KIPS Tr. Comp. and Comm. Sys.
Vol.5, No.12 pp.439~446 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2016.5.12.439

Performance Improvement of Prediction-Based Parallel Gate-Level Timing Simulation Using Prediction Accuracy Enhancement Strategy

Seiyang Yang†

ABSTRACT

In this paper, an efficient prediction accuracy enhancement strategy is proposed for improving the performance of the prediction-based parallel event-driven gate-level timing simulation. The proposed new strategy adopts the static double prediction and the dynamic prediction for input and output values of local simulations. The double prediction utilizes another static prediction data for the secondary prediction once the first prediction fails, and the dynamic prediction tries to use the on-going simulation result accumulated dynamically during the actual parallel simulation execution as prediction data. Therefore, the communication overhead and synchronization overhead, which are the main bottleneck of parallel simulation, are maximally reduced. Throughout the proposed two prediction enhancement techniques, we have observed about 5x simulation performance improvement over the commercial parallel multi-core simulation for six test designs.

Keywords : Verification, Event-Driven Logic Simulation, Parallel Logic Simulation

예측정확도 향상 전략을 통한 예측기반 병렬 게이트수준 타이밍 시뮬레이션의 성능 개선

양 세 양†

요 약

본 논문에서는 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션의 성능 개선을 위한 효율적인 예측정확도 향상 전략을 제안한다. 제안된 기법은 병렬 이벤트구동 로컬시뮬레이션들의 입력값과 출력값에 대한 예측을 이중으로 예측할 뿐만 아니라, 특별한 상황에서는 동적으로 예측할 수 있게 한다. 이중 예측은 첫번째 예측이 틀린 경우에 두번째 정적 예측 데이터로써 새로운 예측을 시도하게 되며, 동적 예측은 실제의 병렬 시뮬레이션 실행 과정 도중에 동적으로 축적되어진 지금까지의 시뮬레이션 결과를 예측 데이터로 활용하는 것이다. 제안된 두가지의 예측정확도 향상 기법은 병렬 시뮬레이션의 성능 향상의 제약 요소인 동기 오버헤드 및 통신 오버헤드를 크게 감소시킨다. 이 두가지 중요한 예측정확도 향상 방법을 통하여 6개의 디자인들에 대한 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션이 기존 통상적 방식의 상용 병렬 멀티-코어 시뮬레이션에 비하여 약 5배의 시뮬레이션 성능이 향상됨을 확인할 수 있었다.

키워드 : 검증, 이벤트구동 로직 시뮬레이션, 병렬 로직 시뮬레이션

1. 서 론

병렬 이벤트구동 로직 시뮬레이션은 다중 코어 내지는 다중 마이크로프로세서에 다수의 이벤트구동 로직 시뮬레이터들을 병렬적으로 연동시켜서 시뮬레이션의 성능을 높이고자 하는 것으로, 이미 오래전부터 최근까지 많은 연구가 진행되어져 왔으며[1-5, 9-11], 최근에는 상용 로직 시뮬레이터들도 대부분 멀티코어를 지원하고 있어서 멀티코어를 활용한 병렬 로직 시뮬레이션이 가능하다[6, 7]. 이와 같은, 병렬 이벤트구동 로직 시뮬레이션이 가장 필요한 단계가 게이트수준 타이밍 시뮬레이션 과정인데, 이는 게이트수준 타이밍 시뮬레이션에 제일 오랜 시뮬레이션 시간이 소요되기 때문이다. 반면에, 최근의 SOC (System On Chip)들은 대부분 디자인 전체에 수많은 비동기 클럭들이 존재하고 초미세 공정을 사용함으로 인하여 이들 최근의 SOC 설계들에서 게이

※ 이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음.
† 정 회 원 : 부산대학교 정보컴퓨터공학부 교수
Manuscript Received : June 8, 2016
First Revision : August 18, 2016
Second Revision : September 20, 2016
Accepted : September 22, 2016
* Corresponding Author : Seiyang Yang(syyang@pusan.ac.kr)

트수준 타이밍 시뮬레이션의 중요성과 필요성이 다시 증대되고 있다[12].

병렬 이벤트구동 게이트수준 타이밍 시뮬레이션에서는 시뮬레이션 하고자 하는 디자인 전체를 하나의 시뮬레이터로 실행시키지 않고 디자인을 여러개의 설계객체(이를 본 논문에서는 로컬설계객체라 칭함)들로 분할(partition)한 후에 각각의 설계객체에 하나의 시뮬레이터를 할당(본 논문에서는 이와 같이 병렬 시뮬레이션에서 하나의 시뮬레이터에서 수행되는 시뮬레이션을 로컬시뮬레이션이라 칭함)하여 이들을 연동하여 병렬적으로 실행시킴으로서 하나의 시뮬레이터로 실행시키는 것에 비하여 높은 시뮬레이션 성능을 얻고자 한다. 그러나, 지금까지의 연구들과 상용화 시도는 그리 성공적이라 할 수 없는데, 이의 핵심적 원인으로는 병렬 게이트수준 타이밍 시뮬레이션에서 존재하는 동기 오버헤드와 통신 오버헤드가 매우 과도하게 존재하기 때문이다[8]. 최근에 이와 같은 과도한 동기 오버헤드와 통신 오버헤드를 줄이는 새로운 방법으로 예측기반 병렬 이벤트구동 로직 시뮬레이션 기법이 제안되었다[13]. [13]에서 제안된 병렬 시뮬레이션 기법은 올바른 예측을 통하여 동기 오버헤드와 통신 오버헤드를 완전히 제거할 수 있어서 통상적인 기존의 병렬 시뮬레이션 기법들을 통해서는 이룰 수 없는 시뮬레이션 성능 향상을 이룰 수 있지만, 이의 실제 달성 여부는 전적으로 예측의 정확도에 달려있다. 본 논문에서는 이와 같이 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션의 성능에 결정적 요소인 높은 예측정확도를 달성하기 위해 중요한 두가지 새로운 방법을 제안하고, 이의 효능을 실험을 통하여 보인다. 앞으로 본 논문의 구성은 2장에서 배경 및 관련연구를 언급하고, 본론에 해당하는 3장에서 제안한 예측정확도 향상 기법을 자세히 설명하고, 4 장에서 실험결과를 보이고, 마지막 장에서 결론을 이야기 한다.

2. 배경 및 관련 연구[13, 14]

이벤트구동 로직 시뮬레이션의 병렬화를 통한 성능 향상이 있어서 제일 큰 방해 요소가 되는 것은 병렬 이벤트구동 로직 시뮬레이션에 필수적인 동기(synchronization)와 통신(communication)에 과도한 오버헤드가 발생한다는 것인데, 이를 각각 동기 오버헤드와 통신 오버헤드라고 한다. 동기란, 병렬 이벤트구동 로직 시뮬레이션에서 모든 로컬시뮬레이션들 각각이 자체의 시뮬레이션 시간들인 로컬시뮬레이션 시간들을 가지고 있어야 하는데 이들간에 시뮬레이션 시간상에서 인과(casauality) 관계를 올바르게 유지하는 과정이다. 통신이란 동기를 맞추면서 병렬적으로 실행되는 로컬시뮬레이션들에서 시뮬레이션되는 로컬설계객체들 사이에서 일어나게 되는 이벤트들의 전달 과정이다. 지금까지의 많은 연구들에서 밝혀진 사실은 이벤트구동 로직 시뮬레이션의 병렬화를 통하여 얻을 수 있는 성능 향상의 대부분 혹은 그 이상을 병렬화를 위하여 필수적인 동기 과정과 통신 과정에서 발생되는 동기 오버헤드와 통신 오버헤드로 인한 성능 저하가 상쇄시켜 버린다는 것이다.

이벤트구동 로직 시뮬레이션에서 동기 오버헤드가 과도한 이유는 시뮬레이션 시간 상에서 미래에 발생하게되는 이벤트를 올바르게 알 수 없다는 것에서 연유한다. 동기 방식의 두가지 방법인 비관적 동기(pessimistic synchronization)와 낙관적 동기(optimistic synchronization)[1-3]는 각기 고유의 약점으로 인하여 동기 오버헤드가 커지게 되는데, 인과 관계 오류를 원천적으로 봉쇄하기 위한 너무 빈번한 동기화와 일시적으로 발생한 인과 관계 오류를 수정하기 위한 체크포인트와 롤백 오버헤드가 각각 그것이다. 뿐만 아니라, 통신 오버헤드가 과도하게 되는 이유는 일반적으로 로컬설계객체들 사이에서 대단히 많은 수의 이벤트들이 발생하기 때문이며, 이를 최소화하는 디자인을 로컬설계객체들로 최적 분할 문제 자체가 이론적으로 지극히 어려운 문제이기 때문이다.

최근의 병렬 이벤트구동 로직 시뮬레이션 연구들[5, 10]은 GPU (Graphics Processing Unit)을 활용하여 시뮬레이션의 병렬적 실행을 극대화시킨 것인데, 이를 위하여 CMB (Chandy-Misra-Bryant) 알고리즘[15]을 사용하였다. 그러나 이 경우에서도 하나의 GPU 상에서 실행되는 시뮬레이션은 로컬시뮬레이션이 되며, 일반적으로 GPU의 수는 범용 프로세서 코어수와는 달리 수천개 이상이 되는 것이 일반적이므로 수천개의 GPU를 활용한 대규모 병렬 로직 시뮬레이션에서는 수개 내지는 수십개의 프로세서 코어를 활용한 병렬 로직 시뮬레이션과는 차원이 다른 엄청한 동기 오버헤드 및 통신 오버헤드가 발행하게 될 것이다. 그런데, 병렬 이벤트구동 로직 시뮬레이션의 문제점인 과도한 동기 오버헤드 및 통신 오버헤드에 인한 병렬 시뮬레이션 성능 제약을 효과적으로 해결할 수 있는 새로운 예측기반의 병렬 이벤트구동 로직 시뮬레이션 기법이 최근 새롭게 제안되었다[9, 13]. 제안된 예측기반 병렬 이벤트구동 로직 시뮬레이션은 예상입출력이용-런 모드와 실제입출력이용-런 모드의 두 가지 실행 모드가 상황에 따라서 번갈아 가면서 다음과 같이 실행된다.

처음에 각 로컬설계객체들 각각은 로컬시뮬레이션에 저장되어 있는 예상입력을 가지고 로컬시뮬레이션을 동기 및 통신없이 독립적으로 실행시킨다. 이 경우 각 로컬설계객체의 출력에서는 시뮬레이션이 진행되면서 실제출력들이 생성되게 되는데, 이들 실제출력들은 시뮬레이션 진행 과정에서 각 로컬시뮬레이션에 저장되어 있는 예상출력들과 비교가 진행된다. 만일 이 비교에서 실제출력이 예상출력과 일치하게 되면 각 로컬시뮬레이션은 계속하여서 다른 로컬시뮬레이션들과 동기 및 통신없이 실행되어진다. 단, 이와 같이 실행되는 경우에 실제출력과 예상출력이 일치하지 않게 되는 시점에서 요구되는 올바른 인과 관계 유지를 위하여 각 로

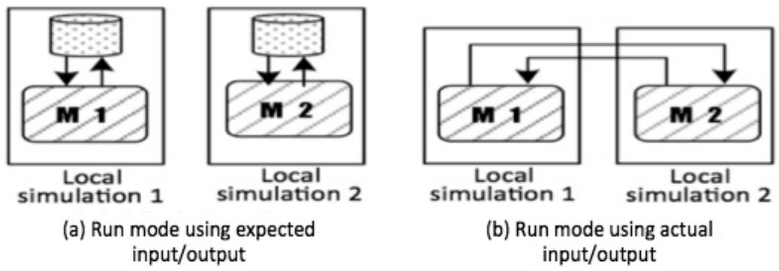

Fig. 1. Two Execution Modes of Prediction-Based Parallel Logic Simulation

컬시뮬레이션들은 체크포인트를 진행하여야 한다[13, 14]. 이와 같이 진행되는 것을 예상입출력이용-런 모드라고 하며 이 모드에서는 병렬 시뮬레이션 성능 제약을 유발케하는 동기 오버헤드 및 통신 오버헤드가 전혀 존재하지 않음으로 병렬화 정도에 비례하는 큰 폭의 시뮬레이션의 성능 향상을 기대할 수 있다. 그러나, 실제출력과 예상출력의 비교에서 이들간에 불일치가 발생하게 되면 이 불일치가 발생한 시점에서부터는 모든 로컬시뮬레이터들은 동기 및 통신 과정을 통하여 통상적인 병렬 시뮬레이션을 진행하게 되며, 이와 같이 진행되는 것을 실제입출력이용-런 모드라 한다. 따라서 실제입출력이용-런 모드로 전환된 후에는 가능한 빨리 예상입출력이용-런 모드로 재전환하는 것이 절대적으로 바람직하다. 이를 위해서 실제입출력이용-런 모드로 각 로컬시뮬레이션들이 실행되는 경우에서도 각 로컬설계객체의 출력에서 생성되는 실제출력들은 각 로컬시뮬레이션에 저장되어 있는 예상출력들과 실시간으로 비교가 진행되고, 만일 이와 같은 비교에서 실제출력과 예상출력이 일치되는 횟수가 사전에 정한 특정수와 같아지는 조건을 모든 로컬시뮬레이션들이 만족하게 되면 이 시점에서부터 다시 모든 로컬시뮬레이션들은 동기 및 통신이 필요치 않는 예상입출력이용-런 모드로 재전환되어 시뮬레이션을 진행한다. Fig. 1은 예측기반의 병렬 이벤트구동 로직 시뮬레이션의 예상입출력이용-런 모드 실행과 실제입출력이용-런 모드 실행 상황 각각을 개념적으로 설명한 그림이다. Fig. 1(a)에서 처럼 예상입출력이용-런 모드 실행에서는 각 로컬시뮬레이션이 다른 로컬시뮬레이션들과의 동기 및 통신 없이 완전 독립적으로 실행됨으로 병렬화를 통한 대폭적 성능 향상이 가능함을 잘 보여주고 있다.

3. 예측정확도 향상 기법

3.1 기존의 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션의 문제점

예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션의 핵심은 “어떤 방법을 통하여 병렬 시뮬레이션 실행 과정에서 높은 예측정확도를 계속 유지할 수 있는가?”이다. 현재 SOC 등과 같은 비메모리반도체 설계는 Verilog 또는 System Verilog와 같은 하드웨어구술언어를 사용하여 레지스터전송수준 내지는 시스템수준에서 설계하여 게이트수준을 거쳐서 레이아웃 수준까지 진행되는데 이 여러 추상화 수준 단계들에서 게이트수준 타이밍 시뮬레이션들이 반복적으로 실행되게 된다. 따라서 특정 게이트수준 타이밍 시뮬레이션 실행을 예측기반 병렬 시뮬레이션으로 실행하고자 하는 경우에 사용할 수 있는 예측데이터는 바로 이전 시뮬레이션 실행 과정에서 상기 로컬시뮬레이션들 각각에서 시뮬레이션되는 로컬설계객체들 각각의 입력정보와 출력정보를 저장한 것이다. 그런데, 이 두 시뮬레이션들(현재 시뮬레이션과 바로 이전 시뮬레이션) 사이에는 매우 높은 유사성이 존재하는 경우에는 이전 시뮬레이션 실행 과정에서 얻어진 예측데이터의 예측정확도는 매우 높을 수 있지만, 만일 이 두 시뮬레이션들 사이에는 낮은 유사성이 존재하는 경우에는 이전 시뮬레이션 실행 과정에서 얻어진 예측데이터의 예측정확도는 상당히 낮을 수 밖에는 없다. 이와 같은 상황에서, 단순하게 이전 시뮬레이션 실행 과정에서 얻어진 예측데이터를 그대로 활용한 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션에서는 빈번하게 예상출력/실제출력 불일치가 발생함으로 인하여 전체 시뮬레이션 시간 구간에서부터 높은 동기 오버헤드와 통신 오버헤드가 존재하는 실제입출력이용-런 모드로 진행되어지는 시간 구간의 증가로 인한 성능 저하뿐만 아니라, 예상출력/실제출력 불일치시점에서 제일 가까운 체크포인트로의 롤백이 빈번하게 일어나게 됨으로 인하여 시뮬레이션의 성능 향상을 기대할 수 없는 문제점이 존재한다.

3.2 제안되는 예측정확도 향상 기법

[13, 14]에서는 이와 같이 빈번한 예상출력/실제출력 불일치가 발생할 가능성이 있는 경우가 디자인에 대한 함수적 설계변경이 일어나게 되는 경우이며, 이를 여러가지 산업체 디자인들을 대상으로 실험적으로도 확인하였다. 이는 충분히 예상되는 결과인데, 즉 디자인에 대한 함수적 기능(functionality)은 그대로 유지되고 단지 국지적으로 타이밍만을 변경하는 타이밍 설계변경은 설계변경이 미치는 곳이 디자인 전체에서 국지적인 것에 비하여, 함수적 설계변경은

함수적 기능이 변경됨으로 설계변경이 디자인 전체에 걸쳐서 전반적으로 영향을 줄 수 있기 때문이다. 따라서 [13]에서의 실험에서도 나타난대로 함수적 설계변경이 이루어진 경우에서는 설계변경 이전의 시뮬레이션 실행 과정에서 얻어지는 예측데이터는 상대적으로 낮은 예측정확도를 가짐으로 인하여 설계변경 이후의 시뮬레이션을 예측기반 병렬 게이트수준 타이밍 시뮬레이션으로 수행하는 경우에는 기대와 달리 괄목할만한 시뮬레이션 성능 향상을 기대하기 어렵다. 본 논문에서는 이와 같은 타이밍 설계변경 뿐만 아니라 함수적 설계변경에 의하여 이전 시뮬레이션 실행 과정에서 얻어진 예측데이터의 예측정확도가 낮은 상황에서도 예측기반 병렬 게이트수준 타이밍 시뮬레이션을 통한 시뮬레이션 성능 향상을 달성할 수 있는 효과적인 방법들을 새롭게 제안한다.

첫 번째 방법은, 함수적 설계변경에 의하여 디자인의 함수적 기능이 변경된 경우에는 해당 함수적 설계변경 전의 게이트수준 타이밍 시뮬레이션 실행에서 얻어진 예측데이터(1차 예측데이터)와 더불어서 함수적 설계변경이 이루어진 디자인의 상위추상화 모델인 레지스터전송수준 디자인에 대한 레지스터전송수준 시뮬레이션을 추가적으로 실행하여 얻어진 예측데이터(2차 예측데이터)를 함께 사용하는 것이다. 즉, 기존의 예측기반 병렬 게이트수준 타이밍 시뮬레이션에서는 임의의 로컬시뮬레이션에서 오직 하나의 예측데이터로 예측을 실행하여 이 예측이 틀리게 되면 즉각적으로 시뮬레이션 실행 모드가 예상입출력이용-런 모드에서 실제입출력이용-런 모드로 전환되어 실행됨으로 큰 동기 오버헤드와 통신 오버헤드가 발생되는 문제점이 있는 것에 비하여, 본 논문에서 새롭게 제안되는 방식은 임의의 로컬시뮬레이션에서 1차 예측데이터로 예측을 실행하여 이 예측이 틀리게 되면 2차 예측데이터로 예측을 실행하고 이 2차 예측마져도 틀려야지만 시뮬레이션 실행 모드가 예상입출력이용-런 모드에서 실제입출력이용-런 모드로 전환되게 됨으로 실제입출력이용-런 모드에서 유발되는 과도한 동기 오버헤드와 통신 오버헤드의 문제점을 최소화시키는 것이 가능하게 된다. 물론, 설계변경 전 시뮬레이션 실행 과정에서 자연적으로 생성되는 1차 예측데이터와는 달리 2차 예측데이터를 얻기 위해서는 별도의 시뮬레이션 실행을 위한 추가적인 시뮬레이션 시간이 필요하다. 그러나, 앞서 이미 설명된 것처럼 2차 예측데이터를 얻기 위한 추가적인 시뮬레이션은 게이트수준 디자인에 대한 타이밍 시뮬레이션이 아니고 레지스터전송수준 디자인에 대한 함수적 시뮬레이션임으로 매우 빠르게 실행될 수 있다(일반적으로, 레지스터전송수준에서의 함수적 시뮬레이션은 게이트수준에서의 타이밍 시뮬레이션에 비하여 50-100배 정도 빠르다). 이와 같은 방법을 본 논문에서는 앞으로 이중 예측이라 부르기로 한다. 두번째 방법은, 예측기반의 병렬 시뮬레이션이 실행되는 과정에서 시뮬레이션 시작에서부터 현재 시뮬레이션 시간까지의 시뮬레이션 결과를 바탕으로 앞으로의 시뮬레이션 결과를 동적으로(dynamically) 예측하여 이를 예측 데이터로 사용하는 것이다 (이를 본 논문에서는 앞으로 동적 예측이라 부르기로 함). 이에 적합한 대상 시그널은 일정한 주기를 갖고 주기적으로 변화하거나, 신호변화가 없이 상수값을 가지거나, 또는 1차 예측데이터와 일정한 시간 차이로 앞당겨지거나 지연된 시그널들이다. 이와 같은 시그널들은 실제 시뮬레이션 시간의 초반에서는 동적 예측을 할 수 없지만 시뮬레이션 실행이 조금 더 진행하게 되면 어렵지 않게 파악될 수 있으며, 이 이후로부터는 올바른 예측이 가능하게 된다.

이상과 같은 예측정확도 향상 기법을 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션에 적용하기 위해서는 중요한 사항 하나를 디자인 분할 과정에서 고려하여야 한다. 왜냐하면 복수예측에 사용되는 1차 예측데이터는 설계변경 이전의 게이트수준 타이밍 시뮬레이션에서 얻어진 것

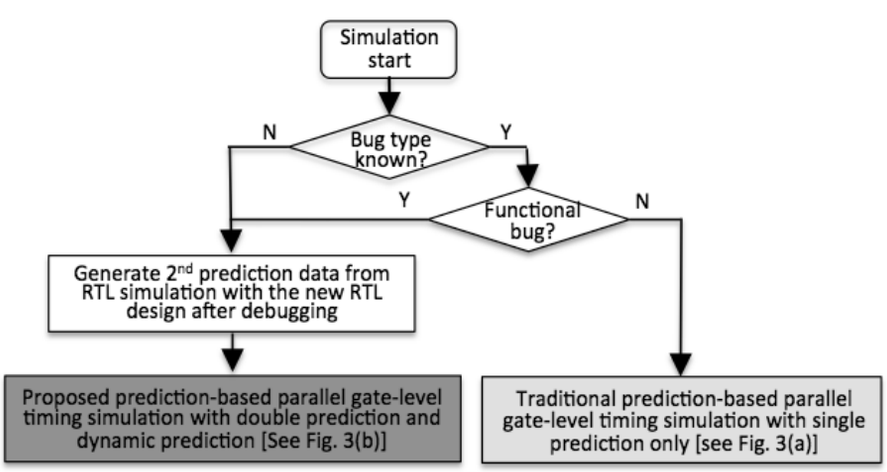

Fig. 2. Parallel Gate-Level Timing Simulation Flow Compatible With The Proposed Approach

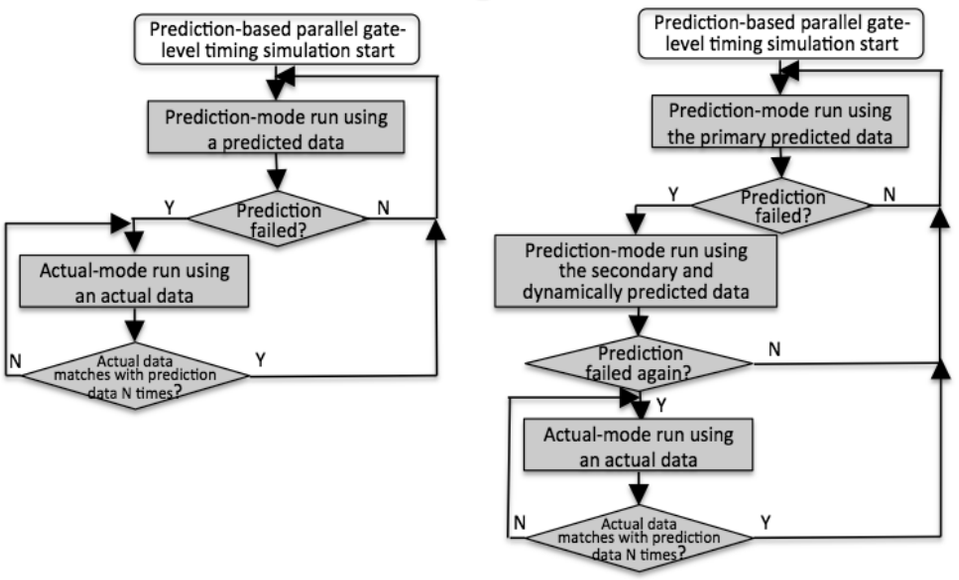

Fig. 3. Comparison of the Previous Approach and the Proposed Approach

임으로 지연시간 정보를 가지고 있지만, 2차 예측데이터는 설계변경 이후의 레지스터전송수준의 함수적 시뮬레이션에서 얻어진 것이므로 지연시간 정보를 가지고 있지 않기 때문이다. 즉, 지연시간 정보를 가지고 있지 않은 2차 예측데이터를 활용하여서 예측이 성공하기 위해서는 지연시간이 복잡한 상황을 고려하지 않아도 되도록 하는 것이 필요하다. 지연시간이 복잡한 상황은 조합회로내의 게이트들과 게이트들간의 연결선으로 다중경로 지연시간(multiple paths delay)을 통하여 발생한다. 따라서 분할 과정에서 각 로컬설계객체들의 모든 출력들은 플립플롭의 출력과 게이트를 거치지 않고 직접 연결되거나, 게이트를 거치는 경우에는 플립플롭의 출력에서 NOT 게이트나 버퍼 게이트들만을 거쳐 다중경로가 존재하지 않도록 분할하는 것이 필요하다. 최근의 SOC 디자인들은 워낙 규모가 크고 매우 많은 수의 설계블럭 또는 IP들을 사용하기 때문에 이들 일정크기 이상의 설계블럭 또는 IP들은 통상적으로 모든 출력들이 플립플롭 출력과 게이트를 거치지 않도록 직접 연결되도록 설계 가이드라인을 따르는 것이 일반적이므로 위와 같은 조건을 추가한 분할이 크게 문제가 되지는 않는다.

새롭게 제안되는 예측정확도 향상 기법을 적용한 예측기반의 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션을 디자인에 대하여 적용하는 경우에 설계변경이 디자인의 함수적 기능을 전면적으로 변화시킬 가능성이 매우 큰 함수적 변경인지, 아니면 디자인의 함수적 기능을 그대로 유지하면서 타이밍만을 변화시키는 타이밍 변경인지에 따라서 예측정확도 향상 기법을 적용할지 아닐지를 결정하게 되는데, 이를 플로우차트 형식으로 나타내면 Fig. 2와 같다.

즉, 타이밍 변경만이 이루어지는 설계변경에 대해서는 설계변경 이전의 시뮬레이션 실행 과정에서 얻어지는 예측데이터만을 활용하는 기존의 통상적인 예측기반의 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션을 실행한다. 그러나, 함수적 변경이 이루어진 경우 또는 함수적 변경과 타이밍 변경이 동시에 이루어진 경우 또는 설계변경의 성격을 잘 판달할 수 없는 경우에 대해서는 설계변경 이전의 시뮬레이션 실행 과정에서 얻어지는 예측데이터를 1차 예측데이터로 설계변경 이후의 레지스터전송수준 디자인에 대한 레지스터전송수준의 함수적 시뮬레이션 실행을 통하여 얻어지는 예측데이터를 2차 예측데이터로 사용하는 예측정확도 향상 기법을 적용한 예측기반의 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션을 실행한다.

Fig. 3은 Fig. 2의 플로우차트의 2개의 마지막 블럭들(하단의 짙은 회색과 옅은 회색 블럭들)을 구체적으로 설명하는 그림이다. Fig. 3(a)는 전통적인 기존의 예측기반 병렬 시뮬레이션의 흐름도로 오직 단일 예측 데이터를 사용하는 것이며, Fig. 3(b)는 본 논문에서 새롭게 제안된 이중 예측을 수행하는 예측기반 병렬 시뮬레이션의 흐름도이다. 두 흐름도를 비교하면 알 수 있듯이, 두 번의 예측을 통하여 이 두 번의 예측 모두가 실패하는 경우에서만 예상입출력이용-런 모드에서 동기 오버헤드 및 통신 오버헤드로 인한 시뮬레이션의 속도 저하가 초래되는 실제입출력이용-런 모드

Table 1. Experimental Result

Design name	**A** Execution time of traditional sequential logic sim (sec) [6]	**B** Traditionalmulti-coreparallellogicsim [6]		**C** Previous predicition-based parallel logic sim [9]		**D** Proposed approach	
		# of cores	Execution time (sec)	Prediction accuracy	Execution time (sec)	Prediction accuracy	Execution time (sec)
AC97	173	4	5332	45.3%	3181	95.1%	246
JPEG	5442	3	3775	0.02%	3773	100%	1777
3serial_atpg	7136	4	6410	33.0%	4956	99.5%	3529
AES	6957	2	18470	0%	18543	100%	3483
PCI	941	3	10357	26.9%	7708	100%	595
PIC-c	611	2	9339	39.9%	6356	100%	328
Total time (sec)	21260	NA	53683	NA	44517	NA	9958

로 전환되어지는 본 논문에서 제안된 방법이 단 한번의 예측이 실패하면 곧바로 실제입출력이용-런 모드로 전환하는 기존의 방법보다 효과적으로 병렬 시뮬레이션의 속도를 높일 수 있음을 알 수 있다.

4. 실험 결과

본 논문에서 제안된 예측정확도 향상 기법을 적용한 새로운 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션을 통한 시뮬레이션 성능 향상의 정도를 실험적으로 확인하기 위하여 다수의 디자인들을 대상으로 디자인에 대한 함수적 변경을 필요로 하는 디버깅을 수행한 후에 게이트수준 타이밍 시뮬레이션을 진행하는 것에 대한 실험을 진행하였다. 기존의 논문 [13]에서 이미 타이밍 변경만을 필요로 하는 디버깅을 수행한 후에 진행되는 게이트수준 타이밍 시뮬레이션을 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션으로 진행하는 경우에는 설계변경 전의 게이트수준 타이밍 시뮬레이션 실행에서 얻어진 예측데이터만으로도 높은 예측정확도를 얻는 것이 가능하고, 이를 통한 기존의 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션에서 이미 시뮬레이션의 성능 향상이 관측되었음으로 본 논문에서는 타이밍 변경만을 초래하는 설계 수정은 고려하지 않고 함수적 변경을 초래하는 설계 수정만을 고려한다.

기존의 순차 게이트수준 타이밍 로직 시뮬레이션(A)과 기존 통상적 병렬 게이트수준 타이밍 로직 시뮬레이션(B)은 순차 시뮬레이션과 병렬 시뮬레이션 모두를 지원하는 Cadence 사의 Verilog 시뮬레이터인 IUS(version 13.1)를 싱글 코어 상에서 순차 시뮬레이션과 멀티 코어(각 디자인별로 모든 비교 대상 병렬 시뮬레이션들에서 사용된 코어수는 Table 1의 3번째 컬럼 참조)상에서 병렬 시뮬레이션으로 각각 실행하여 시뮬레이션 수행시간을 측정하였다. 이 실험에서 JPEG과 3serial_atpg에서는 병렬 시뮬레이션이 순차 시뮬레이션에 비하여 각각 1.44(5442/3775)배, 1.11(7136/6410)배의 속도 향상을 얻을 수 있었으나 AC97, AES, PCI, PIC-c에서는 오히려 병렬 시뮬레이션이 순차 시뮬레이션에 비하여 각각 30.8(5332/173)배, 2.7(18470/6957)배, 11(10357/941)배, 15.3(9339/611)배의 대폭적인 속도 저하가 관찰되었다. 이 실험을 통하여 본문에서 언급된 것과 같이 통상적 병렬 게이트수준 타이밍 로직 시뮬레이션의 경우에는 로컬시뮬레이션들 사이에 발생하는 과도한 동기 오버헤드와 통신 오버헤드가 병렬 실행을 통하여 기대할 수 있는 성능 향상을 무효화시키고 오히려 순차 시뮬레이션에 비하여 속도 저하가 쉽게 발생할 수 있음을 알 수 있다. 이번에는 함수적 설계변경 이전의 시뮬레이션에서 얻어진 단일 예측데이터만을 활용하는 기존의 통상적인 예측기반 병렬 이벤트구동 게이트수준 타이밍 로직 시뮬레이션(C) 경우를 보도록 하자. 예상하는 것과 같이 설계변경이 함수적 변경임으로 설계변경 이전의 게이트수준 타이밍 시뮬레이션 실행에서 얻어진 단일 예측데이터의 예측정확도는 0~45.3%로 높지 않아서, JPEG과 3serial_atpg를 제외하고는 순차 로직 시뮬레이션(A)과 비교하여 보면, 기존 방식의 예측기반 병렬 시뮬레이션(C)도 실익이 없음을 알 수 있다(이벤트구동 로직 시뮬레이션의 특성을 반영하여 예측정확도는 전체 시뮬레이션 시간구간에서의 예측이 맞는 이벤트들 총 수를 발생하는 모든 이벤트들의 총 수로 나눈 백분율로 나타내었다).

본 논문에서 제안된 예측정확도 향상 기법(D)을 통하여 모든 디자인들에서 예측정확도(5번째 컬럼과 대비하여 7번째 컬럼)가 큰 폭으로 상승했음을 알 수 있다. 시뮬레이션 수행시간 역시 대폭적으로 개선(6번째 컬럼과 대비하여 8번째 컬럼)되었으며, AC97을 제외하고 모든 5개의 나머지 디자인들에서 기존 통상적 병렬 로직 시뮬레이션(B)과 기존 예측기반 병렬 로직 시뮬레이션(C) 뿐만 아니라, 기존 순차 로직 시뮬레이션(A)과 비교해도 성능 향상(JPEG: 3배, 3

serial_atpg: 2배, AES: 2배, PCI: 1.6배, PIC-c: 1.9배)이 관찰되어 제안된 방법의 실체적 유용성이 확인된다. 단, AC97의 경우에는 예측정확도가 큰 폭으로 향상되었지만 향상된 예측정확도가 95.1%임으로 전체 시뮬레이션 시간에서 4.9% 시간 구간에서는 로컬시뮬레이션들 사이에 동기 오버헤드와 통신 오버헤드를 발생시키는 실제입출력이용런-모드로 진행하여야 하는데, 여기에 아직도 많은 시간이 소요됨(높은 동기 오버헤드와 통신 오버헤드로 인하여)으로 인하여, 기존 통상적 병렬 로직 시뮬레이션(B)과 기존 예측기반 병렬 로직 시뮬레이션(C)에 비해서는 대폭적 시뮬레이션 수행 시간 단축을 보이고 있지만 기존의 순차 로직 시뮬레이션(A)에 비교해서는 수행시간이 42% 더 걸린다. 또한, 제안된 방법을 통하여 예측정확도가 100%로 향상된 경우가 4개의 사례에서 나왔다. 본 논문의 예측정확도 향상 기법을 통하여 예측정확도가 최대치인 100%까지 도달한 경우에는 레지스터전송수준 시뮬레이션의 함수적 행태와 게이트수준 시뮬레이션의 함수적 행태가 완전히 일치한 상황이다. 그런데, 이들 4개의 사례들 중에서 유독 PCI는 사용된 코어수에 선형적인 성능향상을 나타내지 못하고 있다. 이의 원인은 PCI에서 3개의 로컬시뮬레이션의 부하(load) 측면에서 특정 로컬시뮬레이션에 편중되게 분할된 것이 원인이다. 이와 같은 문제점들까지도 해결 또는 완화시키는 것은 추후 연구 과제이다.

5. 결 론

본 논문에서는 예측정확도 향상 기법을 적용한 예측기반의 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션을 제안하였다. 제안된 기법은 함수적 설계변경을 통하여 디자인의 함수적 기능이 바뀌어져 설계변경 이전의 시뮬레이션에서 획득된 예측데이터의 낮은 예측정확도로 인하여 시뮬레이션 성능 향상이 크제 제약을 받는 상황에 대처할 수 있다. 즉, 설계변경 이후의 레지스터전송수준 디자인을 활용한 함수적 시뮬레이션을 매우 빠르게 실행하여 2차 예측데이터를 생성한 후, 두개의 예측데이터를 활용하여 두번의 예측을 하는 이중 예측을 진행하고, 병렬 시뮬레이션을 실행하는 과정 중에서 동적예측도 병행 함으로서 시뮬레이션 실행 과정에서 동기 오버헤드와 통신 오버헤드가 유발됨으로 시뮬레이션 속도가 크게 떨어지는 실제입출력이용-런 모드의 실행을 최대한 억제함으로서 전체 병렬 시뮬레이션의 성능을 기존의 예측기반 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션과 비교하여 일괄적으로 향상 시키는 것이 가능하다. 제안된 새로운 병렬 이벤트구동 게이트수준 타이밍 시뮬레이션 방법은 산업체에서 설계된 대규모 디자인을 포함한 다수의 디자인들을 대상으로한 실험을 통해 병렬 시뮬레이션의 성능 향상에 효과적임을 확인하였다.

References

[1] R. M. Fujimoto, “Parallel Discrete Event Simulation,” *Communication of the ACM*, Vol.33, No.10, pp.30-53, Oct., 1990.

[2] D. M. Nicol, “Principles of Conservative Parallel Simulation,” *Proceedings of the 28th Winter Simulation Conference*, pp.128-135, 1996.

[3] R. M. Fujimoto, “Time Warp on a Shared Memory Multiprocessor,” *Transactions of the Society for Computer Simulation*, Vol.6, No.3, pp.211-239, Jul., 1989.

[4] L. Li and C. Tropper, “A design-driven partitioning algorithm for distributed Verilog simulation,” in *Proc. 20th International Workshop on Principles of Advanced and Distributed Simulation (PADS)*, pp.211-218, 2007.

[5] D. Chatterjee, A. DeOrio, and V. Bertacco, “Event-driven gate-level simulation with general purpose GPUs,” *Proc. of Design Automation Conference (DAC09)*, pp.557-562, Jun., 2009.

[6] IUS Simulator Usermanual, Cadence Design Systems [Internet], http://www.cadence.com.

[7] VCS Simulator Usermanual, Synopsys [Internet], http://www.synopsys.com.

[8] K. Chang and C. Browy, “Parallel Logic Simulation: Myth or Reality?” *Computer*, Vol.45, No.4, pp.67-73, Apr., 2012.

[9] Jaehoon Han et al, “Predictive parallel event-driven HDL simulation with a new powerful prediction strategy,” *Proc. of Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp.1-3, Mar., 2014.

[10] Yuhao Zhu, Bo Wang, and Yangdong Deng, “Massively Parallel Logic Simulation with GPUs,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol.16, No.3, pp.1-20, Jun., 2011.

[11] James Gross et al, “Multi-Level Parallelism for Time- and Cost-efficient Parallel Discrete-Event Simulation on GPUs,” *Proc. of 26th ACM/IEEE Workshop on Principles of Advanced and Distributed Simulation 2012 (PADS 2012)*, 2012.

[12] Gate-level Simulation Methodology, Whitepaper, Cadence Design Systems (www.cadence.com), 2013.

[13] Seiyang Yang, “A New Prediction-based Parallel Event-driven Logic Simulation,” *Journals of KIPS/ Computer and Communication Systems*, Vol.4, No.3, pp.85-90, 2015.

[14] Doohwan Kwak and Seiyang Yang, “Checkpoint/resimulation Overhead Minimization with Sporadic Synchronization in Prediction-based Parallel Logic Simulation,” *Journals of KIPS/Computer and Communication Systems*, Vol.4. No.5, pp.147-152, 2015.

[15] K. M. Chandy and J. Misra, “Distributed simulation: A case study in design and verification of distributed programs,” *IEEETrans.Softw. Engin. SE-5*, Vol.5, pp.440-452, 1979.

양 세 양

e-mail : syyang@pusan.ac.kr

1981년 고려대학교 전자공학과(학사)

1985년 고려대학교 컴퓨터공학과 (공학석사)

1990년 University of Massachusetts, Amherst 컴퓨터공학과(공학박사)

1991년~현 재 부산대학교 정보컴퓨터공학부 교수

관심분야 : 설계자동화, 특히SoC검증