬아웃이 뻗어있음을 확인할 수 있다. 그림 14는 전체적인 CTS 구조를 나타내고 있고, 그림 15는 그림 14의 일부분을 확대한 것이다. 앞 절에서 설명한 것과 같이 그림 15는 팬아웃의 균일성을 확인할 수 있고, 그림 15(c)에서는 특히 삽입된 클록 버퍼의 영향을 관찰할 수 있다.

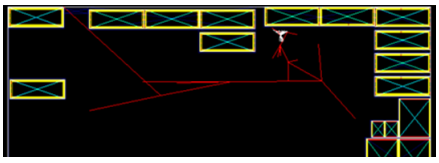
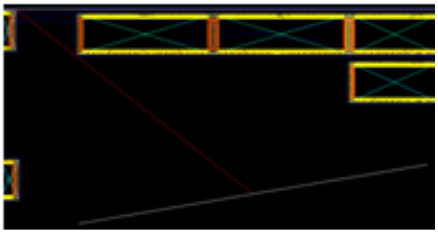
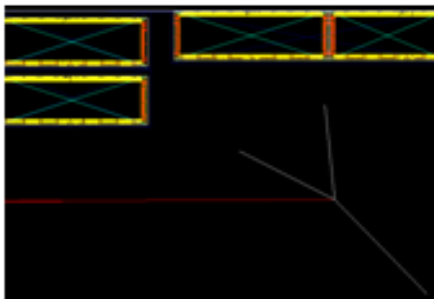


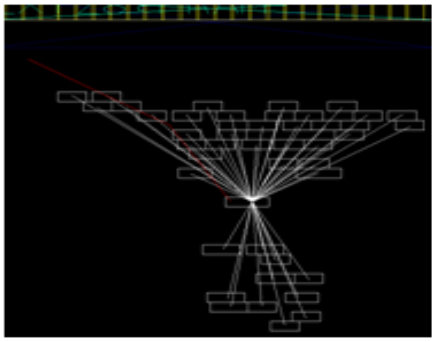
그림 14. CTS 트레이
Fig. 14 CTS tray



(a)



(b)



(c)

그림 15. 그림 14의 (a) 왼쪽 아래, (b) 오른쪽 아래, (c) 오른쪽 위 CTS 결과
Fig. 15 CTS results of Fig. 14. (a) left down, (b) right down, (c) right up

V. 결 론

본 논문에서는 CTS 기법들 중에서 실제 현장에서는 주로 사용되고 있는 버퍼 삽입 방법을 이용한 다중 클록 영역에서의 클록 트리 합성(clock tree synthesis, CTS) 기법에 대해서 제안하였다. 먼저 CTS를 수행하는데 있어서 고려해야할 사항들과 CTS 수행 방법들에 대해서 세부적인 기술들을 제안하였고, CTS 수행 이후의 후처리 과정에서 수행해야할 기법들에 대해서도 제안하였다. 버퍼 삽입 기반의 CTS는 이미 널리 사용되는 방법인데 본 논문은 주로 ASIC 및 SoC 상용 작업 현장에서 사용될 수 있는 실전적인 기법들을 대상으로 하였다. 그러나 이러한 CTS 기법에 대해서 좋은 이론적인 개념이나 새로운 방식을 소개하는 연구는 많지만 실제적인 현장에서 접목할 수 있는 세부적인 기법들에 대해서 논의된 연구가 많지 않았다. 본 논문의 연구팀은 지금까지 ASIC 및 SoC 설계를 수행하면서 얻은 많은 경험적이고 실제적인 CTS 관련 기술들에 대해서 실제 현장에서 사용할 수 있는 수준으로 상세하게 설명하였다. CTS는 사용되는 툴에 매우 의존적인데 본 논문은 Synopsys의 Astro를 대상으로 하였다. 본 논문을 통해 제안된 기법들은 많은 백앤드 설계자들에게 좋은 가이드가 되기를 희망한다.

감사의 글

이 논문은 2010년도 정부재원 (교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음 (KRF-2008-331-D00405).

참고문헌

[1] 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.

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

[3] 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.

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

[5] EE Times, April 07, 2009, http://www.eetimes.com

[6] W. C. Elmore, “The transient response of damped linear network with particular regard to wideband amplifier,” In J. Applied Physics, pp. 55 – 63, 1948.

[7] M. Edahiro, “A clustering-based optimization algorithm in zero-skew routings,” In Proc. DAC, pp. 612 – 616, 1993.

[8] M. Jackson, A. Srinivasan, and E. S. Kuh, “Clock routing for high performance ICs,” In Proc. DAC, pp. 573 – 579, 1990.

[9] A. Kahng, J. Cong, and G. Robins, “High-performance clock routing based on recursive geometric matching,” In Proc. DAC, pp. 322 – 327, 1991.

[10] R.-S. Tsay, “Exact zero skew,” In Proc. DAC, pp. 336 – 339, 1991.

[11] K. D. Boese and A. B. Kahng, “Zero-skew clock routing trees with minimum wirelength,” In Proc. ASICON, pp. 17 – 21, 1992.

[12] T.-H. Chao, Y.-C. H. Hsu, and J.-M. Ho, “Zero skew clock net routing,” In Proc. DAC, pp. 518 – 523, 1992.

[13] M. Edahiro, “Minimum skew and minimum path length routing in VLSI layout design,” NEC Research and Development, 32(4), pp. 569 – 575, 1991.

[14] R. Chaturvedi and J. Hu, “Buffered clock tree for high quality IC design,” In Proc. ISQED, pp. 381 – 386, 2004.

[15] Y. P. Chen and D. F. Wong, “An algorithm for zero-skew clock tree routing with buffer insertion,” In Proc. ED & TC, pp. 230 – 236, 1996.

[16] G. E. Tellez and M. Sarrafzadeh, “Minimal buffer insertion in clock trees with skew and slew rate constraints,” In Trans. CAD, pp. 333 – 342, 1997.

[17] X.-W. Shih, C.-C. Cheng, Y.-K. Ho, and Y.-W. Chang, “Blockage-Avoiding buffered Clock-Tree Synthesis for Clock Latency-Range and Skew Minimization,” In Proc. ASP-DAC, 2010.

[18] A. Rajaram and D. Z. Pan, “MeshWorks: an efficient framework for planning, synthesis and optimization of clock mesh networks,” In Proc. ASPDAC, pp. 250 – 257, 2008.

http://blog.naver.com/zelkobaray?Redirect=Log&logNo=10024006446

저자소개

서영호(Young-Ho Seo)

1999년 2월 광운대학교 전자재료 공학과 졸업(공학사)
2001년 2월 광운대학교 일반대학원 졸업(공학석사)
2004년 8월 광운대학교 일반대학원 졸업(공학박사)
2003년 9월 ~ 2004년 6월 한국전기연구원 연구원
2005년 9월 ~ 2008년 2월 한성대학교 조교수
2008년 3월 ~ 현재 광운대학교 교양학부 조교수
※관심분야 : 실감미디어, 2D/3D 영상 신호처리, 디지털 홀로그램

최의선(Eui-Sun Choi)

2000년 2월 광운대학교 일반대학원 졸업(공학석사)
2004년 8월 광운대학교 일반대학원 졸업(공학박사)
2004년 9월 ~ 현재 (주)소노비전 연구소장
※관심분야 : 마이크로웨이브, RFID, LTCC, 안테나, 세라믹

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 신기술 연구원
2000년 3월 ~ 2001년 12월 인티스닷컴(주) 연구원
2009년 3월 ~ 현재 광운대학교 실감미디어 연구소 연구소장
2006년 3월 ~ 현재 (사)실감미디어 산업협회 이사
※관심분야 : 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP설계, Wireless Communication