(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0137409
(43) 공개일자 2024년09월20일

(51) 국제특허분류(Int. Cl.)
*G06F 30/3308* (2020.01) *G06F 30/323* (2020.01)
(52) CPC특허분류
*G06F 30/3308* (2020.01)
*G06F 30/323* (2020.01)
(21) 출원번호 10-2023-0030782
(22) 출원일자 2023년03월08일
심사청구일자 2023년03월08일

(71) 출원인
**인하대학교 산학협력단**
인천광역시 미추홀구 인하로 100(용현동, 인하대학교)
(72) 발명자
**한영신**
인천광역시 연수구 송도국제대로 261(송도동, 송도 더샵 센트럴시티)
(74) 대리인
**양성보**

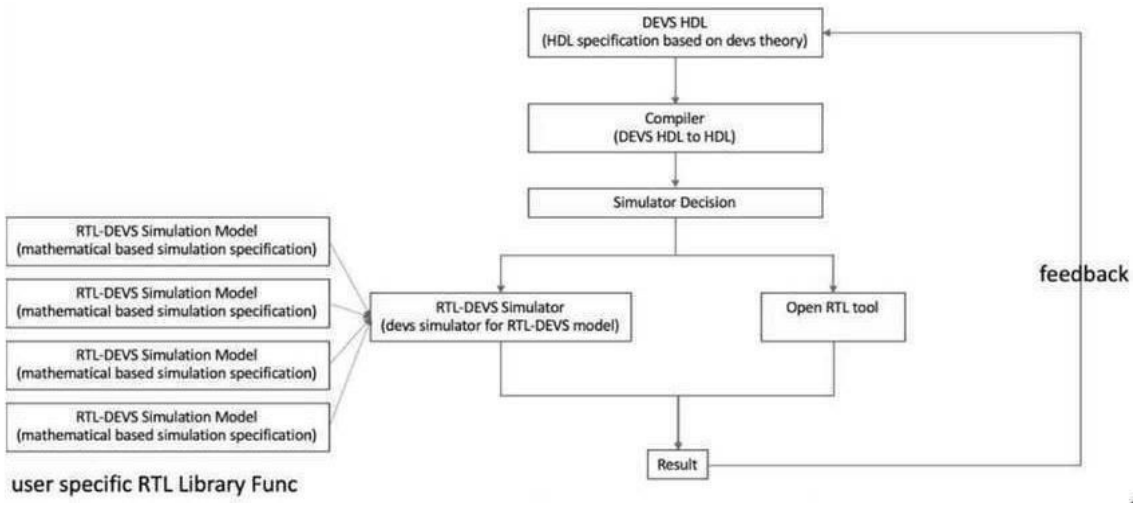
전체 청구항 수 : 총 5 항

(54) 발명의 명칭 **천이 모델에 기초한 RTL-DEVS 시뮬레이터**

## (57) 요 약

반도체 설계 공정 비용 감소를 위한 HDEVS 기반 RTL 설계 도구 및 복합 시뮬레이터가 개시된다. 컴퓨터 장치로 구현되는 RTL 시뮬레이션 시스템은 오픈소스 RTL(register transfer level) 설계 도구에 미실장된 기능이 HDEVS(hybrid discrete event system specification) 기반 RTL 설계 도구를 통해 추가됨에 따라 상기 오픈소스 RTL 설계 도구와 함께 시뮬레이션이 가능한 HDEVS 형태의 RTL-DEVS 시뮬레이터를 포함하고, 상기 RTL-DEVS 시뮬레이터는, AST(Abstract Syntax Tree)를 RTL-DEVS 타입 시스템의 구조로 변경하는 시스템 간의 천이 모델(transition model)을 포함할 수 있다.

## 대 표 도



이 발명을 지원한 국가연구개발사업

| | |
|---|---|
| 과제고유번호 | 1711169451 |
| 과제번호 | 2021R1A2C1003122 |
| 부처명 | 과학기술정보통신부 |
| 과제관리(전문)기관명 | 한국연구재단 |
| 연구사업명 | 중견연구 |
| 연구과제명 | [Ezbaro] 반도체 설계 공정 비용 감소를 위한 HDEVS기반 RTL 설계도구 및 복합 시뮬레이터개발 |
| 기 여 율 | 1/1 |
| 과제수행기관명 | 인하대학교 |
| 연구기간 | 2022.03.01 ~ 2023.02.28 |

# 명 세 서

## 청구범위

### 청구항 1

컴퓨터 장치에 있어서,

메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서

를 포함하고,

상기 적어도 하나의 프로세서는,

오픈소스 RTL(register transfer level) 설계 도구에 미실장된 기능이 HDEVS(hybrid discrete event system specification) 기반 RTL 설계 도구를 통해 추가됨에 따라 상기 오픈소스 RTL 설계 도구와 함께 시뮬레이션이 가능한 HDEVS 형태의 RTL-DEVS 시뮬레이터

를 포함하고,

상기 RTL-DEVS 시뮬레이터는,

AST(Abstract Syntax Tree)를 RTL-DEVS 타입 시스템의 구조로 변경하는 시스템 간의 천이 모델(transition model)을 포함하는 것

을 특징으로 하는 컴퓨터 장치.

### 청구항 2

제1항에 있어서,

상기 RTL-DEVS 시뮬레이터는,

HDL(Hardware Description Language) 모델을 DEVS(hybrid discrete event system specification) 모델로 변환(translation)하는 과정에서 상기 HDL의 대수적 구조를 상기 DEVS 모델에 반영하는 것

을 특징으로 하는 컴퓨터 장치.

### 청구항 3

제1항에 있어서,

상기 RTL-DEVS 시뮬레이터는,

기능(Function) 중심의 시뮬레이션 방식을 수행하는 것으로,

입력과 출력 값에 연결 계층(Connection Layer) 역할을 하는 와이어-동작 모델(Wire-Behavioral Model)을 추가하여 DEVS 모델링을 수행하는 것

을 특징으로 하는 컴퓨터 장치.

### 청구항 4

제1항에 있어서,

상기 천이 모델은 레지스터(Register) 모델과 연결성을 갖는 와이어 모델(Wire Model)을 구현한 후 결합 모델(Coupled Model)로 추상화함으로써 생성되는 것

을 특징으로 하는 컴퓨터 장치.

### 청구항 5

제1항에 있어서,

상기 적어도 하나의 프로세서는,

반정형 형태의 데이터 형태로 기술된 RTL-DEVS 모델의 동작을 HDL(Hardware Description Language)로 변환해주는 변환기; 및

상기 HDL에 대한 구문 분석 결과인 상기 AST의 노드 타입을 RTL-DEVS 타입으로 전환하는 구문 분석기(syntax analyzer)

를 더 포함하고,

상기 구문 분석기는,

타입(Type), 표현(Statement), 연산(Operation), 및 기능(Function)을 중심으로 HDL 언어인 베릴로그(Verilog)의 언어 구조를 분석하는 것

을 특징으로 하는 컴퓨터 장치.

## 발명의 설명

### 기 술 분 야

[0001] 아래의 설명은 오픈소스 RTL(register transfer level) 설계 도구를 보완할 수 있는 기술에 관한 것이다.

### 배 경 기 술

[0002] 반도체는 세계적으로 아주 큰 규모의 사업이며 반도체 웨이퍼 제작 전에 검증 과정은 오랜 시간과 돈이 들어가는 과정으로, 반도체 공정과정 중 로직 설계 및 시뮬레이션을 위해 현재 다양한 오픈소스 RTL 설계 도구가 산업계에서 활용되고 있다.

[0003] 이러한 오픈소스 RTL 설계 도구는 동작 설계 및 시뮬레이션시의 기능 제약이 많다.

[0004] 설계자가 새로운 제약조건의 동작 시뮬레이션을 수행하고자 한다면 기존의 오픈소스 RTL 설계 도구의 코드를 새롭게 개발하여 기능을 추가할 수 있다. 또 다른 방법으로, 라이선스 비용을 지불하고 해외 상용 RTL 설계 도구를 사용할 수 있다.

[0005] 하지만, 두 가지 방법 모두 대량의 비용과 많은 시간을 필요로 한다. 오픈소스를 추가로 개발하는 경우 시간적, 인적 비용이 크게 발생하게 되어 스위트 스팟을 잡기 어렵고, 상용 RTL 설계 도구를 사용하는 경우 로직 설계 단계부터 시뮬레이션까지 많은 비용이 발생한다.

[0006] 이로 인해 일부 대기업을 제외하면 중소 팹리스 업체들이 칩 설계부터 생산 공정에 도달하기까지 큰 장벽이 있으며, 프론트엔드 단계에서의 부담을 줄이는 방법들을 연구할 필요가 있다.

## 발명의 내용

### 해결하려는 과제

[0007] 반도체 설계 공정 비용 감소를 위한 HDEVS(hybrid discrete event system specification) 기반 RTL 설계 도구 및 복합 시뮬레이터를 제공할 수 있다.

[0008] HDL(Hardware Description Language) 기반의 AST(Abstract Syntax Tree)를 RTL-DEVS 타입 시스템으로 전환하는 시스템을 제공할 수 있다.

[0009] AST를 RTL-DEVS 타입 시스템의 구조로 변경하는 시스템 간의 천이 모델을 제공할 수 있다.

### 과제의 해결 수단

[0010] 컴퓨터 장치에 있어서, 메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 상기 적어도 하나의 프로세서는, 오픈소스 RTL(register transfer level) 설계 도구에 미실장된 기능이 HDEVS(hybrid discrete event system specification) 기반 RTL 설계 도구를 통해 추가됨에 따라 상기 오픈소스 RTL 설계 도구와 함께 시뮬레이션이 가능한 HDEVS 형태의 RTL-DEVS 시뮬레이터를 포함하고, 상기 RTL-DEVS 시뮬레이터는, AST(Abstract Syntax Tree)를 RTL-DEVS 타입 시스템의 구조로 변경하는 시스템 간의 천이

모델(transition model)을 포함하는 것을 특징으로 하는 컴퓨터 장치를 제공한다.

[0011] 일 측면에 따르면, 상기 RTL-DEVS 시뮬레이터는, HDL(Hardware Description Language) 모델을 DEVS(hybrid discrete event system specification) 모델로 변환(translation)하는 과정에서 상기 HDL의 대수적 구조를 상기 DEVS 모델에 반영할 수 있다.

[0012] 다른 측면에 따르면, 상기 RTL-DEVS 시뮬레이터는, 기능(Function) 중심의 시뮬레이션 방식을 수행하는 것으로, 입력과 출력 값에 연결 계층(Connection Layer) 역할을 하는 와이어-동작 모델(Wire-Behavioral Model)을 추가하여 DEVS 모델링을 수행할 수 있다.

[0013] 또 다른 측면에 따르면, 상기 천이 모델은 레지스터(Register) 모델과 연결성을 갖는 와이어 모델(Wire Model)을 구현한 후 결합 모델(Coupled Model)로 추상화함으로써 생성될 수 있다.

[0014] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 반정형 형태의 데이터 형태로 기술된 RTL-DEVS 모델의 동작을 HDL(Hardware Description Language)로 변환해주는 변환기; 및 상기 HDL에 대한 구문 분석 결과인 상기 AST의 노드 타입을 RTL-DEVS 타입으로 전환하는 구문 분석기(syntax analyzer)를 더 포함하고, 상기 구문 분석기는, 타입(Type), 표현(Statement), 연산(Operation), 및 기능(Function)을 중심으로 HDL 언어인 베릴로그(Verilog)의 언어 구조를 분석할 수 있다.

## *발명의 효과*

[0015] 본 발명의 실시예에 따르면, 오픈소스 RTL 설계 도구를 보완할 수 있는 DEVS 기반 모델을 통해 반도체 프론트엔드 설계 단계 중 RTL 설계와 HDL 변환 후 검증 단계까지의 비용이 크게 개선될 수 있다.

[0016] 본 발명의 실시예에 따르면, 백엔드 설계 과정에서 다시 로직 설계 단계로 돌아오는 상황을 최소화할 수 있고 차후 새로운 시뮬레이션 기능에 대한 확장 또한 용이하다.

[0017] 본 발명의 실시예들에 따르면, 오픈소스 RTL 설계 도구에서 부족한 기능을 DEVS 모델로 설계하여 오픈소스 RTL 설계 도구와 함께 시뮬레이션이 가능한 HDEVS 형태의 RTL-DEVS 시뮬레이터를 제공할 수 있다.

## *도면의 간단한 설명*

[0018] 도 1은 본 발명의 일실시예에 있어서 컴퓨터 장치의 내부 구성의 일례를 설명하기 위한 블록도이다.

도 2는 본 발명의 일실시예에 있어서 오픈소스 미실장 시뮬레이터 기능을 추가한 RTL-DEVS 개요를 나타낸 것이다.

도 3은 본 발명의 일실시예에 있어서 RTL 설계 및 시뮬레이션 단계를 나타낸 것이다.

도 4는 본 발명의 일실시예에 있어서 반도체 설계 공정 흐름도 및 RTL 설계 및 검증 실패가 전체 프로세스에 미치는 영향을 설명하기 위한 도면이다.

도 5는 본 발명의 일실시예에 있어서 오픈소스 시뮬레이터와의 HDEVS 복합 시뮬레이터 구상도를 나타낸 것이다.

## *발명을 실시하기 위한 구체적인 내용*

[0019] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0021] 본 발명의 실시예들은 오픈소스 RTL 설계 도구를 보완할 수 있는 기술에 관한 것이다.

[0022] 본 명세서에서 구체적으로 개시되는 것들을 포함하는 실시예들은 반도체 설계 공정 비용 감소를 위한 HDEVS 기반 RTL 설계 도구 및 복합 시뮬레이터를 제공할 수 있다.

[0023] 도 1은 본 발명의 일실시예에 따른 컴퓨터 장치의 예를 도시한 블록도이다. 예를 들어, 본 발명의 실시예들에 따른 RTL 시뮬레이션 시스템은 도 1을 통해 도시된 컴퓨터 장치(100)에 의해 구현될 수 있다.

[0024] 도 1에 도시된 바와 같이, 컴퓨터 장치(100)는 본 발명의 실시예들에 따른 RTL 시뮬레이션 방법을 실행하기 위한 구성요소로서, 메모리(110), 프로세서(120), 통신 인터페이스(130) 그리고 입출력 인터페이스(140)를 포함할 수 있다.

[0025] 메모리(110)는 컴퓨터에서 판독 가능한 기록매체로서, RAM(random access memory), ROM(read only memory) 및 디스크 드라이브와 같은 비소멸성 대용량 기록장치(permanent mass storage device)를 포함할 수 있다. 여기서 ROM과 디스크 드라이브와 같은 비소멸성 대용량 기록장치는 메모리(110)와는 구분되는 별도의 영구 저장 장치로서 컴퓨터 장치(100)에 포함될 수도 있다. 또한, 메모리(110)에는 운영체제와 적어도 하나의 프로그램 코드가 저장될 수 있다. 이러한 소프트웨어 구성요소들은 메모리(110)와는 별도의 컴퓨터에서 판독 가능한 기록매체로부터 메모리(110)로 로딩될 수 있다. 이러한 별도의 컴퓨터에서 판독 가능한 기록매체는 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록매체를 포함할 수 있다. 다른 실시예에서 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록매체가 아닌 통신 인터페이스(130)를 통해 메모리(110)에 로딩될 수도 있다. 예를 들어, 소프트웨어 구성요소들은 네트워크(160)를 통해 수신되는 파일들에 의해 설치되는 컴퓨터 프로그램에 기반하여 컴퓨터 장치(100)의 메모리(110)에 로딩될 수 있다.

[0026] 프로세서(120)는 기본적인 산술, 로직 및 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리하도록 구성될 수 있다. 명령은 메모리(110) 또는 통신 인터페이스(130)에 의해 프로세서(120)로 제공될 수 있다. 예를 들어 프로세서(120)는 메모리(110)와 같은 기록 장치에 저장된 프로그램 코드에 따라 수신되는 명령을 실행하도록 구성될 수 있다.

[0027] 통신 인터페이스(130)은 네트워크(160)를 통해 컴퓨터 장치(100)가 다른 장치와 서로 통신하기 위한 기능을 제공할 수 있다. 일례로, 컴퓨터 장치(100)의 프로세서(120)가 메모리(110)와 같은 기록 장치에 저장된 프로그램 코드에 따라 생성한 요청이나 명령, 데이터, 파일 등이 통신 인터페이스(130)의 제어에 따라 네트워크(160)를 통해 다른 장치들로 전달될 수 있다. 역으로, 다른 장치로부터의 신호나 명령, 데이터, 파일 등이 네트워크(160)를 거쳐 컴퓨터 장치(100)의 통신 인터페이스(130)를 통해 컴퓨터 장치(100)로 수신될 수 있다. 통신 인터페이스(130)를 통해 수신된 신호나 명령, 데이터 등은 프로세서(120)나 메모리(110)로 전달될 수 있고, 파일 등은 컴퓨터 장치(100)가 더 포함할 수 있는 저장 매체(상술한 영구 저장 장치)로 저장될 수 있다.

[0028] 통신 방식은 제한되지 않으며, 네트워크(160)가 포함할 수 있는 통신망(일례로, 이동통신망, 유선 인터넷, 무선 인터넷, 방송망)을 활용하는 통신 방식뿐만 아니라 기기들 간의 근거리 유선/무선 통신 역시 포함될 수 있다. 예를 들어, 네트워크(160)는, PAN(personal area network), LAN(local area network), CAN(campus area network), MAN(metropolitan area network), WAN(wide area network), BBN(broadband network), 인터넷 등의 네트워크 중 하나 이상의 임의의 네트워크를 포함할 수 있다. 또한, 네트워크(160)는 버스 네트워크, 스타 네트워크, 링 네트워크, 메쉬 네트워크, 스타-버스 네트워크, 트리 또는 계층적(hierarchical) 네트워크 등을 포함하는 네트워크 토폴로지 중 임의의 하나 이상을 포함할 수 있으나, 이에 제한되지 않는다.

[0029] 입출력 인터페이스(140)는 입출력 장치(150)와의 인터페이스를 위한 수단일 수 있다. 예를 들어, 입력 장치는 마이크, 키보드, 카메라 또는 마우스 등의 장치를, 그리고 출력 장치는 디스플레이, 스피커와 같은 장치를 포함할 수 있다. 다른 예로 입출력 인터페이스(140)는 터치스크린과 같이 입력과 출력을 위한 기능이 하나로 통합된 장치와의 인터페이스를 위한 수단일 수도 있다. 입출력 장치(150)는 컴퓨터 장치(100)와 하나의 장치로 구성될 수도 있다.

[0030] 또한, 다른 실시예들에서 컴퓨터 장치(100)는 도 1의 구성요소들보다 더 적은 혹은 더 많은 구성요소들을 포함할 수도 있다. 그러나, 대부분의 종래기술적 구성요소들을 명확하게 도시할 필요성은 없다. 예를 들어, 컴퓨터 장치(100)는 상술한 입출력 장치(150) 중 적어도 일부를 포함하도록 구현되거나 또는 트랜시버(transceiver), 카메라, 각종 센서, 데이터베이스 등과 같은 다른 구성요소들을 더 포함할 수도 있다.

[0031] 이하에서는 반도체 설계 공정 비용 감소를 위한 HDEVS 기반 RTL 설계 도구 및 복합 시뮬레이터의 구체적인 실시예를 설명하기로 한다.

[0032] DEVS(discrete event system specification) 시뮬레이션 도구로 RTL 검증 단계의 구현이 가능하다.

[0033] 본 실시예에서는 기존의 오픈소스 RTL 설계 도구에서 부족한 기능을 DEVS 모델로 설계하여 기존의 오픈소스 RTL 설계 도구와 함께 시뮬레이션이 가능한 HDEVS(hybrid DEVS) 형태의 RTL-DEVS 시뮬레이터를 제공한다.

[0034] 이에 따라 기존의 오픈소스 시뮬레이션 도구에서 제공하지 못하는 기능은 RTL-DEVS로 설계하여 시뮬레이션하고, 이를 통해 RTL의 설계 스펙을 만족하는지 검증할 수 있다.

[0035] 오픈소스 RTL 설계 도구와 RTL-DEVS 시뮬레이션 도구를 활용한 이중 검정이 가능해지며, 두 가지 시뮬레이션을 모두 만족할 때에만 합성(synthesis) 단계로 넘어갈 수 있다.

[0036] 원하는 기능을 RTL-DEVS 모델을 활용하여 빠르게 기술 가능하다. 그리고, 정의된 모델을 별도의 도구 개발없이 기존의 DEVS 시뮬레이터로 통합 시뮬레이션이 가능하다.

[0037] 이를 통하여 RTL 설계 단계에서의 라이센스 비용 및 검증 시간을 절감 가능하며, 동작의 정확도를 높일 수 있다.

[0038] 도 2는 본 발명의 일실시예에 있어서 오픈소스 미실장 시뮬레이터 기능을 추가한 RTL-DEVS 개요를 나타낸 것이다.

[0039] 도 2를 참조하면, 프로세서(120)는 반도체 디자인을 위해 DEVS 이론을 기반으로 HDL(Hardware Description Language) 요구사양(specification)을 결정한다.

[0040] RTL-DEVS 모델은 일부 기능에 대해서 기존의 오픈소스 RTL 설계 도구를 대체할 수 있고, 기존의 오픈소스 RTL 설계 도구들이 제공하지 못하는 기능에 대해서 추가적인 모델링 및 시뮬레이션을 제공하는 상호보완적인 형태로 동작할 수 있다.

[0041] 우선 기존 오픈소스 RTL 설계 도구와 상용 RTL 설계 도구들의 기능들을 비교 분석하여 RTL-DEVS에서 모델링 되어야 할 RTL 수준의 디자인 스펙을 정의한다. 기존 오픈소스 도구들의 구조분석을 통하여 시뮬레이션 동작에 대하여 시뮬레이션 요구사양을 정의한다.

[0042] RTL-DEVS에서 정의된 모델의 구현체를 기존의 오픈소스 도구를 활용하여 HDL로 재설계하고 두 모델의 시뮬레이션 결과를 분석하여 RTL-DEVS 동작 모델의 검증을 진행한다.

[0043] 프로세서(120)는 DEVS HDL에서 HDL로의 컴파일러(compiler)를 수행한다. 합성을 위해 컴파일러라는 도구를 많이 사용한다. 설계자가 반정형 형태의 데이터 형태로 RTL-DEVS Model 동작을 기술하면 HDL로 변환해주는 변환기를 사용한다. 반정형 데이터는 json, xml, yaml과 같은 데이터 형태로서 형식에 자유롭게 빠르게 기술 가능한 특성을 가진다. 설계자는 RTL-DEVS로 모델을 빠르게 정의하고 DEVS 환경에서 이를 검증할 수 있다. 그 후 이를 HDL로 변환하여 기존의 오픈소스 RTL 도구에서 등가(Equivalence) 검증을 수행한다.

[0044] 프로세서(120)는 디자인 검증을 위한 시뮬레이터를 결정한다. 검증 방법에서 가장 중요한 것은 시뮬레이션이다. HDL은 하드웨어 설명(hardware description)뿐 아니라 시뮬레이션을 위해서도 사용된다.

[0045] 본 실시예에서는 시뮬레이션 도구로서 오픈소스 RTL 도구와 함께 RTL-DEVS 모델을 위한 DEVS 시뮬레이터를 사용한다.

[0046] 일반화된 모델을 바탕으로 RTL-DEVS 동작 모델을 정의하고 RTL-DEVS 모델을 활용한 설계 도구 라이브러리를 정의한다. 기존의 오픈소스 도구들의 부재한 기능을 RTL-DEVS 모델로 재정의하고 라이브러리 구현체로 설계할 수 있다. UDP(User Defined Primitive) 사양들 및 다중 클럭 소스(Multi-Clock-Source) 도메인, 게이트 레벨(Gate level)에 대한 기능 라이브러리들을 분석하여 설계할 수 있다.

[0047] HDL 스펙 수준에서 기존 오픈소스 도구들이 제공하지 못하는 메인 기능들에 대하여 RTL-DEVS를 활용하여 라이브러리로 설계한다(예를 들어, RTL-DEVS for Gate, RTL-DEVS for UDP 등). 베릴로그 시뮬레이션(Verilog Simulation) 단계에서 적용하기 어려운 수학적 분석이 필요한 경우 이를 RUST 기반의 DEVS 기반 시뮬레이션 영역으로 가져와서 수행할 수 있다. 시뮬레이션 동작 방식에서 기존의 오픈소스 도구들이 제공하지 못하는 다양한 시뮬레이션 형태를 지원하는 라이브러리로써 설계한다(예를 들어, RTL-DEVS for MultiClock source). 이미 상용 도구로 개발되어 검증이 완료된 HDL모델을 RTL-DEVS 라이브러리로 재설계하여 시뮬레이션하고 이를 사전 검증 완료된 HDL 기반의 모델과 비교 분석하여 라이브러리의 동작을 검증한다.

[0048] 본 실시예들은 반도체 공정에서의 오픈소스 RTL 설계도구와 함께 동작하는 HDEVS 기반의 RTL 설계 도구를 제공한다. 반도체 설계 공정 중 프론트-엔드 단계 공정의 비용 이슈를 해결하기 위해 ① RTL-DEVS의 모델 스펙 설계, ② RTL-DEVS 모델의 동작 설계 및 검증, ③ RTL-DEVS를 활용한 Library의 설계 및 검증, ④ RTL-DEVS to HDL 변환기 설계, ⑤ HDEVS 시뮬레이터 개발을 통해 제한적인 오픈소스 RTL 설계도구를 고확장성, 저비용의 HDEVS 기반 RTL 설계도구와 병용하여 설계 효용을 높일 수 있다.

[0049] 현대 집적 회로 설계 작업은 ESL(Electronic system level) 설계, RTL 설계 또는 검증, 물리적 칩 설계의 3단계로 나눠진다. 설계 단계 중 RTL 설계는 조합 논리와 순차 논리를 설계하고 검증하기 위한 매우 중요한 단계이다.

[0050] 도 3은 RTL 설계 및 시뮬레이션 단계를 나타내고 있다.

[0051] RTL 설계 단계의 중요성에도 불구하고 HDL 언어 기반 설계 도구들에 대한 매우 값비싼 라이선스 비용 문제로 인해 MyHDL과 같은 다양한 오픈소스 RTL 설계 도구들이 사용되고 있다.

[0052] 반도체 설계 단계에서 오픈소스 소프트웨어의 도입은 이론상 프로토타입 개발 시간 및 비용 측면의 효율성 증대, 커뮤니티를 기반으로 수행되는 집단 검증 등 오픈소스 활용에 따른 여러 장점이 있으나 실제 상용 소프트웨어에 비해서 기능에 많은 제약이 있고 최신 기술이 지속적으로 업데이트 되기 어렵다는 단점이 있다.

[0053] RTL 단계에서의 다양한 검증 부족은 논리 설계 이후의 회로 설계와 같은 백엔드 설계 공정에서 요구사양 위반을 발생시키는 요인이 될 수 있으므로 RTL 설계 단계에서 최대한 많은 검증이 이루어져야 한다.

[0054] 그러나, 오픈소스 RTL 설계 도구에 상용 설계 도구에 준하는 기능을 추가하기 위해서는 고수준의 내부 구조분석이 필요하고, 필요로 하는 기능을 새롭게 추가하는 데에는 상용 설계 도구 사용을 위한 비용보다 훨씬 더 큰 시간적, 인적 비용이 들어갈 수 있다.

[0055] 최근 RTL 이후의 반도체 설계 공정에 HDEVS를 적용하여 비용 및 검증 구조를 개선하려는 연구들이 활발하게 진행되고 있다.

[0056] RTL 설계 작업 또한 효과적인 비용 절약을 위한 검증 구조 개선이 필요하며, HDEVS가 RTL 설계 검증을 위한 새로운 대안이 될 수 있다. 기존 오픈소스 RTL 설계도구가 제공하지 못하는 기능들을 HDEVS 기반 RTL 설계 도구를 통해 쉽게 확장할 수 있다면 상용 툴의 값비싼 라이선스 비용 지불 없이 디자인 동작 설계 수준에서 더욱 다양한 검증이 가능하다.

[0057] 따라서, 논리 합성 이후 발생하는 다양한 요구사양 위반에 의한 반복적인 재설계 빈도를 낮춤으로써 설계 효용을 높여줄 수 있다.

[0058] 오픈소스 RTL 설계도구와 함께 동작하는 HDEVS 기반의 RTL 설계 도구는 다음과 같은 세부 내용을 포함한다(도 2 참조)

[0059] 1) RTL-DEVS 모델 스펙 설계

[0060] 2) RTL-DEVS 모델 동작 설계 및 검증

[0061] 3) RTL-DEVS를 활용한 설계 도구 라이브러리의 개발

[0062] 4) RTL-DEVS to HDL 변환기의 설계

[0063] 5) HDEVS 기반의 RTL-DEVS with 오픈소스 RTL 설계도구 시뮬레이터 설계

[0064] 도 4는 반도체 설계 공정 흐름도 및 RTL 설계 및 검증 실패가 전체 프로세스에 미치는 영향을 설명하기 위한 도면이다.

[0065] 본 발명은 오픈소스 RTL 설계 도구와 상호보완으로 동작 가능한 DEVS 시뮬레이터를 제공할 수 있다. 1) 기존 설계 도구들의 특징과 DEVS 기반의 시뮬레이션 방식에 대한 비교평가를 진행한다. 2) DEVS 모델 기반 RTL을 설계하고 해당 동작이 기존 HDL 기반의 동작과 일치하는지 검증한다. 3) 시뮬레이터는 상기한 비교평가 결과와 검증 결과를 바탕으로 HDEVS 형태로 고도화한다.

[0066] 도 5는 오픈소스 시뮬레이터와의 HDEVS 복합 시뮬레이터 구상도를 나타내고 있다.

[0067] 설계자가 반정형 모델에 대한 설계 요구사항의 정의를 하고 이를 인자로 넘기면 자동으로 RTL-DEVS와 오픈소스 도구에서 검증을 해주는 통합 시뮬레이터를 구현할 수 있으며, 이를 통하여 기존의 도구들의 단점을 보완하는 HDEVS 형태의 시뮬레이션을 제공할 수 있다.

[0068] HDL 기반의 시뮬레이션을 지원하는 오픈소스 시뮬레이터들에 대한 구조 분석을 수행하고 이를 바탕으로 하여 실제 시뮬레이션에 사용되는 기능들의 다양한 스펙을 정의할 수 있다.

[0069] 예를 들어, 오픈소스 시뮬레이터인 Verilator에 대해 구조적 분석을 진행할 수 있다. 이때, 오픈소스 HDL 시뮬레이터는 빠른 속도를 바탕으로 시장 점유율을 확보하고 있으며, 이를 위해 시뮬레이션을 위한 기반 언어를 컴파일이 가능한 C++를 채택하고 있다.

[0070] 본 실시예에서는 타입 변환에 대한 안전성을 추가적으로 제공할 필요가 있으며, 이를 위해 언어 차원에서 타입

변환을 지원하는 RUST 언어를 기반으로 시스템을 설계할 수 있다.

[0071] 시뮬레이터가 HDL 로직에 대한 빠른 검증을 목표로 하고 있는 것에 반하여, 본 발명의 결과물은 다양한 로직 스펙이 추가되었을 때 해당 로직에 대한 일반화를 통하여 빠르게 기능을 확장하는 것을 목표로 한다. 이를 통하여 서로 상호 보완적인 관계로써 HDL 시뮬레이션을 대체할 수 있는 HDEVS RTL Co-Simulator의 설계를 목표로 시스템을 구성하고자 한다.

[0072] 오픈소스 HDL 시뮬레이션을 위해 사용자가 입력으로 작성하는 HDL 로직을 DEVS 기반 RTL 시뮬레이션으로의 입력 데이터(Input Data) 전환을 위해 이를 분해하는 구문 분석기(Syntax Analyzer) 구조를 포함할 수 있다.

[0073] 일례로, HDL 언어인 베릴로그(Verilog)의 언어 구조 분석을 진행할 수 있다. 주로, 타입(Type), 표현(Statement), 연산(Operation), 기능(Function)을 중심으로 언어 구조를 분석할 수 있으며, 이를 바탕으로 다음과 같은 세부 요소들을 설계할 수 있다.

[0074] 1. 베릴로그 언어 구조를 바탕으로 한 토크나이저(Tokenizer)

[0075] 2. 베릴로그 언어 구조를 바탕으로 한 구문 파서(Syntax Parser)

[0076] 3. 분석된 베릴로그 언어 구조를 바탕으로 한 AST(Abstract Syntax Tree) 구조

[0077] 베릴로그는 반도체 설계 업계에서 표준으로 사용되는 HDL 언어이므로 본 발명 또한 해당 언어를 분석하는 것을 기본적인 목표로 한다.

[0078] 사용자가 베릴로그 로직을 입력하면 이를 적절히 토큰화(tokenization)하여 로직의 구문을 분석한 뒤 AST로 변환하는 흐름을 기반으로 한 구문 분석기 구조를 설계할 수 있다. 또한, 새롭게 추가되는 구문에 대해서 효과적으로 구조의 확장이 가능한 컴파일러 구조를 포함할 수 있다.

[0079] 본 실시예들은 RTL-DEVS를 위한 타입 시스템을 설계하고 이를 바탕으로 HDL 기반의 AST를 RTL-DEVS 타입 시스템으로 전환하는 시스템 구성을 포함할 수 있다.

[0080] HDL 스펙과 앞서 정의된 디자인 스펙을 바탕으로 RTL 동작에 대한 일반화 모델을 설계한다. 일반화된 모델을 바탕으로 RTL-DEVS 동작 모델을 정의하여 구현할 수 있으며, RTL-DEVS에서 정의된 모델의 구현체를 기존의 오픈소스 도구를 활용하여 HDL로 재설계하고 두 모델의 시뮬레이션 결과를 분석하여 RTL-DEVS 동작 모델을 검증할 수 있다.

[0081] 1. RTL-DEVS 타입 시스템

[0082] DEVS의 이벤트와 시스템의 행위를 모두 타입 카테고리로 추상화하여 상위 수준의 RTL-DEVS 타입 시스템을 설계한다. 이때, 개별 타입에 대한 일반적인 행위 모델의 동작을 기술한다.

[0083] 2. RTL-DEVS 타입 시스템 기반의 DEVS 시뮬레이터

[0084] 정해진 타입 시스템을 기반으로 하여 DEVS 시뮬레이션 엔진을 포함할 수 있다.

[0085] 3. AST를 RTL-DEVS 타입 시스템으로 변환하는 천이 함수

[0086] HDL에 대한 분석 결과가 되는 AST를 RTL-DEVS 타입 시스템으로 변환하기 위한 시스템을 포함할 수 있다.

[0087] 4. RTL-DEVS 시뮬레이션 검증 수행

[0088] 검증 가능한 단순한 HDL에 대하여 정상적으로 동작하는지 검증한다. RTL 일반화 모델을 바탕으로 한 RTL-DEVS의 세부 사항을 정의하고 동작을 구현한다.

[0089] 따라서, 본 실시예들은 AST를 RTL-DEVS 타입 시스템의 구조로 변경하는 시스템 간의 천이(Transition) 모델을 포함할 수 있다.

[0090] 수학적 대수 구조를 확장한 모델 설계를 바탕으로 한 정합성 검증 과정에서, DEVS 기반 시뮬레이터 엔진의 모델 처리 방법론으로 HDL 시뮬레이션을 수행할 때 HDL의 언어적 특성으로 인해서 발생하는 결정적 속성(Deterministic Property)로 인하여 시뮬레이션 속도가 제한될 수 있다.

[0091] 이러한 문제 해결을 위해 HDL의 수학적인 특성을 고려하여 DEVS 모델을 추가적으로 확장하고 시뮬레이션 성능의 최적화를 위한 모델링을 함께 수행할 수 있다.

[0092] HDL 모델을 DEVS 모델로 변환(translation)하는 과정에서 구조적 동등성을 보장하기 위해서 HDL의 동작의 대수

적 구조를 정의하고, 시그널의 세미그룹(SemiGroup), 모노이드(Monoid)와 같은 대수 구조를 정의하고, 해당 관계를 DEVS 모델에 확장하는 형태로 구현할 수 있다.

[0093] HDL 모델의 언어의 구조적 분석 및 DEVS 형식론과 관련하여, 해당 과정에서 DEVS 시뮬레이션 방법론이 테스트 복잡성을 높이기 때문에 값(value) 중심의 시뮬레이션 방식에서 기능(Function) 중심의 시뮬레이션 방식으로 전환할 수 있다.

[0094] DEVS 모델의 경우 X, Y 입력 및 출력 집합의 값 전환에 따른 시뮬레이션을 수행할 수 있다. 그러나, 값 중심의 시뮬레이션이 동작의 정합성을 검증하기에는 높은 테스트 복잡도를 가지기 때문에, 동작의 정합성 평가를 위해 입력, 출력 값에 연결 계층(Connection Layer) 역할을 하는 와이어-동작 모델(Wire-Behavioral Model)을 새롭게 추가하여 RTL-DEVS 모델의 추상화 수준을 높이고, 관련된 모델링을 수행할 수 있다.

[0095] HDL의 언어적인 카테고리를 통해 얻어낸 HDL의 대수적 구조를 RTL-DEVS로 재정의 하는 과정에서 HDL의 대수론에 따라 DEVS 형식론이 HDL 목적에 한정하여 확장 가능한 구조를 적용할 수 있고, 이를 통해 인스턴트 천이(Instant Transition)라는 개념을 바탕으로 시뮬레이터 및 RTL-DEVS 모델의 구조를 개선할 수 있다.

[0096] 추가된 RTL-DEVS 명세를 바탕으로 결합 모델(Coupled Model)을 기술하는 작업을 수행할 수 있다. HDL은 레지스터(Register)와 레지스터의 연결이라는 DAG(Directed Acyclic Graph)와 논리 구조가 유사하므로 DAG의 수학적 특성을 시뮬레이션에 반영할 수 있다.

[0097] 따라서, 본 실시예에서는 해당 구조를 이용하여 레지스터라는 기본적인 원자 모델(Atomic Model)을 정의하고, 레지스터 간의 연결이라는 행위를 결합 모델로 정의할 수 있다. 레지스터의 연결은 다양한 형태로 발생할 수 있기에 이를 위하여 다양한 논리 연결을 위한 빌딩 블록을 설계할 수 있다.

[0098] 본 실시예들은 AST를 RTL-DEVS 타입 시스템의 구조로 변경하는 시스템 간의 천이 모델을 적용할 수 있으며, 레지스터 모델과 연결성을 갖는 와이어 모델을 구현한 후 결합 모델로 추상화함으로써 천이 모델을 만들 수 있다.

[0099] 이처럼 본 발명의 실시예들에 따르면, 오픈소스 RTL 설계 도구를 보완할 수 있는 DEVS 기반 모델을 통해 반도체 프론트엔드 설계 단계 중 RTL 설계와 HDL 변환 후 검증 단계까지의 비용이 크게 개선될 수 있고 백엔드 설계 과정에서 다시 로직 설계 단계로 돌아오는 상황을 최소화할 수 있다.

[0100] DEVS를 통해 RTL 수준에서의 모델 검증이 가능해지므로 절차가 상대적으로 기존 오픈 소스코드를 사용하는 것보다 시간 상으로 짧아지며 절차를 단축시킬 수 있다.

[0101] DEVS는 동작의 천이를 수학적 모델로 기술하며 이는 HDL 수준에서의 설계보다 동작 형태를 단순 명료하게 해준다. 이를 통해 스펙 정의 및 구현 수준에서 일차적으로 빠르게 수학적 검증을 가능하게 한다. 이는 추후 시뮬레이션 과정에서 발견되는 일부 설계상의 문제를 발생하지 않도록 하는 장점이 있다.

[0102] DEVS로 모델을 기술하고 DEVS 시뮬레이터로 동작을 검증한다. 따라서 아무리 복잡한 동작이라도 DEVS 모델의 설계 명세만으로 간단히 해결할 수 있다. 이는 기능 추가를 위해 오픈소스 도구를 추가 개발하는 시간적 비용, 해외 라이선스 비용을 지불하고 상용 도구를 사용하는 금전적 비용을 둘 다 효과적으로 절감할 수 있다.

[0103] 더 나아가, 본 발명의 실시예들에 따르면, 기존의 오픈소스 RTL 설계 도구에서 부족한 기능을 DEVS 모델로 설계하여 기존의 오픈소스 RTL 설계 도구와 함께 시뮬레이션이 가능한 HDEVS 형태의 RTL-DEVS 시뮬레이터를 제공할 수 있다. 이에 따라 기존의 오픈소스 시뮬레이션 도구에서 제공하지 못하는 기능은 RTL-DEVS로 설계하여 시뮬레이션하고, 이를 통해 RTL의 설계 스펙을 만족하는지 검증할 수 있다. 원하는 기능을 RTL-DEVS 모델을 활용하여 빠르게 기술 가능하다. 그리고, 정의된 모델을 별도의 도구 개발없이 기존의 DEVS 시뮬레이터로 통합 시뮬레이션이 가능하다.

[0104] 타입 시스템과 천이 모델의 설계를 활용하면 기존의 다양한 오픈소스 HDL 시뮬레이터에서 휴리스틱하게 진행하던 시뮬레이션 사양의 확장을 다음과 같은 4가지 과정으로 일반화할 수 있다.

[0105] 1) 스펙 정의(Spec-Definition): 확장하고자 하는 HDL의 문법적 요구사항을 정의한다.

[0106] 2) 토큰화(Tokenization): 문법적 요구사항에 대한 토큰화를 진행하거나 생략할 수 있다.

[0107] 3) 파싱(Parsing): 해당 문법에 대한 파싱 및 파싱의 결과물을 통해 AST로 변환한다.

[0108] 4) 천이(Transition): AST 노드 타입을 RTL-DEVS 타입으로 변환한다.

[0109] 만약, 새로운 HDL 스펙의 확장이 필요한 경우, 개발자는 위의 네가지 절차에 따라 구문 지원(Syntax Support)을

기계적으로 확장해 나갈 수 있다. 이는 타입 시스템에 기반한 HDL 로직의 정량적 천이만을 지원해 주기 때문에 가능하며, 해당 구조를 통하여 로직의 추가 시 기능 검증 및 테스트 단계(Functional Verification and Testing)에서 발생할 수 있는 설계 상의 문제를 빠르게 검증하여 불필요한 시뮬레이션 비용을 절감할 수 있다.

[0110] 또한, 본 발명의 실시예들에 따르면, RTL-DEVS 시뮬레이터에서 기존 오픈소스 HDL 시뮬레이터에 적용할 수 없었던 동기식/비동기식/특정-타이밍 제어를 위한 라이브러리나 무작위 시뮬레이션(Randomized Simulation)과 같은 고급 분석을 위한 확장 라이브러리를 제공함으로써 상용 도구를 사용하지 않고도 검증환경을 구축할 수 있다.

[0111] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 어플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0112] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

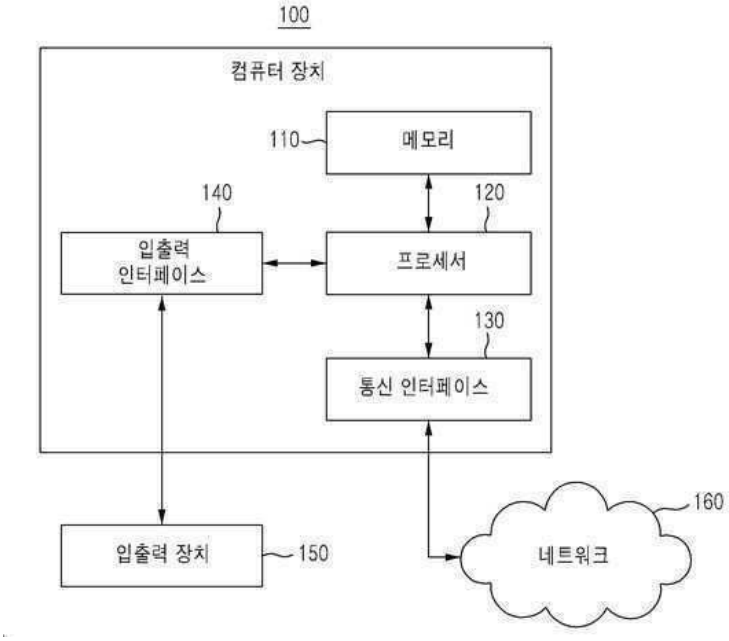
[0113] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 이때, 매체는 컴퓨터로 실행 가능한 프로그램을 계속 저장하거나, 실행 또는 다운로드를 위해 임시 저장하는 것일 수도 있다. 또한, 매체는 단일 또는 수 개의 하드웨어가 결합된 형태의 다양한 기록수단 또는 저장수단일 수 있는데, 어떤 컴퓨터 시스템에 직접 접속되는 매체에 한정되지 않고, 네트워크 상에 분산 존재하는 것일 수도 있다. 매체의 예시로는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM 및 DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical medium), 및 ROM, RAM, 플래시 메모리 등을 포함하여 프로그램 명령어가 저장되도록 구성된 것이 있을 수 있다. 또한, 다른 매체의 예시로, 어플리케이션을 유통하는 앱 스토어나 기타 다양한 소프트웨어를 공급 내지 유통하는 사이트, 서버 등에서 관리하는 기록매체 내지 저장매체도 들 수 있다.

[0114] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

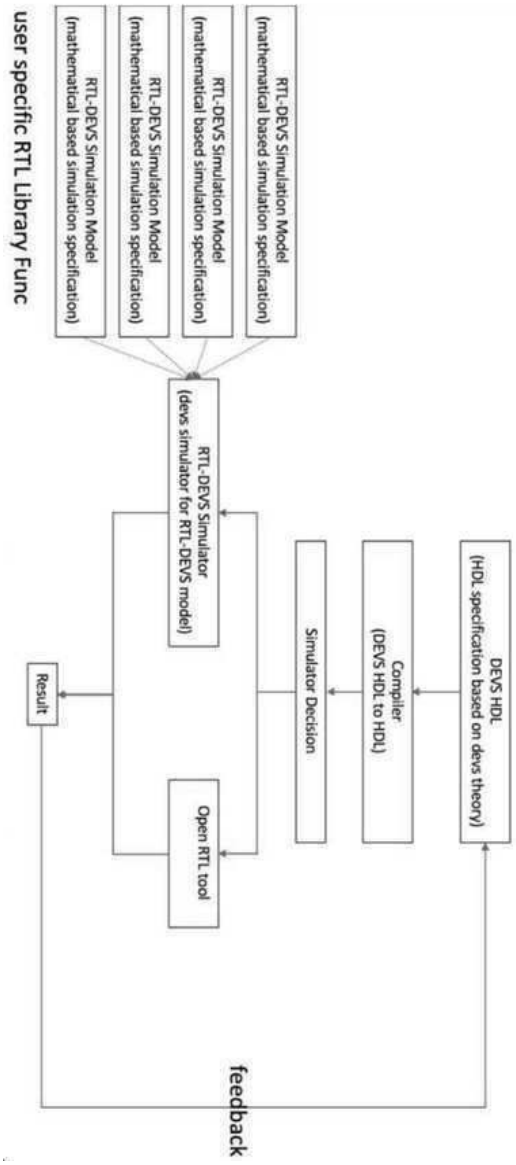
[0115] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

## 도면

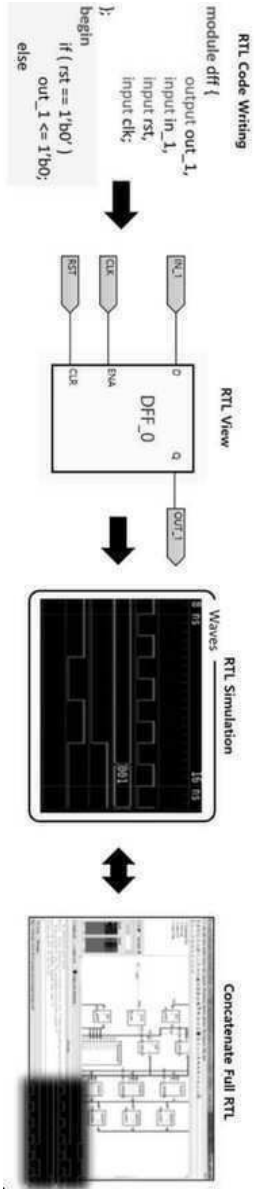
### 도면1


100
컴퓨터 장치
110
메모리
140
120
입출력
인터페이스
프로세서
130
통신 인터페이스
160
입출력 장치
150
네트워크

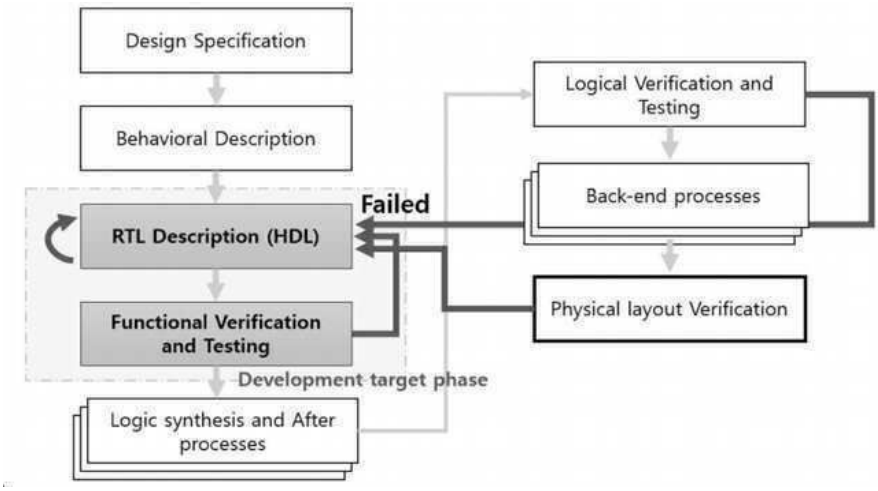
도면2


user specific RTL Library Func
RTL-DEVS Simulation Model (mathematical based simulation specification)
RTL-DEVS Simulation Model (mathematical based simulation specification)
RTL-DEVS Simulation Model (mathematical based simulation specification)
RTL-DEVS Simulation Model (mathematical based simulation specification)
RTL-DEVS Simulator (devs simulator for RTL-DEVS model)
Result
Open RTL tool
Simulator Decision
Compiler (DEVS HDL to HDL)
DEVS HDL (HDL specification based on devs theory)
feedback

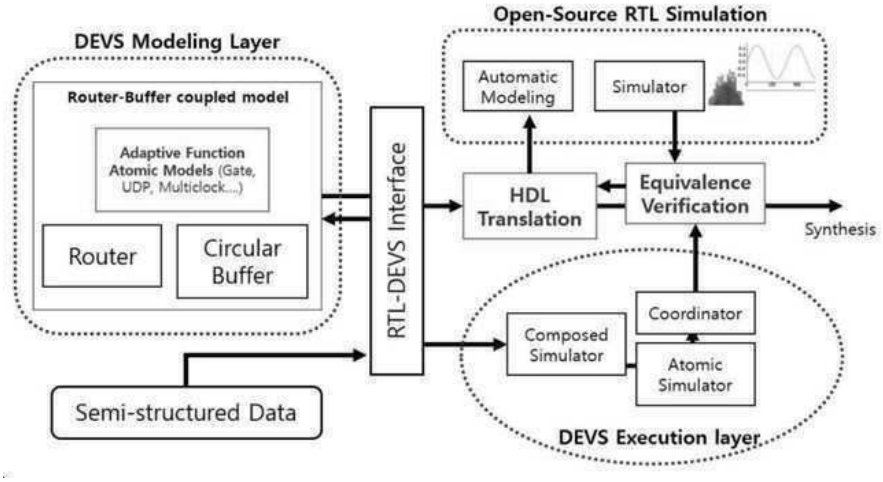
도면3


RTL Code Writing
module dff (
output out_1,
input in_1,
input rst,
input clk;
);
begin
if ( rst == 1'b0 )
out_1 <= 1'b0;
else
RST
CLK
IN_1
CLR
ENA
D
DFF_0
Q
OUT_1
RTL View
Waves
RTL Simulation
Concatenate Full RTL

도면4


Design Specification
Behavioral Description
RTL Description (HDL)
Functional Verification and Testing
Development target phase
Logic synthesis and After processes
Failed
Logical Verification and Testing
Back-end processes
Physical layout Verification

도면5


DEVS Modeling Layer
Router-Buffer coupled model
Adaptive Function Atomic Models (Gate, UDP, Multiclock....)
Router
Circular Buffer
Semi-structured Data
RTL-DEVS Interface
Open-Source RTL Simulation
Automatic Modeling
Simulator
HDL Translation
Equivalence Verification
Synthesis
Coordinator
Composed Simulator
Atomic Simulator
DEVS Execution layer