



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0094123
(43) 공개일자 2025년06월25일

- | | |
|--|--|
| <p>(51) 국제특허분류(Int. Cl.)
G06F 30/327 (2020.01) G06F 30/3308 (2020.01)
G06F 8/41 (2018.01)</p> <p>(52) CPC특허분류
G06F 30/327 (2020.01)
G06F 30/3308 (2020.01)</p> <p>(21) 출원번호 10-2023-0184660
(22) 출원일자 2023년12월18일
심사청구일자 2023년12월18일</p> | <p>(71) 출원인
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50</p> <p>(72) 발명자
문현곤
울산광역시 울주군 언양읍 유니스트길 50</p> <p>하선
울산광역시 울주군 언양읍 유니스트길 50</p> <p>(74) 대리인
특허법인비엘티</p> |
|--|--|

전체 청구항 수 : 총 20 항

(54) 발명의 명칭 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치 및 방법

(57) 요약

본 발명에 따른 도메인 특화 프로세서의 ISA 명세서 자동 추출을 수행하는 전자 장치는, 레지스터 전송 레벨(RTL) 프로세서 및 적어도 하나의 명령어를 기호 실행하여 비트 벡터 시맨틱(Bit-Vector Semantic)을 출력하는 기호 실행기(Smbolic simulator); 상기 기호 실행기에 상기 명령어를 위한 입력을 전달하고, 상기 기호 실행기로 (뒷면에 계속)

대표도 - 도1

100



부터 출력을 전달받는 외부 메모리; 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 상기 명령어에 대한 ISA 명세를 생성하는 코드 생성기; 및 상기 기호 실행기, 상기 외부 메모리 및 상기 코드 생성기를 제어하는 적어도 하나의 프로세서;를 포함하며, 상기 적어도 하나의 프로세서는, 상기 기호 실행기로 도메인 특화 프로세서에 대응하는 상기 RTL 프로세서 및 기저장된 명령어 비트 패턴 집합 중 적어도 하나의 명령어를 입력하고, 상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하고, 출력한 상기 비트 벡터 시맨틱을 상기 코드 생성기에 입력하고, 상기 코드 생성기가 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 세일 명령어 함수를 출력하도록 구성될 수 있다.

(52) CPC특허분류

G06F 8/41 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711193910
과제번호	2021-0-01817-003
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	하이퍼-컴포저블 데이터센터를 위한 차세대 컴퓨팅 기술 개발
기 여 율	1/1
과제수행기관명	울산과학기술원
연구기간	2023.01.01 ~ 2023.12.31

명세서

청구범위

청구항 1

레지스터 전송 레벨(RTL) 프로세서 및 적어도 하나의 명령어를 기호 실행하여 비트 벡터 시맨틱(Bit-Vector Semantic)을 출력하는 기호 실행기(Smbolic simulator);

상기 기호 실행기에 상기 명령어를 위한 입력을 전달하고, 상기 기호 실행기로부터 출력을 전달받는 외부 메모리;

상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 상기 명령어에 대한 ISA 명세를 생성하는 코드 생성기; 및

상기 기호 실행기, 상기 외부 메모리 및 상기 코드 생성기를 제어하는 적어도 하나의 프로세서;를 포함하며,

상기 적어도 하나의 프로세서는,

상기 기호 실행기로 도메인 특화 프로세서에 대응하는 상기 RTL 프로세서 및 기저장된 명령어 비트 패턴 집합 중 적어도 하나의 명령어를 입력하고,

상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하고,

출력한 상기 비트 벡터 시맨틱을 상기 코드 생성기에 입력하고,

상기 코드 생성기가 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 세일 명령어 함수를 출력하도록 구성됨

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 2

제1 항에 있어서,

상기 적어도 하나의 프로세서는,

상기 기호 실행기가 출력한 명령어의 상태 변화와 상기 외부 메모리와의 상호작용 상태를 기반으로 상기 명령어에 대한 비트 벡터 시맨틱을 출력하도록 구성됨

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 3

제2 항에 있어서,

상기 적어도 하나의 프로세서는,

상기 코드 생성기가 상기 명령어에 대한 비트 벡터 시맨틱을 트리 패턴 매칭을 사용하여 상기 세일 언어로 컴파일하도록 구성됨

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 4

제3 항에 있어서,

상기 적어도 하나의 프로세서는,

상기 코드 생성기가 기설정된 상태 요소 맵핑 명세를 고려하여 상기 RTL 프로세서의 상태명과 대응하는 상기 세일 언어의 상태명으로 컴파일하도록 구성됨

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 5

제4 항에 있어서,

상기 적어도 하나의 프로세서는,

상기 세일 명령어 함수와 세일 스켈레톤 모델을 조합하여 이플레이터를 생성하도록 구성된

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 6

제5 항에 있어서,

상기 세일 스켈레톤 모델은 상기 세일 언어로 작성된 Sail-RISC-V 모델을 기반으로 하는 모델인 것을 특징으로 하는

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 7

제1 항에 있어서,

상기 기호 실행기는

상기 RTL 프로세서의 하드웨어를 리셋하여 상기 RTL 프로세서의 파이프라인을 초기화하는 디버그 인터페이스와,

상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 시뮬레이션 코어와,

상기 외부 메모리에 연결된 메모리 인터페이스를 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 8

제7 항에 있어서,

상기 적어도 하나의 프로세서는,

상기 디버그 인터페이스가 상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션하여 기호화된 명령어를 상기 RTL 프로세서에서 실행함으로써 상기 명령어의 초기화시 의미(Reset semantics)를 추출하도록 구성된

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 9

제8 항에 있어서,

상기 적어도 하나의 프로세서는,

상기 메모리 인터페이스를 통해 상기 외부 메모리가 상기 시뮬레이션 코어로 명령어 읽기(fetch) 접근에 대해 첫 사이클에는 시험대상 명령어를 반환하고, 나머지 사이클에는 NOP 명령어를 반환하도록 구성된

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 10

제9 항에 있어서,

상기 적어도 하나의 프로세서는,

상기 시뮬레이션 코어가 상기NOP 명령어와 미 구현 명령어를 사용해 RTL 프로세서를 시뮬레이션하고 시뮬레이션 과정에서 상기 명령어의 상태 변화를 학습하도록 구성된

도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치.

청구항 11

레지스터 전송 레벨(RTL) 프로세서 및 적어도 하나의 명령어를 기호 실행하여 비트 벡터 시맨틱(Bit-Vector Semantic)을 출력하는 기호 실행기(Smbolic simulator)와 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 상기 명령어에 대한 ISA 명세를 생성하는 코드 생성기를 포함하는 전자 장치의 도메인 특화 프로세서의 ISA 명세 자동 추출 방법으로서,

상기 기호 실행기로 도메인 특화 프로세서에 대응하는 상기 RTL 프로세서 및 기저장된 명령어 비트 패턴 집합 중 적어도 하나의 명령어를 입력하는 단계;

상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계;

출력한 상기 비트 벡터 시맨틱을 상기 코드 생성기에 입력하는 단계; 및

상기 코드 생성기가 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 세일 명령어 함수를 출력하는 단계를 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 12

제11 항에 있어서,

상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계는,

상기 기호 실행기가 출력한 명령어의 상태 변화와 상기 외부 메모리와의 상호작용 상태를 기반으로 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계를 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 13

제12 항에 있어서,

상기 세일 명령어 함수를 출력하는 단계는,

상기 코드 생성기가 상기 명령어에 대한 비트 벡터 시맨틱을 트리 패턴 매칭을 사용하여 상기 세일 언어로 컴파일하는 단계를 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 14

제13 항에 있어서,

상기 세일 명령어 함수를 출력하는 단계는,

상기 코드 생성기가 기설정된 상태 요소 맵핑 명세를 고려하여 상기 RTL 프로세서의 상태명과 대응하는 상기 세일 언어의 상태명으로 컴파일하는 단계를 더 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 15

제14 항에 있어서,

상기 방법은,

상기 세일 명령어 함수와 세일 스켈레톤 모델을 조합하여 이플레이터를 생성하는 단계를 더 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 16

제15 항에 있어서,

상기 세일 스켈레톤 모델은 상기 세일 언어로 작성된 Sail-RISC-V 모델을 기반으로 하는 모델인 것을 특징으로 하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 17

제11 항에 있어서,

상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계는,

상기 RTL 프로세서의 하드웨어를 리셋하여 상기 RTL 프로세서의 파이프라인을 초기화하는 단계; 및

상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 단계; 를 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 18

제17 항에 있어서,

상기 RTL 프로세서의 파이프라인을 초기화하는 단계는,

상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션하여 기호화된 명령어를 상기 RTL 프로세서에서 실행함으로써 상기 명령어의 초기화시 의미(Reset semantics)를 추출하는 단계를 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 19

제18 항에 있어서,

상기 RTL 프로세서의 파이프라인을 초기화하는 단계 후에,

상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 단계는,

상기 메모리 인터페이스를 통해 상기 외부 메모리가 상기 시뮬레이션 코어로 명령어 읽기(fetch) 접근에 대해 첫 사이클에는 시험대상 명령어를 반환하고, 나머지 사이클에는 NOP 명령어를 반환하는 단계를 더 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

청구항 20

제19 항에 있어서,

상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 단계는,

상기 NOP 명령어와 미 구현 명령어를 사용해 RTL 프로세서를 시뮬레이션하고 시뮬레이션 과정에서 상기 명령어의 상태 변화를 학습하는 단계를 포함하는

도메인 특화 프로세서의 ISA 명세 자동 추출 방법.

발명의 설명

기술 분야

[0001] 본 발명은 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치 및 방법에 관한 것이다.

배경 기술

[0002] 공정 측면에서의 성능 향상이 한계에 다다르면서 점점 많은 분야에서 도메인 특화 프로세서가 개발 및 사용되고 있다. 특정 도메인에 대해 전문화된 명령어를 지원함으로써 해당 분야 응용이 더 높은 성능을 얻을 수 있도록 한다.

[0003] 도메인 특화 프로세서는 특정 도메인의 연산 특성을 고려해 설계됨으로써 해당 도메인에서 고성능 고효율을 달

성하는 프로세서의 한 종류로, 그래픽 처리 장치(GPU), 텐서 처리 장치(TPU) 등이 그 예이다.

[0004] 도메인 특화 프로세서에서 사용되는 프로그램은 명령어 집합이 특화 명령어를 포함하기 때문에 소프트웨어 버그를 자동으로 찾는 하이브리드 피징, 기호 실행 기술 등 범용적인 프로그램을 해당 명령어 추가 없이 사용하기가 어렵다. 상기 특수 명령어를 포함하는 프로세서를 위한 프로그램을 작성 및 분석하기 위하여 그에 맞는 컴파일러 또는 분석기를 개발하여야 한다.

[0005] 다시 말해, 새로운 명령어를 지원하도록 수동으로 확장하지 않으면 고급 언어용 도구나 바이너리용 도구 모두 도메인 특화 구조 프로그램을 분석하는데 사용하기가 어려웠다.

[0006] 이를 자동 확장하기 위하여 각 명령어의 시맨틱이 형식 언어로 표현되도록 하여야 하나, 명령어들의 시맨틱은 일반적으로 거대하고 복잡하며, 대부분의 매뉴얼이 자연어로 작성되어 정확한 사양을 관리하고 작성하기가 어렵다.

선행기술문헌

비특허문헌

[0007] (비특허문헌 0001) J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," Commun., ACM, vol. 62, no.2, p. 48-60, Jan, 2019.

발명의 내용

해결하려는 과제

[0008] 본 발명의 실시예는 상술한 문제점을 해결하기 위하여 제안된 것으로, 레지스터 전송 레벨(RTL, Register Transfer Level)의 하드웨어 설명 언어로 구현된 도메인 특화 구조로부터 자동으로 시맨틱을 추출할 수 있는, 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치 및 방법을 제공할 수 있다.

[0009] 본 발명이 해결하고자 하는 과제들은 이상에서 언급된 과제로 제한되지 않으며, 언급되지 않은 또 다른 과제들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0010] 상술한 과제를 해결하기 위하여, 본 발명에 따른 도메인 특화 프로세서의 ISA 명세서 자동 추출을 수행하는 전자 장치는, 레지스터 전송 레벨(RTL) 프로세서 및 적어도 하나의 명령어를 기호 실행하여 비트 벡터 시맨틱(Bit-Vector Semantic)을 출력하는 기호 실행기(Smbolic simulator); 상기 기호 실행기에 상기 명령어를 위한 입력을 전달하고, 상기 기호실행기로부터 출력을 전달받는 외부 메모리; 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 상기 명령어에 대한 ISA 명세를 생성하는 코드 생성기; 및 상기 기호 실행기, 상기 외부 메모리 및 상기 코드 생성기를 제어하는 적어도 하나의 프로세서;를 포함하며, 상기 적어도 하나의 프로세서는, 상기 기호 실행기로 도메인 특화 프로세서에 대응하는 상기 RTL 프로세서 및 기저장된 명령어 비트 패턴 집합 중 적어도 하나의 명령어를 입력하고, 상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하고, 출력한 상기 비트 벡터 시맨틱을 상기 코드 생성기에 입력하고, 상기 코드 생성기가 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 세일 명령어 함수를 출력하도록 구성될 수 있다.

[0011] 상술한 과제를 해결하기 위하여, 본 발명에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출 방법은, 레지스터 전송 레벨(RTL) 프로세서 및 적어도 하나의 명령어를 기호 실행하여 비트 벡터 시맨틱(Bit-Vector Semantic)을 출력하는 기호 실행기(Smbolic simulator)와 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 상기 명령어에 대한 ISA 명세를 생성하는 코드 생성기를 포함하는 전자 장치의 도메인 특화 프로세서의 ISA 명세 자동 추출 방법으로서, 상기 기호 실행기로 도메인 특화 프로세서에 대응하는 상기 RTL 프로세서 및 기저장된 명령어 비트 패턴 집합 중 적어도 하나의 명령어를 입력하는 단계; 상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계; 출력한 상기 비트 벡터 시맨틱을 상기 코드 생성기에 입력하는 단계; 및 상기 코드 생성기가 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 세일 명령어 함수를 출력하는 단계를 포함할 수 있다.

[0012] 이 외에도, 본 발명을 구현하기 위한 다른 방법, 다른 시스템 및 상기 방법을 실행하기 위한 컴퓨터 프로그램을

기록하는 컴퓨터로 판독 가능한 기록 매체가 더 제공될 수 있다.

[0013] 이 외에도, 본 발명을 구현하기 위한 방법이 컴퓨터 상에서 수행되도록 매체에 저장되는 컴퓨터 프로그램이 더 제공될 수 있다.

발명의 효과

[0014] 본 발명의 진술한 과제 해결 수단에 의하면, RTL 프로세서를 이용하여 명령어의 시맨틱을 자동으로 추출하고, 각 명령어의 시맨틱을 ISA 명세서 작성을 위해 개발된 언어로 표현하여 ISA 수준 시뮬레이션 또는 이를 기반으로 한 분석 도구 생성에 사용되도록 하는 효과를 제공할 수 있다.

[0015] 본 발명의 효과들은 이상에서 언급된 효과로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일 실시예에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치의 구성을 간략하게 도시한 블록도이다.

도 2는 본 발명의 일 실시예에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치의 ISA 명령어 추출 과정을 도시한 블록도이다.

도 3은 본 발명의 일 실시예에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치에 입력되는 명령어 비트 패턴 집합의 추가 명령어를 도시한 것이다.

도 4는 본 발명의 일 실시예에 따른 비트 벡터 시맨틱을 나타낸 개략도이다.

도 5는 본 발명의 일 실시예에 따른 기호 실행기의 구성을 간략하게 도시한 블록도이다.

도 6은 본 발명의 일 실시예에 따른 세일 함수를 나타낸 개략도이다.

도 7은 Sail-riscv 참조 모델과 본 발명에 따른 전자 장치가 생성한 코드를 XOR 명령어를 이용하여 비교한 코딩을 도시한 도면이다.

도 8은 본 발명의 일 실시예에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 방법의 플로우 차트를 도시한 것이다.

발명을 실시하기 위한 구체적인 내용

[0017] 본 발명 전체에 걸쳐 동일 참조 부호는 동일 구성요소를 지칭한다. 본 발명이 실시예들의 모든 요소들을 설명하는 것은 아니며, 본 발명이 속하는 기술분야에서 일반적인 내용 또는 실시예들 간에 중복되는 내용은 생략한다. 명세서에서 사용되는 '부, 모듈, 부재, 블록'이라는 용어는 소프트웨어 또는 하드웨어로 구현될 수 있으며, 실시예들에 따라 복수의 '부, 모듈, 부재, 블록'이 하나의 구성요소로 구현되거나, 하나의 '부, 모듈, 부재, 블록'이 복수의 구성요소들을 포함하는 것도 가능하다.

[0018] 명세서 전체에서, 어떤 부분이 다른 부분과 "연결"되어 있다고 할 때, 이는 직접적으로 연결되어 있는 경우뿐 아니라, 간접적으로 연결되어 있는 경우를 포함하고, 간접적인 연결은 무선 통신망을 통해 연결되는 것을 포함한다.

[0019] 또한 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.

[0020] 명세서 전체에서, 어떤 부재가 다른 부재 "상에" 위치하고 있다고 할 때, 이는 어떤 부재가 다른 부재에 접해 있는 경우뿐 아니라 두 부재 사이에 또 다른 부재가 존재하는 경우도 포함한다.

[0021] 제 1, 제 2 등의 용어는 하나의 구성요소를 다른 구성요소로부터 구별하기 위해 사용되는 것으로, 구성요소가 전술된 용어들에 의해 제한되는 것은 아니다.

[0022] 단수의 표현은 문맥상 명백하게 예외가 있지 않는 한, 복수의 표현을 포함한다.

[0023] 각 단계들에 있어 식별부호는 설명의 편의를 위하여 사용되는 것으로 식별부호는 각 단계들의 순서를 설명하는 것이 아니며, 각 단계들은 문맥상 명백하게 특정 순서를 기재하지 않는 이상 명기된 순서와 다르게 실시될 수

있다.

- [0024] 이하 첨부된 도면들을 참고하여 본 발명의 작용 원리 및 실시예들에 대해 설명한다.
- [0025] 본 명세서에서 '본 발명에 따른 장치'는 연산처리를 수행하여 사용자에게 결과를 제공할 수 있는 다양한 장치들이 모두 포함된다. 예를 들어, 본 발명에 따른 장치는, 컴퓨터, 서버 장치 및 휴대용 단말기를 모두 포함하거나, 또는 어느 하나의 형태가 될 수 있다.
- [0026] 여기에서, 상기 컴퓨터는 예를 들어, 웹 브라우저(WEB Browser)가 탑재된 노트북, 데스크톱(desktop), 랩톱(laptop), 태블릿 PC, 슬레이트 PC 등을 포함할 수 있다.
- [0027] 상기 서버 장치는 외부 장치와 통신을 수행하여 정보를 처리하는 서버로써, 애플리케이션 서버, 컴퓨팅 서버, 데이터베이스 서버, 파일 서버, 게임 서버, 메일 서버, 프록시 서버 및 웹 서버 등을 포함할 수 있다.
- [0028] 상기 휴대용 단말기는 예를 들어, 휴대성과 이동성이 보장되는 무선 통신 장치로서, PCS(Personal Communication System), GSM(Global System for Mobile communications), PDC(Personal Digital Cellular), PHS(Personal Handyphone System), PDA(Personal Digital Assistant), IMT(International Mobile Telecommunication)-2000, CDMA(Code Division Multiple Access)-2000, W-CDMA(W-Code Division Multiple Access), WiBro(Wireless Broadband Internet) 단말, 스마트폰(Smart Phone) 등과 같은 모든 종류의 핸드헬드(Handheld) 기반의 무선 통신 장치와 시계, 반지, 팔찌, 발찌, 목걸이, 안경, 콘택트 렌즈, 또는 머리 착용형 장치(head-mounted-device(HMD)) 등과 같은 웨어러블 장치를 포함할 수 있다.
- [0029] 본 발명에 따른 인공지능과 관련된 기능은 프로세서와 메모리를 통해 동작된다. 프로세서는 하나 또는 복수의 프로세서로 구성될 수 있다. 이때, 하나 또는 복수의 프로세서는 CPU, AP, DSP(Digital Signal Processor) 등과 같은 범용 프로세서, GPU, VPU(Vision Processing Unit)와 같은 그래픽 전용 프로세서 또는 NPU와 같은 인공지능 전용 프로세서일 수 있다. 하나 또는 복수의 프로세서는, 메모리에 저장된 기 정의된 동작 규칙 또는 인공지능 모델에 따라, 입력 데이터를 처리하도록 제어한다. 또는, 하나 또는 복수의 프로세서가 인공지능 전용 프로세서인 경우, 인공지능 전용 프로세서는, 특정 인공지능 모델의 처리에 특화된 하드웨어 구조로 설계될 수 있다.
- [0030] 기 정의된 동작 규칙 또는 인공지능 모델은 학습을 통해 만들어진 것을 특징으로 한다. 여기서, 학습을 통해 만들어진다는 것은, 기본 인공지능 모델이 학습 알고리즘에 의하여 다수의 학습 데이터들을 이용하여 학습됨으로써, 원하는 특성(또는, 목적)을 수행하도록 설정된 기 정의된 동작 규칙 또는 인공지능 모델이 만들어짐을 의미한다. 이러한 학습은 본 발명에 따른 인공지능이 수행되는 기기 자체에서 이루어질 수도 있고, 별도의 서버 및/또는 시스템을 통해 이루어 질 수도 있다. 학습 알고리즘의 예로는, 지도형 학습(supervised learning), 비지도형 학습(unsupervised learning), 준지도형 학습(semi-supervised learning) 또는 강화 학습(reinforcement learning)이 있으나, 전술한 예에 한정되지 않는다.
- [0031] 인공지능 모델은, 복수의 신경망 레이어들로 구성될 수 있다. 복수의 신경망 레이어들 각각은 복수의 가중치들(weight values)을 갖고 있으며, 이전(previous) 레이어의 연산 결과와 복수의 가중치들 간의 연산을 통해 신경망 연산을 수행한다. 복수의 신경망 레이어들이 갖고 있는 복수의 가중치들은 인공지능 모델의 학습 결과에 의해 최적화될 수 있다. 예를 들어, 학습 과정 동안 인공지능 모델에서 획득한 로스(loss) 값 또는 코스트(cost) 값이 감소 또는 최소화되도록 복수의 가중치들이 갱신될 수 있다. 인공지능 신경망은 심층 신경망(DNN:Deep Neural Network)를 포함할 수 있으며, 예를 들어, CNN (Convolutional Neural Network), DNN (Deep Neural Network), RNN (Recurrent Neural Network), RBM (Restricted Boltzmann Machine), DBN (Deep Belief Network), BRDNN(Bidirectional Recurrent Deep Neural Network) 또는 심층 Q-네트워크 (Deep Q-Networks) 등이 있으나, 전술한 예에 한정되지 않는다.
- [0032] 본 발명의 예시적인 실시예에 따르면, 프로세서는 인공지능을 구현할 수 있다. 인공지능이란 사람의 신경세포(biological neuron)를 모사하여 기계가 학습하도록 하는 인공신경망(Artificial Neural Network) 기반의 기계 학습법을 의미한다. 인공지능의 방법론에는 학습 방식에 따라 훈련데이터로서 입력데이터와 출력데이터가 같이 제공됨으로써 문제(입력데이터)의 해답(출력데이터)이 정해져 있는 지도학습(supervised learning), 및 출력데이터 없이 입력데이터만 제공되어 문제(입력데이터)의 해답(출력데이터)이 정해지지 않는 비지도학습(unsupervised learning), 및 현재의 상태(State)에서 어떤 행동(Action)을 취할 때마다 외부 환경에서 보상(Reward)이 주어지는데, 이러한 보상을 최대화하는 방향으로 학습을 진행하는 강화학습(reinforcement learning)으로 구분될 수 있다. 또한, 인공지능의 방법론은 학습 모델의 구조인 아키텍처에 따라 구분될 수도

있는데, 널리 이용되는 딥러닝 기술의 아키텍처는, 합성곱신경망(CNN; Convolutional Neural Network), 순환신경망(RNN; Recurrent Neural Network), 트랜스포머(Transformer), 생성적 대립 신경망(GAN; generative adversarial networks) 등으로 구분될 수 있다.

[0033] 본 장치와 시스템은 인공 지능 모델을 포함할 수 있다. 인공 지능 모델은 하나의 인공 지능 모델일 수 있고, 복수의 인공 지능 모델로 구현될 수도 있다. 인공 지능 모델은 뉴럴 네트워크(또는 인공 신경망)로 구성될 수 있으며, 기계학습과 인지과학에서 생물학의 신경을 모방한 통계학적 학습 알고리즘을 포함할 수 있다. 뉴럴 네트워크는 시냅스의 결합으로 네트워크를 형성한 인공 뉴런(노드)이 학습을 통해 시냅스의 결합 세기를 변화시켜, 문제 해결 능력을 가지는 모델 전반을 의미할 수 있다. 뉴럴 네트워크의 뉴런은 가중치 또는 바이어스의 조합을 포함할 수 있다. 뉴럴 네트워크는 하나 이상의 뉴런 또는 노드로 구성된 하나 이상의 레이어(layer)를 포함할 수 있다. 예시적으로, 장치는 input layer, hidden layer, output layer를 포함할 수 있다. 장치를 구성하는 뉴럴 네트워크는 뉴런의 가중치를 학습을 통해 변화시킴으로써 임의의 입력(input)으로부터 예측하고자 하는 결과(output)를 추론할 수 있다.

[0034] 프로세서는 뉴럴 네트워크를 생성하거나, 뉴럴 네트워크를 훈련(train, 또는 학습(learn)하거나, 수신되는 입력 데이터를 기초로 연산을 수행하고, 수행 결과를 기초로 정보 신호(information signal)를 생성하거나, 뉴럴 네트워크를 재훈련(retrain)할 수 있다. 뉴럴 네트워크의 모델들은 GoogleNet, AlexNet, VGG Network 등과 같은 CNN(Convolution Neural Network), R-CNN(Region with Convolution Neural Network), RPN(Region Proposal Network), RNN(Recurrent Neural Network), S-DNN(Stacking-based deep Neural Network), S-SDNN(State-Space Dynamic Neural Network), Deconvolution Network, DBN(Deep Belief Network), RBM(Restrcted Boltzman Machine), Fully Convolutional Network, LSTM(Long Short-Term Memory) Network, Classification Network 등 다양한 종류의 모델들을 포함할 수 있으나 이에 제한되지는 않는다. 프로세서는 뉴럴 네트워크의 모델들에 따른 연산을 수행하기 위한 하나 이상의 프로세서를 포함할 수 있다. 예를 들어 뉴럴 네트워크는 심층 뉴럴 네트워크(Deep Neural Network)를 포함할 수 있다.

[0035] 뉴럴 네트워크는 CNN(Convolutional Neural Network), RNN(Recurrent Neural Network), 퍼셉트론(perceptron), 다층 퍼셉트론(multilayer perceptron), FF(Feed Forward), RBF(Radial Basis Network), DFF(Deep Feed Forward), LSTM(Long Short Term Memory), GRU(Gated Recurrent Unit), AE(Auto Encoder), VAE(Variational Auto Encoder), DAE(Denoising Auto Encoder), SAE(Sparse Auto Encoder), MC(Markov Chain), HN(Hopfield Network), BM(Boltzmann Machine), RBM(Restricted Boltzmann Machine), DBN(Depp Belief Network), DCN(Deep Convolutional Network), DN(Deconvolutional Network), DCIGN(Deep Convolutional Inverse Graphics Network), GAN(Generative Adversarial Network), LSM(Liquid State Machine), ELM(Extreme Learning Machine), ESN(Echo State Network), DRN(Deep Residual Network), DNC(Differentiable Neural Computer), NTM(Neural Turning Machine), CN(Capsule Network), KN(Kohonen Network) 및 AN(Attention Network)를 포함할 수 있으나 이에 한정되는 것이 아닌 임의의 뉴럴 네트워크를 포함할 수 있음은 통상의 기술자가 이해할 것이다.

[0036] 본 발명의 예시적인 실시예에 따르면, 프로세서는 GoogleNet, AlexNet, VGG Network 등과 같은 CNN(Convolution Neural Network), R-CNN(Region with Convolution Neural Network), RPN(Region Proposal Network), RNN(Recurrent Neural Network), S-DNN(Stacking-based deep Neural Network), S-SDNN(State-Space Dynamic Neural Network), Deconvolution Network, DBN(Deep Belief Network), RBM(Restrcted Boltzman Machine), Fully Convolutional Network, LSTM(Long Short-Term Memory) Network, Classification Network, Generative Modeling, eXplainable AI, Continual AI, Representation Learning, AI for Material Design, 자연어 처리를 위한 BERT, SP-BERT, MRC/QA, Text Analysis, Dialog System, GPT-3, GPT-4, 비전 처리를 위한 Visual Analytics, Visual Understanding, Video Synthesis, ResNet 데이터 지능을 위한 Anomaly Detection, Prediction, Time-Series Forecasting, Optimization, Recommendation, Data Creation 등 다양한 인공지능 구조 및 알고리즘을 이용할 수 있으며, 이에 제한되지 않는다. 이하, 첨부된 도면을 참조하여 본 발명의 실시예를 상세하게 설명한다.

[0037] 도 1은 본 발명의 일 실시예에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치(100)의 구성을 간략하게 도시한 블록도이고, 도 2는 본 발명의 일 실시예에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치(100)의 ISA 명령어 추출 과정을 도시한 블록도, 도 3은 입력되는 명령어 비트 패턴 집합의 추가 명령어를 도시한 것이다.

- [0038] 도 4는 본 발명의 일 실시예에 따른 비트 벡터 시맨틱을 나타낸 개략도이며, 도 5는 본 발명의 일 실시예에 따른 기호 실행기(110)의 구성을 간략하게 도시한 블록도이다.
- [0039] 도 1에 도시된 바와 같이, 본 발명에 따른 도메인 특화 프로세서의 ISA 명세서 자동 추출을 수행하는 전자 장치(100)는, 레지스터 전송 레벨(RTL) 프로세서 및 적어도 하나의 명령어를 기호 실행하여 비트 벡터 시맨틱(Bit-Vector Semantic)을 출력하는 기호 실행기(Smbolic simulator, 110), 상기 기호 실행기(110)에 상기 명령어를 위한 입력을 전달하고, 상기 기호 실행기(110)로부터 출력을 전달받는 외부 메모리(120), 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 상기 명령어에 대한 ISA 명세를 생성하는 코드 생성기(130), 및 상기 기호 실행기(110), 상기 외부 메모리(120) 및 상기 코드 생성기(130)를 제어하는 적어도 하나의 프로세서(140)를 포함할 수 있다.
- [0040] 일 실시예로서, 상기 적어도 하나의 프로세서(140)는, 상기 기호 실행기(110)로 도메인 특화 프로세서에 대응하는 상기 RTL 프로세서 및 기저장된 명령어 비트 패턴 집합 중 적어도 하나의 명령어를 입력하고, 상기 기호 실행기(110)를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하고, 출력한 상기 비트 벡터 시맨틱을 상기 코드 생성기(130)에 입력하고, 상기 코드 생성기(130)가 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 세일 명령어 함수를 출력하도록 구성될 수 있다.
- [0041] 본 발명에 따른 전자 장치(100)는 RTL 프로세서(11), RTL 프로세서(11)에 입력되는 명령어 비트 벡터 집합, 명령어 시맨틱의 상태 목록을 입력으로 받을 수 있다.
- [0042] 구체적으로, 도 2에 도시된 바와 같이, 일 실시예로서, 상기 적어도 하나의 프로세서(140)는 RTL 프로세서(11) 및 명령어 비트 패턴 집합(12)을 기호 실행기(110)로 입력할 수 있다.
- [0043] 기호 실행기(110)는 RTL 프로세서(11) 및 명령어 비트 패턴 집합(12)을 기반으로 명령어 비트 벡터 시맨틱(21)을 출력할 수 있다.
- [0044] 상기 명령어 비트 패턴 집합(12)의 명세는, 도 3에 도시된 바와 같은 명령어 구조로 표현될 수 있다.
- [0045] 일 실시예로서, 상기 적어도 하나의 프로세서(140)는 상기 기호 실행기(110)가 출력한 명령어의 상태 변화와 상기 외부 메모리(120)와의 상호작용 상태를 기반으로 상기 명령어에 대한 비트 벡터 시맨틱을 출력하도록 구성될 수 있다.
- [0046] 상기 비트 벡터 시맨틱은 도 4에 도시된 바와 같이, 의미 명세의 한 부분으로 생성될 수 있다. 조건 표현과 참 표현, 거짓 표현으로 구분하여 명세를 구분할 수 있다.
- [0047] 상기 기호 실행기(110)는 외부 메모리(120)와 반복하여 명령어 비트 벡터 시맨틱(21) 출력을 위해 필요한 정보를 주고 받으며, 명령어 비트 패턴 집합(12) 내의 패턴이 떨어질 때까지 반복할 수 있다.
- [0048] RTL 프로세서(11)가 구현하는 명령어 집합을 식별하고 시맨틱 체계를 따르기 위해서, 본 발명에 따른 전자 장치(100)의 기호 실행기(110)는 사용자로부터 입력 받은 NOP 명령어와 미 구현 명령어를 사용하여 RTL 프로세서(110)를 시뮬레이션할 수 있다.
- [0049] 상기 기호 실행기(110)는 시뮬레이션 과정에서 NOP 명령어와 미 구현 명령어의 실행 과정에서 발생하는 상태 변화를 학습할 수 있다. 학습된 상태 변화는 테스트 중인 기호 명령어가 구현되었는지 여부를 확인하고 프로그램 카운터 레지스터를 인식하는데 사용될 수 있다.
- [0050] 본 발명에 따른 전자 장치(100)는 각 명령어의 의미명세 추출을 위해 NOP 명령어의 의미 명세를 가질 수 있다. 다른 명령어의 명세 추출 과정을 시작하기 전에 기저장되거나 전달 받은 알려진 NOP 명령어를 이용하여 NOP 명령어의 동작을 학습하고 NOP 명령어의 의미명세를 추출하기 위한 시뮬레이션을 수행할 수 있다.
- [0051] 또한, 각 상태의 값이 일정하게 유지되거나 선형적으로 변경되는 경우 시뮬레이션을 종료할 수 있다. 추적하는 명령어의 상태 변화 집합이 더 이상 변경되지 않고, RTL 프로세서의 사이클이 변경되어도 실행 결과가 NOP 명령어의 결과일 때 RTL 프로세서가 안정된 상태라고 간주하고 한 번의 시뮬레이션을 종료할 수 있다.
- [0052] 외부 메모리(120)는 첫 사이클에는 시험대상 명령어인 기호 명령어를 반환하고, 나머지 사이클에는 NOP 명령어를 반환할 수 있다. 이를 통해 RTL 프로세서(11)가 액세스하는 기호 명령어 및 데이터의 주소를 기록할 수 있다. RTL 프로세서(11)로부터 명령어 fetch 요청 주소를 확인하여 이전 위치의 바로 다음 위치가 요청되지 않는 경우 이를 점프로 간주하여 그 주소를 기록할 수 있다.

- [0053] 더하여, 본 발명에 따른 전자 장치(100)는 미 구현 명령어(Illegal instruction)를 기설정된 명령어 리스트와 비교하여 인식하고 추출에서 제외하여야 한다. 프로세서들이 일반적으로 미구현 명령어를 실행하는 경우 이에 해당하는 예외를 발생시키는 바, 모든 미구현 명령어에 대해 같은 동작을 한다는 점을 이용하여 임의의 기호 명령어를 실행한 결과 동일한 의미 명세가 추출될 경우 상기 명령어를 미 구현 명령어로 분류할 수 있다.
- [0054] 일 실시예로서, 상기 코드 생성기(130)가 기설정된 상태 요소 맵핑 명세(13)를 고려하여 상기 RTL 프로세서(11)의 상태명과 대응하는 상기 세일 언어의 상태명으로 컴파일하도록 구성될 수 있다.
- [0055] 상기 명령어 비트 벡터 시맨틱(21)에 대해 상태 요소 맵핑 명세(13)에 따라 RTL 프로세서(11)의 상태명과 대응하는 세일 언어로 된 상태명을 맵핑하여 명령어 비트 패턴 집합(22)으로 출력할 수 있다.
- [0056] 일 실시예로서, 상기 적어도 하나의 프로세서(140)는 상기 코드 생성기(130)가 상기 명령어에 대한 비트 벡터 시맨틱(21)을 트리 패턴 매칭을 사용하여 상기 세일 언어로 컴파일하도록 구성될 수 있으며, 그에 따라 명령어 비트 패턴 집합(22)을 출력할 수 있다.
- [0057] 추출한 명령어 비트 벡터 시맨틱(21)을 세일(sail) 코드로 표현하는 과정에서 본 발명에 따른 전자 장치(100)는 각 명령어에 대해 세 개의 코드 조각을 생성할 수 있다. 첫 번째 코드 조각은 명령어를 바이트 단위로 구문 분석하는데 사용하는 AST 맵핑 코드일 수 있다. 두 번째 코드 조각은 명령어 디코딩 과정에 대한 디코드 함수일 수 있다. 세 번째 코드 조각은 상기 명령어가 프로세서 상태를 변경하는 방법을 정의하는 실행 함수일 수 있다.
- [0058] 코드 생성기(130)는 명령어 비트 벡터 시맨틱(21)을 트리 패턴 매칭을 사용하여 세일 언어로 컴파일 할 수 있다. 예를 들어, 먼저 비트 벡터 표현식을 내부 중간언어로 변환하고 이를 하위 트리와 일치하는 세일 언어 표현식으로 바꿀 수 있다. 코드 생성기(130)는 트리를 순회하며 트리 템플릿과 일치하는 하위 트리를 만날 때마다 이를 대응하는 세일 언어 코드로 대체하여 세일 코드를 생성할 수 있다.
- [0059] 세일 언어로 명령어 비트 벡터 시맨틱(21)을 번역하기 위해 상태 요소 맵핑(13) 명세에서 RTL 프로세서 구조의 상태명에 대응하는 세일 상태 명을 제공할 수 있다.
- [0060] 또한, 일 실시예로서, 상기 적어도 하나의 프로세서(140)는 상기 세일 명령어 함수와 세일 스켈레톤 모델(200)을 조합하여 이물레이터(300)를 생성하도록 구성될 수 있다. 상기 세일 명령어 함수가 상기 명령어 비트 패턴 집합(22)일 수 있으며, 여기에 세일 스켈레톤 모델(200)을 적용하여 이물레이터(300)를 생성할 수 있다.
- [0061] 일 실시예로서, 상기 세일 스켈레톤 모델(200)은 상기 세일 언어로 작성된 Sail-RISC-V 모델을 기반으로 하는 모델일 수 있다.
- [0062] 한편, 도 5에 도시된 바와 같이, 본 발명의 일 실시예에 따른 기호 실행기(110)는 상기 RTL 프로세서의 하드웨어를 리셋하여 상기 RTL 프로세서의 파이프라인을 초기화하는 디버그 인터페이스(111), 상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 시뮬레이션 코어(112), 상기 외부 메모리(120)에 연결된 메모리 인터페이스(113)를 포함할 수 있다.
- [0063] 일 실시예로서, 상기 디버그 인터페이스(111)가 상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션하여 기호화된 명령어를 상기 RTL 프로세서에서 실행함으로써 상기 명령어의 초기화시 의미(Reset semantics)를 추출할 수 있다.
- [0064] 초기화는 명령어 추출을 위한 기호 시뮬레이션 과정에서 발생할 수 있는 상태폭발 문제를 극복하기 위한 것으로, 명령어의 의미에 큰 영향을 주지 않으며 파이프라인에 남아있는 명령어를 제거할 수 있도록 파이프라인을 적절히 초기화할 수 있다.
- [0065] 디버그 인터페이스(111)를 이용한 프로세서의 하드웨어 리셋을 프로세서의 파이프라인을 초기화하는 방법으로 활용할 수 있다.
- [0066] 디버그 인터페이스(111)를 통해 RTL 프로세서가 초기화된 후 의미 명세 추출 대상 명령어를 실행하고, 그 이후에는 NOP 명령어만 실행하는 방법을 이용해 프로세서의 상태 변화에 추출 대상 명령어만이 영향을 주도록 할 수 있다.
- [0067] 일 실시예로서, 상기 메모리 인터페이스(113)를 통해 상기 외부 메모리(120)가 상기 시뮬레이션 코어로 명령어 읽기(fetch) 접근에 대해 첫 사이클에는 시험대상 명령어를 반환하고, 나머지 사이클에는 NOP 명령어를 반환할 수 있다.

- [0068] 다시 말해, 기호 실행기(110)는 디버그 인터페이스(111)를 통해 RTL 프로세서를 초기화시킨 뒤 메모리 인터페이스(113)에 연결된 외부 메모리(120)를 통해 원하는 명령어를 실행시키고 메모리 접근을 관찰하며 메모리 읽기접근의 결과로 원하는 기호 또는 값을 제공할 수 있다.
- [0069] 또한, 상기 시뮬레이션 코어(112)가 상기NOP 명령어와 미 구현 명령어를 사용해 RTL 프로세서를 시뮬레이션하고 시뮬레이션 과정에서 상기 명령어의 상태 변화를 학습할 수 있다.
- [0070] 메모리 인터페이스(113)를 통해 시뮬레이션 코어(112)의 RTL 프로세서에 필요한 입력을 제공하거나 시뮬레이션 코어(112)의 출력 내용을 기록(Recording)할 수 있다.
- [0071] 그리고나서, 본 발명에 따른 전자 장치(100)는 RTL 프로세서(11)를 리셋한 뒤 추출 대상 기호 명령어 하나를 실행 시키고, 이후 NOP 명령어를 실행하도록 하며 그 실행 과정을 시뮬레이션할 수 있다. 시뮬레이션 결과 상태 및 비트 벡터 시맨틱을 획득할 수 있다.
- [0072] 도 6은 본 발명의 일 실시예에 따른 세일 함수를 나타낸 개략도이다.
- [0073] 도 6에 도시된 바와 같이, 세일 함수는 AST, 디코드, 실행 함수로 구분될 수 있다. 상이하게 정의되는 세 가지 코드 조각을 포함하는 것이다. 본 발명에 따른 전자 장치(100)는 입력 명령 패턴의 인스턴스를 AST와 디코드 함수로 변환할 수 있다. 그리고 기호 실행을 통해 생성한 비트 벡터의 의미를 실행 함수로 컴파일 할 수 있다.
- [0074] 수동으로 작성된 오픈 소스의 의미 명세서와 본 발명에 따른 전자 장치(100)를 비교한 결과 의미가 정확하게 일치하였음을 확인하였다.
- [0075] 도 7은 Sail-riscv 참조 모델과 본 발명에 따른 전자 장치(100)가 생성한 코드를 XOR 명령어를 이용하여 비교한 코딩을 도시한 도면으로, 도 7에 도시된 바와 같이, Sail-riscv 참조 모델은 본 발명에 따른 전자 장치(100)와 달리 동일한 코드 조각의 집합을 사용해서 여러 산술 명령을 처리하기 때문에 AST 매핑 문법은 다르지만, 실행 함수는 실제 시스템에서 여러 산술 명령의 동작을 설명할 수 있다.
- [0076] 도 7의 23번 라인을 참조하면, Sail-riscv 참조 모델의 RISC-V-XOR의 동작은 rs1 레지스터의 값과 r2 레지스터의 값을 XOR한 것과 동일하다. 본 발명에 따른 전자 장치(100)에서 추출한 실행 함수는 도 7의 49번 라인이며, X(rs1)은 rs1 레지스터값을 의미하기 때문에 두 의미 명세는 동일함을 알 수 있다.
- [0077] 이를 통해, 본 발명에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 전자 장치(100)는 기호 실행기(110)를 통해 도메인 특화 프로세서의 도메인 특화 명령어의 의미명세를 자동으로 추출할 수 있다.
- [0078] 한편, 도 8은 본 발명의 일 실시예에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출을 수행하는 방법의 플로우 차트를 도시한 것이다.
- [0079] 본 발명에 따른 도메인 특화 프로세서의 ISA 명세 자동 추출 방법(800)은, 레지스터 전송 레벨(RTL) 프로세서 및 적어도 하나의 명령어를 기호 실행하여 비트 벡터 시맨틱을 출력하는 기호 실행기와 상기 비트 벡터 시맨틱을 세일 언어로 컴파일하여 상기 명령어에 대한 ISA 명세를 생성하는 코드 생성기를 포함하는 전자 장치의 도메인 특화 프로세서의 ISA 명세 자동 추출 방법일 수 있다.
- [0080] 도 8에 도시된 바와 같이, 상기 기호 실행기로 도메인 특화 프로세서에 대응하는 상기 RTL 프로세서 및 기저장된 명령어 비트 패턴 집합 중 적어도 하나의 명령어를 입력하는 단계(S810), 상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계(S820), 출력한 상기 비트 벡터 시맨틱을 상기 코드 생성기에 입력하는 단계(S830) 및 상기 코드 생성기가 상기 비트 벡터 시맨틱을 세일(Sail) 언어로 컴파일하여 세일 명령어 함수를 출력하는 단계(S840)를 포함할 수 있다.
- [0081] 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계(S820)는 상기 기호 실행기가 출력한 명령어의 상태 변화와 상기 외부 메모리와의 상호작용 상태를 기반으로 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계를 포함할 수 있다.
- [0082] 상기 세일 명령어 함수를 출력하는 단계(S840)는 상기 코드 생성기가 상기 명령어에 대한 비트 벡터 시맨틱을 트리 패턴 매칭을 사용하여 상기 세일 언어로 컴파일하는 단계를 포함할 수 있다.
- [0083] 또한, 상기 세일 명령어 함수를 출력하는 단계(S840)는, 상기 코드 생성기가 기설정된 상태 요소 맵핑 명세를 고려하여 상기 RTL 프로세서의 상태명과 대응하는 상기 세일 언어의 상태명으로 컴파일하는 단계를 더 포함할 수 있다.

- [0084] 더하여, 상기 방법(800)은 상기 세일 명령어 함수와 세일 스켈레톤 모델을 조합하여 이플레이터를 생성하는 단계를 더 포함할 수 있다. 이때, 상기 세일 스켈레톤 모델은 상기 세일 언어로 작성된 Sail-RISC-V 모델을 기반으로 하는 모델일 수 있다.
- [0085] 상기 기호 실행기를 기호 실행하여 상기 명령어에 대한 비트 벡터 시맨틱을 출력하는 단계(S820)는, 상기 RTL 프로세서의 하드웨어를 리셋하여 상기 RTL 프로세서의 파이프라인을 초기화하는 단계 및 상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 단계를 포함할 수 있다.
- [0086] 또한, 상기 RTL 프로세서의 파이프라인을 초기화하는 단계는, 상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션하여 기호화된 명령어를 상기 RTL 프로세서에서 실행함으로써 상기 명령어의 초기화시 의미(Reset semantics)를 추출하는 단계를 포함할 수 있다.
- [0087] 더하여, 상기 RTL 프로세서의 파이프라인을 초기화하는 단계 후에, 상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 단계는, 상기 메모리 인터페이스를 통해 상기 외부 메모리가 상기 시뮬레이션 코어로 명령어 읽기(fetch) 접근에 대해 첫 사이클에는 시험대상 명령어를 반환하고, 나머지 사이클에는 NOP 명령어를 반환하는 단계를 더 포함할 수 있다.
- [0088] 상기 RTL 프로세서의 기호와 실제 값을 사용하여 시뮬레이션을 수행하는 단계는, 상기 NOP 명령어와 미 구현 명령어를 사용해 RTL 프로세서를 시뮬레이션하고 시뮬레이션 과정에서 상기 명령어의 상태 변화를 학습하는 단계를 포함할 수 있다.
- [0089] 이상 상술한 내용과 중복되는 내용은 명세서의 간략함을 위해 생략하기로 한다.
- [0090] 한편, 개시된 실시예들은 컴퓨터에 의해 실행 가능한 명령어를 저장하는 기록매체의 형태로 구현될 수 있다. 명령어는 프로그램 코드의 형태로 저장될 수 있으며, 프로세서에 의해 실행되었을 때, 프로그램 모듈을 생성하여 개시된 실시예들의 동작을 수행할 수 있다. 기록매체는 컴퓨터로 읽을 수 있는 기록매체로 구현될 수 있다.
- [0091] 컴퓨터가 읽을 수 있는 기록매체로는 컴퓨터에 의하여 해독될 수 있는 명령어가 저장된 모든 종류의 기록 매체를 포함한다. 예를 들어, ROM(Read Only Memory), RAM(Random Access Memory), 자기 테이프, 자기 디스크, 플래쉬 메모리, 광 데이터 저장장치 등이 있을 수 있다.
- [0092] 이상에서와 같이 첨부된 도면을 참조하여 개시된 실시예들을 설명하였다. 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 본 발명의 기술적 사상이나 필수적인 특징을 변경하지 않고도, 개시된 실시예들과 다른 형태로 본 발명이 실시될 수 있음을 이해할 것이다. 개시된 실시예들은 예시적인 것이며, 한정적으로 해석되어서는 안 된다.

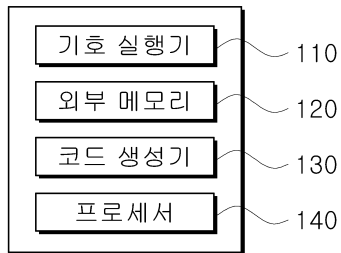
부호의 설명

- [0093]
- 110: 기호 실행기
 - 120: 외부 메모리
 - 130: 코드 생성기
 - 140: 프로세서
 - 200: 세일 스켈레톤 모델
 - 300: 이플레이터
 - 111: 디버그 인터페이스
 - 112: 시뮬레이션 코어
 - 113: 메모리 인터페이스

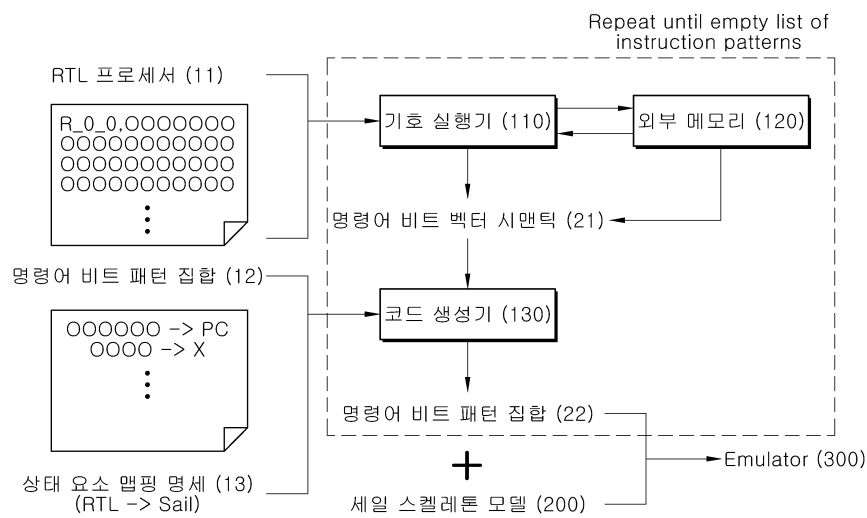
도면

도면1

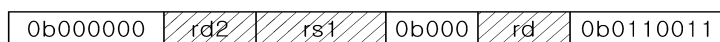
100



도면2



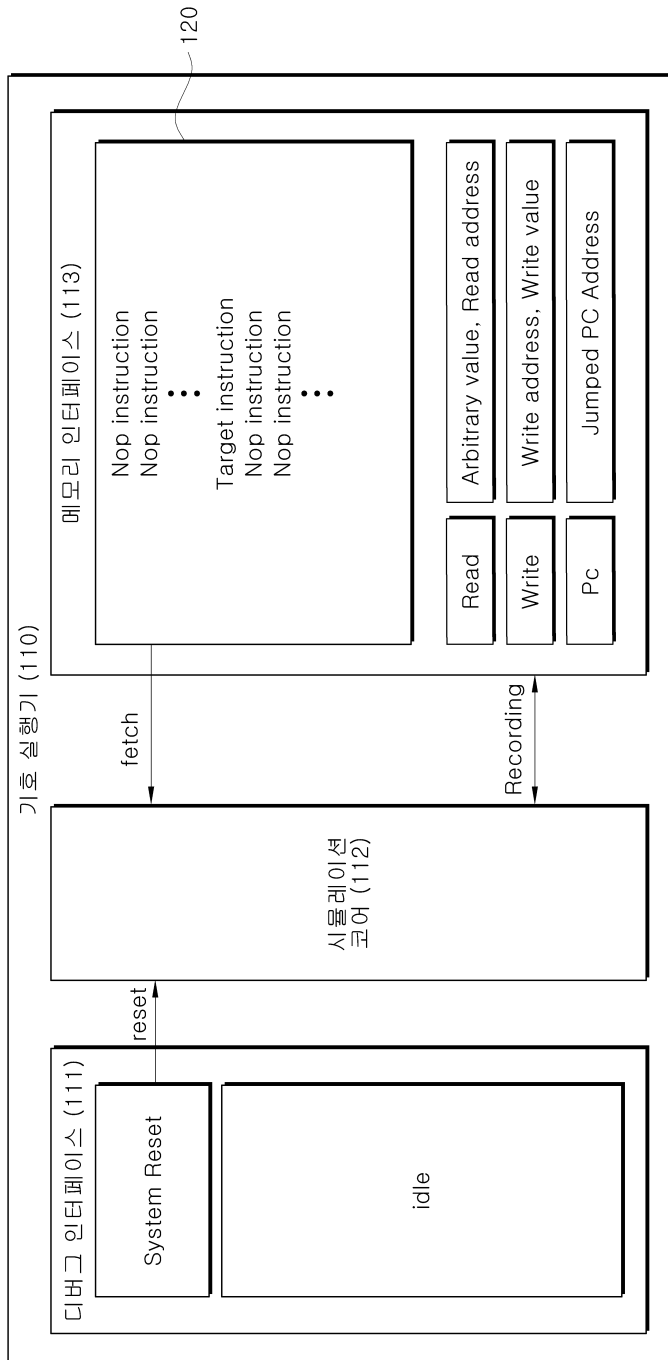
도면3



도면4

(itc (and (= rs1_l #b1) (= rs1_h #x7))	(Condition Expression)
((_extract 1 0) regfile@14)	(True Expression)
#b00)	(False Expression)

도면5



도면6

```

Abstract syntax type
union clause ast = ITYPE : (bits(12), regbits, regbits, iop)
Decode function
function clause decode imm : bits(12) @ rs1 : regbits @ 0b000 @ rd : regbits @ 0b0010011
  = Some(ITYPE(imm, rs1, rd, RISC_V_ADDI))
Execute function
function clause execute (ITYPE (imm, rs1, rd, RISC_V_ADDI)) =
  let rs1_val = X(rs1) in
  let imm_ext : xlenbits = EXTS(imm) in
  let result = rs1_val + imm_ext in
  X(rd) = result
  
```

Sail execute function header

Sail execute function body

도면7

```

1  =====Sail-riscv reference model: =====
2  ----- AST -----
3  union clause ast = RTYPE : (regidx, regidx, regidx, rop)
4
5  ----- Decode function -----
6  mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_SLTU) <-> 0b0000000 @ rs2 @ rs1 @ 0b011 @ rd @ 0b0110011
7  mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_AND) <-> 0b0000000 @ rs2 @ rs1 @ 0b111 @ rd @ 0b0110011
8  mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_OR) <-> 0b0000000 @ rs2 @ rs1 @ 0b110 @ rd @ 0b0110011
9  mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_XOR) <-> 0b0000000 @ rs2 @ rs1 @ 0b100 @ rd @ 0b0110011
10 mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_SLL) <-> 0b0000000 @ rs2 @ rs1 @ 0b001 @ rd @ 0b0110011
11 mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_SRL) <-> 0b0000000 @ rs2 @ rs1 @ 0b101 @ rd @ 0b0110011
12 mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_SUB) <-> 0b0100000 @ rs2 @ rs1 @ 0b000 @ rd @ 0b0110011
13 mapping clause encdec = RTYPE(rs2, rs1, rd, RISC_V_SRA) <-> 0b0100000 @ rs2 @ rs1 @ 0b101 @ rd @ 0b0110011
14 ----- Execute function -----
15 function clause execute (RTYPE(rs2, rs1, rd, op)) = {
16   let rs1_val = X(rs1);
17   let rs2_val = X(rs2);
18   let result : xlenbits = match op {
19     RISC_V_SLT => EXTZ(bool_to_bits(rs1_val <_s rs2_val)),
20     RISC_V_SLTU => EXTZ(bool_to_bits(rs1_val <_u rs2_val)),
21     RISC_V_AND => rs1_val & rs2_val,
22     RISC_V_OR => rs1_val | rs2_val,
23     RISC_V_XOR => rs1_val ^ rs2_val,
24     RISC_V_SLL => if sizeof(xlen) == 32
25                   then rs1_val << (rs2_val[4..0])
26                   else rs1_val << (rs2_val[5..0]),
27     RISC_V_SRL => if sizeof(xlen) == 32
28                   then rs1_val >> (rs2_val[4..0])
29                   else rs1_val >> (rs2_val[5..0]),
30     RISC_V_SUB => rs1_val - rs2_val,
31     RISC_V_SRA => if sizeof(xlen) == 32
32                   then shift_right_arith32(rs1_val, rs2_val[4..0])
33                   else shift_right_arith64(rs1_val, rs2_val[5..0])
34   };
35   X(rd) = result;
36   RETIRE_SUCCESS
37 }
38
39 =====Codgen Result:=====
40 ----- AST -----
41 union clause ast = R_0_4 : (bits(5), bits(5), bits(5))
42 ----- Decode function -----
43 mapping clause encdec = R_0_4(rs2, rs1, rd) <-> 0b0000000 @ rs2 @ rs1 @ 0b100 @ rd @ 0b0110011
44 mapping clause assembly = R_0_4(rs2, rs1, rd) <-> "R_0_4" ^ spc() ^ reg_name(rd) ^ sep() ^
45   reg_name(rs1) ^ sep() ^ reg_name(rs2)
46 ----- Execute function -----
47 function clause execute R_0_4(rs2,rs1,rd) = {
48   X(rd) = (X(rs1) ^ X(rs2));
49   RETIRE_SUCCESS
50 }
51

```

도면8

