(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년04월05일
(11) 등록번호 10-2236700
(24) 등록일자 2021년03월31일

(51) 국제특허분류(Int. Cl.)
G06F 8/41 (2018.01)
(52) CPC특허분류
G06F 8/433 (2013.01)
G06F 8/45 (2013.01)
(21) 출원번호 10-2019-0147771
(22) 출원일자 2019년11월18일
심사청구일자 2019년11월18일
(56) 선행기술조사문헌
KR1020090017400 A
KR100176088 B1
Henning Stubbe, "P4 Compiler & Interpreter: A Survey", Seminars FI / IITM WS 16/17, Network Architectures and Services, Technische Universitat Munchen, p47-52 (2017. 05. 31) 1부.*
Lavanya Jose 등, "Compiling Packet Programs toReconfigurable Switches", the Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI'15), Oakland, CA, USA, p103-115, (2015. 05. 16) 1부*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
연세대학교 산학협력단
서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)
(72) 발명자
김한준
서울특별시 서대문구 연세로 50, 연세대학교 제3공학관 C415호(신촌동)
송승빈
서울특별시 서대문구 연세로 50, 연세대학교 제3공학관 C407호(신촌동)
(74) 대리인
민영준

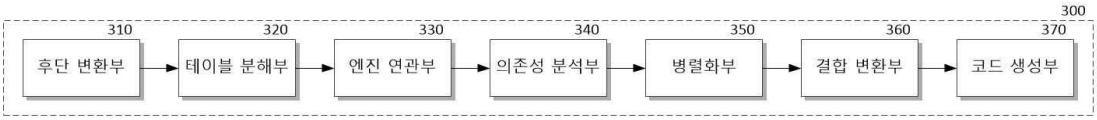
전체 청구항 수 : 총 16 항 심사관 : 지정훈

(54) 발명의 명칭 **패킷 프로세싱 프로그램 병렬화 컴파일 장치 및 방법**

(57) 요 약

본 발명은 컴파일 대상이 되는 소스 코드를 기지정된 방식에 따라 전단 중간 표현으로 변환하는 전단부 및 전단 중간 표현을 분석하여 동작 명령을 나타내는 다수의 테이블을 구분하고, 다수의 테이블 각각을 명령 실행의 대상인지 여부를 판별하기 위한 키가 기술되는 매치 영역과 수행해야 하는 동작이 기술된 액션 영역으로 구분한 후, 매치 영역과 액션 영역을 프로세서의 동작 기본 단위인 베이직 블록 단위로 분해하고, 분해된 베이직 블록 사이의 의존성을 분석하여 병렬화 및 결합하여 소스 코드에 대응하는 패킷 프로세싱 프로그램 코드를 생성하는 후단부를 포함하여, 고속 프로세싱이 가능한 패킷 프로세싱 프로그램을 생성할 수 있고, 다양한 제약 조건에서도 최적화된 패킷 프로세싱 프로그램을 생성할 수 있는 컴파일 장치 및 방법을 제공할 수 있다.

대 표 도



이 발명을 지원한 국가연구개발사업

과제고유번호	1711080849
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원(한국연구재단부설)
연구사업명	정보통신방송연구개발사업
연구과제명	[주관/고려대학교] 멀티 서비스를 지원하는 프로그래머블 스위치 제어 기술 개발 (3/4)
기 여 율	1/1
과제수행기관명	고려대학교 산학협력단
연구기간	2019.01.01 ~ 2019.12.31

명 세 서

청구범위

청구항 1

컴파일 대상이 되는 소스 코드를 기지정된 방식에 따라 전단 중간 표현으로 변환하는 전단부; 및

상기 전단 중간 표현을 분석하여 동작 명령을 나타내는 다수의 테이블을 구분하고, 상기 다수의 테이블 각각을 명령 실행의 대상인지 여부를 판별하기 위한 키가 기술되는 매치 영역과 수행해야 하는 동작이 기술된 액션 영역으로 구분한 후, 상기 매치 영역과 상기 액션 영역을 프로세서의 동작 기본 단위인 베이직 블록 단위로 분해하고, 분해된 베이직 블록 사이의 의존성을 분석하여 병렬화 및 결합하여 소스 코드에 대응하는 패킷 프로세싱 프로그램 코드를 생성하는 후단부를 포함하되,

상기 후단부는

상기 전단 중간 표현을 기지정된 방식으로 후단 중간 표현으로 변환하고, 다수의 테이블을 구분하는 후단 변환부;

상기 후단 중간 표현에서 다수의 테이블 각각을 상기 매치 영역과 상기 액션 영역을 구분하여 베이직 블록 단위인 다수의 엔진으로 분해하는 테이블 분해부;

상기 다수의 엔진 사이의 연관성을 분석하는 엔진 연관부;

분석된 연관성에 기초하여 다수의 엔진 사이의 의존성을 분석하는 의존성 분석부;

분석된 의존성에 기초하여 상기 다수의 엔진 중 의존성이 없는 엔진으로부터 기지정된 방식으로 병렬로 배치하는 병렬화부;

병렬로 배치된 엔진들 중 기지정된 형식의 엔진들을 병합하는 결합 변환부; 및

배치 및 결합된 다수의 엔진의 배치 순서에 기반하여 기지정된 형식의 패킷 프로그램 코드를 생성하는 코드 생성부를 포함하고,

상기 병렬화부는

다수의 테이블에서 분해된 다수의 엔진 중 의존성이 없는 엔진을 탐색하여 이전 엔진이 배치된 시간 구간의 다음 시간 구간에 병렬로 배치하고, 배치된 엔진들에 대한 의존성을 제거하며, 모든 분해된 모든 엔진이 배치될 때까지 반복하는 패킷 프로세싱 프로그램 컴파일 장치.

청구항 2

삭제

청구항 3

제1 항에 있어서, 상기 테이블 분해부는

상기 매치 영역을 상기 매치 영역에 기술된 키를 기반으로 컨트롤 평면 상의 대응하는 값에 대한 요청을 전송하는 동작을 수행하는 제1 튜플 엔진과 컨트롤 평면에서 키에 대응하는 값을 인가받아 매치 여부를 판정하는 룩업 엔진(Lookup)으로 분해하고,

상기 액션 영역을 상기 룩업 엔진의 매치 판정에 따른 동작을 수행하는 제2 튜플 엔진으로 분해하는 패킷 프로세싱 프로그램 컴파일 장치.

청구항 4

제3 항에 있어서, 상기 테이블 분해부는

상기 테이블의 액션 영역이 상기 키가 매치되면 실행되는 제1 액션 영역과 상기 키가 매치되지 않으면 실행되는

제2 액션 영역으로 구분되면, 상기 제1 액션 영역에 대응하는 제2 튜플 엔진과 상기 제2 액션 영역에 대응하는 제3 튜플 엔진으로 분해하는 패킷 프로세싱 프로그램 컴파일 장치.

청구항 5

제4 항에 있어서, 상기 엔진 연관부는

상기 룩업 엔진이 동일 테이블 내의 상기 제1 튜플 엔진에 종속되도록 연관시키고, 상기 제2 또는 제3 튜플 엔진은 동일 테이블 내의 상기 룩업 엔진에 종속되도록 연관시키는 패킷 프로세싱 프로그램 컴파일 장치.

청구항 6

제5 항에 있어서, 상기 엔진 연관부는

상기 룩업 엔진에 조건 구문이 포함되면, 조건 구문에 따라 다른 테이블의 상기 제2 또는 제3 튜플 엔진이 종속되도록 연관시키고, 상기 제2 또는 제3 튜플 엔진 각각의 데이터 종속성을 분석하여 다른 테이블의 상기 제2 또는 제3 튜플 엔진과의 종속성을 연관시키는 패킷 프로세싱 프로그램 컴파일 장치.

청구항 7

삭제

청구항 8

제1 항에 있어서, 상기 병렬화부는

상기 패킷 프로세싱 프로그램 코드가 적용되는 타겟의 하드웨어적 파이프 라인의 개수 이하 개수의 엔진을 동일 시간 구간에 병렬로 배치하는 패킷 프로세싱 프로그램 컴파일 장치.

청구항 9

제1 항에 있어서, 상기 결합 변환부는

동일 시간 구간에 병렬로 배치된 다수의 엔진 중 튜플 엔진들을 결합하는 패킷 프로세싱 프로그램 컴파일 장치.

청구항 10

제1 항에 있어서, 상기 소스 코드는 P4 언어로 작성된 코드이고, 상기 전단 중간 표현은 P4 언어를 변환한 P4 IR(Intermediate Representation)이며, 상기 패킷 프로세싱 프로그램 코드는 PX 언어로 생성되며, 상기 후단 중간 표현은 PX 언어의 중간 표현 형태인 PX IR인 패킷 프로세싱 프로그램 컴파일 장치.

청구항 11

컴파일 대상이 되는 소스 코드를 기지정된 방식에 따라 전단 중간 표현으로 변환하는 단계; 및

상기 전단 중간 표현을 분석하여 동작 명령을 나타내는 다수의 테이블을 구분하고, 상기 다수의 테이블 각각을 명령 실행의 대상인지 여부를 판별하기 위한 키가 기술되는 매치 영역과 수행해야 하는 동작이 기술된 액션 영역으로 구분한 후, 상기 매치 영역과 상기 액션 영역을 프로세서의 동작 기본 단위인 베이직 블록 단위로 분해하고, 분해된 베이직 블록 사이의 의존성을 분석하여 병렬화 및 결합하여 소스 코드에 대응하는 패킷 프로세싱 프로그램 코드를 생성하는 단계를 포함하되,

상기 패킷 프로세싱 프로그램 코드를 생성하는 단계는

상기 전단 중간 표현을 기지정된 방식으로 후단 중간 표현으로 변환하고, 다수의 테이블을 구분하는 단계;

상기 후단 중간 표현에서 다수의 테이블 각각을 상기 매치 영역과 상기 액션 영역을 구분하여 베이직 블록 단위인 다수의 엔진으로 분해하는 단계;

상기 다수의 엔진 사이의 연관성을 분석하는 단계;

분석된 연관성에 기초하여 다수의 엔진 사이의 의존성을 분석하는 단계;

분석된 의존성에 기초하여 상기 다수의 엔진 중 의존성이 없는 엔진으로부터 기지정된 방식으로 병렬로 배치하

는 단계;

병렬로 배치된 엔진들 중 기지정된 형식의 엔진들을 병합하는 단계; 및

배치 및 결합된 다수의 엔진의 배치 순서에 기반하여 기지정된 형식의 패킷 프로그램 코드를 생성하는 단계를 포함하고,

상기 병렬로 배치하는 단계는

다수의 테이블에서 분해된 다수의 엔진 중 의존성이 없는 엔진을 탐색하는 단계;

탐색된 엔진을 이전 엔진이 배치된 시간 구간의 다음 시간 구간에 병렬로 배치하는 단계;

배치된 엔진들에 대한 의존성을 제거하는 단계; 및

분해된 모든 엔진이 배치되었는지 판별하고, 배치되지 않은 엔진이 존재하는 것으로 판단되면, 다시 의존성이 없는 엔진을 탐색하는 단계를 포함하는 패킷 프로세싱 프로그램 컴파일 방법.

청구항 12

삭제

청구항 13

제11 항에 있어서, 상기 분해하는 단계는

상기 매치 영역을 상기 매치 영역에 기술된 키를 기반으로 컨트롤 평면 상의 대응하는 값에 대한 요청을 전송하는 동작을 수행하는 제1 튜플 엔진과 컨트롤 평면에서 키에 대응하는 값을 인가받아 매치 여부를 판정하는 룩업 엔진(Lookup)으로 분해하는 단계; 및

상기 액션 영역을 상기 룩업 엔진의 매치 판정에 따른 동작을 수행하는 제2 튜플 엔진으로 분해하는 단계를 포함하는 패킷 프로세싱 프로그램 컴파일 방법.

청구항 14

제13 항에 있어서, 상기 분해하는 단계는

상기 테이블의 액션 영역이 상기 키가 매치되면 실행되는 제1 액션 영역과 상기 키가 매치되지 않으면 실행되는 제2 액션 영역으로 구분되면,

상기 액션 영역을 상기 제1 액션 영역에 대응하는 제2 튜플 엔진과 상기 제2 액션 영역에 대응하는 제3 튜플 엔진으로 분해하는 패킷 프로세싱 프로그램 컴파일 방법.

청구항 15

제14 항에 있어서, 상기 연관성을 분석하는 단계는

상기 룩업 엔진이 동일 테이블 내의 상기 제1 튜플 엔진에 종속되도록 연관시키는 단계; 및

상기 제2 또는 제3 튜플 엔진은 동일 테이블 내의 상기 룩업 엔진에 종속되도록 연관시키는 단계를 포함하는 패킷 프로세싱 프로그램 컴파일 방법.

청구항 16

제15 항에 있어서, 상기 연관성을 분석하는 단계는

상기 룩업 엔진에 조건 구문이 포함되면, 조건 구문에 따라 다른 테이블의 상기 제2 또는 제3 튜플 엔진이 종속되도록 연관시키는 단계; 및

상기 제2 또는 제3 튜플 엔진 각각의 데이터 종속성을 분석하여 다른 테이블의 상기 제2 또는 제3 튜플 엔진과의 종속성을 연관시키는 단계를 더 포함하는 패킷 프로세싱 프로그램 컴파일 방법.

청구항 17

삭제

청구항 18

제11 항에 있어서, 상기 다음 시간 구간에 병렬로 배치하는 단계는

상기 패킷 프로세싱 프로그램 코드가 적용되는 타겟의 하드웨어적 파이프 라인의 개수 이하 개수의 엔진을 동일 시간 구간에 병렬로 배치하는 패킷 프로세싱 프로그램 컴파일 방법.

청구항 19

제11 항에 있어서, 상기 병합하는 단계는

동일 시간 구간에 병렬로 배치된 다수의 엔진 중 튜플 엔진들을 결합하는 패킷 프로세싱 프로그램 컴파일 방법.

청구항 20

제11 항에 있어서, 상기 소스 코드는 P4 언어로 작성된 코드이고, 상기 전단 중간 표현은 P4 언어를 변환한 P4 IR(Intermediate Representation)이며, 상기 패킷 프로세싱 프로그램 코드는 PX 언어로 생성되며, 상기 후단 중간 표현은 PX 언어의 중간 표현 형태인 PX IR인 패킷 프로세싱 프로그램 컴파일 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 패킷 프로세싱 프로그램 컴파일 장치 및 방법에 관한 것으로, P4 언어로 작성된 패킷 프로세싱 프로그램 코드를 병렬화하여 컴파일하는 패킷 프로세싱 프로그램 컴파일 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] 정보 통신 기술의 발전으로 인해 네트워크 트래픽이 폭발적으로 증가됨에 따라 네트워크에 요구되는 데이터 처리 용량 또한 급격하게 증대되고 있으며, 이에 관리되어야 하는 네트워크 장비의 숫자 또한 크게 증가되고 있다. 그러나 기존의 레거시(Legay) 방식에서는 네트워크 상의 수많은 네트워크 장비들 각각을 개별적으로 관리해야 함에 따라 네트워크 관리가 어렵다는 문제가 있으며 이로 인해 네트워크의 확장, 업데이트, 설정 변경 등이 용이하지 않다는 한계가 있다.

[0003] 상기한 기존 네트워크의 한계를 극복하기 위해, 물리 네트워크 인프라를 공용으로 이용하되, 이용 목적에 따라 독립된 가상의 네트워크 환경을 생성하여 제공하는 네트워크 가상화(network virtualization) 개념이 도입되었다. 특히 최근에는 소프트웨어 프로그래밍을 통해 네트워크를 제어할 수 있도록 하는 소프트웨어 정의 네트워킹(Software Defined networking: 이하 SDN)에 대한 연구가 활발하게 진행되고 있다. SDN은 네트워크 제어 기능이 스위치나 라우터 등의 물리적 네트워크와 분리되어, 네트워크 자체를 소프트웨어적으로 프로그램 가능하게 함으로써 네트워크를 유연하게 구성할 수 있으며, 변경이 용이하다는 장점이 있다. 즉 SDN 에서는 네트워크 관리자가 다수의 네트워크 스위치를 개별적으로 관리하지 않고, 소프트웨어적인 프로그래밍을 통해 다수의 네트워크 스위치의 제어 설정을 용이하게 변경할 수 있다.

[0004] 다만 상기한 바와 같이 기존의 네트워크 스위치는 일부 조정 가능한 매개 변수가 있지만 기본적으로는 다시 프로그래밍될 수 없었으며, 스위치 벤더에서 제공하는 기능을 수행할 수밖에 없었다. 그러나 현재는 프로그래머블 스위치의 보급이 확대되고 있다. 프로그래머블 스위치는 패킷 프로세싱 프로그램을 기반으로 동작하며, 패킷 프로세싱 프로그램에 따른 기능을 수행할 수 있도록 용이하게 수정될 수 있다. 특히 사용자는 패킷 프로세싱 프로그램을 작성하고, 작성된 패킷 프로세싱 프로그램을 FPGA(Field Progammable Gate Array)로 구성하여 자신의 사용 용도에 따른 맞춤형 스위치 칩을 제조함으로써, 최적화된 스위치를 획득할 수 있다.

[0005] 또한 프로그래머블 스위치는 패킷 처리 파이프 라인을 포함하여 병렬 처리가 가능하도록 구성될 수 있다. 이를 통해 프로그래머블 스위치는 고정된 기능을 수행하는 기존의 네트워크 스위치와 비교하여도 뒤지지 않는 고성능을 나타낼 수 있다.

[0006] 한편 SDN에서는 네트워크 스위치를 제어하기 위한 제어 명령이 포함되는 제어 평면(Control Plane)과 제어 명령

에 따라 네트워크 패킷을 프로세싱하는 데이터 평면(Data Plane)을 구분한다. 기존의 SDN 기술에서는 제어 평면에 대한 프로그래밍은 가능하였으나, 데이터 평면에 대한 프로그래밍 기능은 지원하지 않았다. 따라서 기존에는 SDN을 이용하더라도 네트워크 스위치에 새로운 기능을 추가하거나 기존 기능을 변경 및 제거하기 어렵다는 한계가 있었다.

[0007] 이러한 한계를 극복하기 위해, 데이터 평면의 프로그래머블 아키텍처를 정의하여 데이터 평면을 프로그램 가능하도록 함으로써 네트워크 스위치의 기능을 규정할 수 있도록 하는 프로그래밍 언어인 P4 언어가 제안되었다. 패킷 프로세싱 프로그램을 작성할 수 있는 P4 언어가 제안됨에 따라 네트워크 관리자는 P4 언어를 이용하여 소스 코드를 작성하여 용이하게 네트워크 스위치의 기능을 추가, 확장, 변경 및 제거할 수 있게 되었다.

[0008] 다만 P4 언어로 작성된 패킷 프로세싱 프로그램의 소스 코드는 네트워크 스위치에 그대로 적용될 수 없으므로, 컴파일 장치에 의해 실행 가능한 프로그램 코드로 컴파일되야만 하드웨어인 프로그래머블 스위치에 합성될 수 있다. 또한 작성된 소스 코드의 정상 여부를 판정하기 위해서도 패킷 프로세싱 프로그램으로 컴파일 되어 사전에 검증될 필요가 있다. 이때, 컴파일 장치는 적용 대상이 되는 스위치의 하드웨어적 제약 조건을 고려하여 패킷 프로세싱 프로그램이 효율적으로 동작할 수 있도록 컴파일을 수행해야 한다.

선행기술문헌

비특허문헌

[0009] (비특허문헌 0001) Compiling Packet Programs to Reconfigurable Switches, Networked Systems Design and Implementation(NSDI '15)(2015. 05. 06)

발명의 내용

해결하려는 과제

[0010] 본 발명의 목적은 P4 언어로 작성된 패킷 프로세싱 프로그램의 소스 코드를 컴파일할 때, 컴파일된 패킷 프로세싱 프로그램의 저지연성을 개선할 수 있는 패킷 프로세싱 프로그램 컴파일 장치 및 방법을 제공하는데 있다.

[0011] 본 발명의 다른 목적은 패킷 프로세싱 프로그램의 동작을 베이직 블록 단위로 병렬화하여 고성능의 패킷 프로세싱 프로그램을 지원할 수 있는 패킷 프로세싱 프로그램 컴파일 장치 및 방법을 제공하는데 있다.

과제의 해결 수단

[0012] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따른 패킷 프로세싱 프로그램 컴파일 장치는 컴파일 대상이 되는 소스 코드를 기지정된 방식에 따라 전단 중간 표현으로 변환하는 전단부; 및 상기 전단 중간 표현을 분석하여 동작 명령을 나타내는 다수의 테이블을 구분하고, 상기 다수의 테이블 각각을 명령 실행의 대상인지 여부를 판별하기 위한 키가 기술되는 매치 영역과 수행해야 하는 동작이 기술된 액션 영역으로 구분한 후, 상기 매치 영역과 상기 액션 영역을 프로세서의 동작 기본 단위인 베이직 블록 단위로 분해하고, 분해된 베이직 블록 사이의 의존성을 분석하여 병렬화 및 결합하여 소스 코드에 대응하는 패킷 프로세싱 프로그램 코드를 생성하는 후단부를 포함한다.

[0013] 상기 후단부는 상기 전단 중간 표현을 기지정된 방식으로 후단 중간 표현으로 변환하고, 다수의 테이블을 구분하는 후단 변환부; 상기 후단 중간 표현에서 다수의 테이블 각각을 상기 매치 영역과 상기 액션 영역을 구분하여 베이직 블록 단위인 다수의 엔진으로 분해하는 테이블 분해부; 상기 다수의 엔진 사이의 연관성을 분석하는 엔진 연관부; 분석된 연관성에 기초하여 다수의 엔진 사이의 의존성을 분석하는 의존성 분석부; 분석된 의존성에 기초하여 상기 다수의 엔진 중 의존성이 없는 엔진으로부터 기지정된 방식으로 병렬로 배치하는 병렬화부; 병렬로 배치된 엔진들 중 기지정된 형식의 엔진들을 병합하는 결합 변환부; 및 배치 및 결합된 다수의 엔진의 배치 순서에 기반하여 기지정된 형식의 패킷 프로그램 코드를 생성하는 코드 생성부를 포함할 수 있다.

[0014] 상기 테이블 분해부는 상기 매치 영역을 상기 매치 영역에 기술된 키를 기반으로 컨트롤 평면 상의 대응하는 값에 대한 요청을 전송하는 동작을 수행하는 제1 튜플 엔진과 컨트롤 평면에서 키에 대응하는 값을 인가받아 매치 여부를 판정하는 룩업 엔진(Lookup)으로 분해하고, 상기 액션 영역을 상기 룩업 엔진의 매치 판정에 따른 동작을 수행하는 제2 튜플 엔진으로 분해할 수 있다.

[0015] 상기 테이블 분해부는 상기 테이블의 액션 영역이 상기 키가 매치되면 실행되는 제1 액션 영역과 상기 키가 매치되지 않으면 실행되는 제2 액션 영역으로 구분되면, 상기 제1 액션 영역에 대응하는 제2 튜플 엔진과 상기 제2 액션 영역에 대응하는 제3 튜플 엔진으로 분해할 수 있다.

[0016] 상기 엔진 연관부는 상기 룩업 엔진이 동일 테이블 내의 상기 제1 튜플 엔진에 종속되도록 연관시키고, 상기 제2 또는 제3 튜플 엔진은 동일 테이블 내의 상기 룩업 엔진에 종속되도록 연관시킬 수 있다.

[0017] 상기 엔진 연관부는 상기 룩업 엔진에 조건 구문이 포함되면, 조건 구문에 따라 다른 테이블의 상기 제2 또는 제3 튜플 엔진이 종속되도록 연관시키고, 상기 제2 또는 제3 튜플 엔진 각각의 데이터 종속성을 분석하여 다른 테이블의 상기 제2 또는 제3 튜플 엔진과의 종속성을 연관시킬 수 있다.

[0018] 상기 병렬화부는 다수의 테이블에서 분해된 다수의 엔진 중 의존성이 없는 엔진을 탐색하여 이전 엔진이 배치된 시간 구간의 다음 시간 구간에 병렬로 배치하고, 배치된 엔진들에 대한 의존성을 제거하며, 모든 분해된 모든 엔진이 배치될 때까지 반복할 수 있다.

[0019] 상기 병렬화부는 상기 패킷 프로세싱 프로그램 코드가 적용되는 타겟의 하드웨어적 파이프 라인의 개수 이하 개수의 엔진을 동일 시간 구간에 병렬로 배치할 수 있다.

[0020] 상기 결합 변환부는 동일 시간 구간에 병렬로 배치된 다수의 엔진 중 튜플 엔진들을 결합할 수 있다.

[0021] 상기 소스 코드는 P4 언어로 작성된 코드이고, 상기 전단 중간 표현은 P4 언어를 변환한 P4 IR(Intermediate Representation)이며, 상기 패킷 프로세싱 프로그램 코드는 PX 언어로 생성되며, 상기 후단 중간 표현은 PX 언어의 중간 표현 형태인 PX IR일 수 있다.

[0022] 상기 목적을 달성하기 위한 본 발명의 다른 실시예에 따른 패킷 프로세싱 프로그램 컴파일 방법은 컴파일 대상이 되는 소스 코드를 기지정된 방식에 따라 전단 중간 표현으로 변환하는 단계; 및 상기 전단 중간 표현을 분석하여 동작 명령을 나타내는 다수의 테이블을 구분하고, 상기 다수의 테이블 각각을 명령 실행의 대상인지 여부를 판별하기 위한 키가 기술되는 매치 영역과 수행해야 하는 동작이 기술된 액션 영역으로 구분한 후, 상기 매치 영역과 상기 액션 영역을 프로세서의 동작 기본 단위인 베이직 블록 단위로 분해하고, 분해된 베이직 블록 사이의 의존성을 분석하여 병렬화 및 결합하여 소스 코드에 대응하는 패킷 프로세싱 프로그램 코드를 생성하는 단계를 포함한다.

발명의 효과

[0023] 따라서, 본 발명의 실시예에 따른 패킷 프로세싱 프로그램 컴파일 장치 및 방법은 소스 코드에 포함된 다수의 테이블을 매치 영역과 액션 영역으로 구분하고, 구분된 매치 영역에 대응하는 튜플 엔진과 룩업 엔진, 그리고 액션 영역에 대응하는 튜플 엔진들 사이의 의존성을 분석하여 병렬화함으로써, 고속 프로세싱이 가능한 패킷 프로세싱 프로그램을 생성할 수 있다. 또한 하드웨어로 구현되는 네트워크 스위치의 제약 사항을 반영하여 병렬화 개수를 조절할 수 있으므로, 다양한 제약 조건에서도 최적화된 패킷 프로세싱 프로그램을 생성할 수 있다.

도면의 간단한 설명

[0024] 도 1은 본 발명의 일 실시예에 따른 패킷 프로세싱 프로그램 컴파일 장치의 개략적 구조를 나타낸다.

도 2는 도 1의 후단부의 상세 구조를 나타낸다.

도 3은 소스 코드에서 테이블의 구조를 개념적으로 나타낸 도면이다.

도 4는 테이블을 기본 블록 단위로 구분하는 개념을 나타낸 도면이다.

도 5는 조건 구문에 의해 분기되는 두개의 테이블 사이의 관계를 시각적으로 나타낸 도면이다.

도 6은 두개의 테이블에서 매치 영역 및 액션 영역의 엔진들 사이의 의존성을 시각적으로 나타낸 도면이다.

도 7은 도 2의 병렬화부가 엔진들 사이의 의존성에 따른 병렬화를 수행한 결과를 나타낸다.

도 8은 도 2의 결합 변환부가 병렬로 배치된 튜플 엔진을 결합한 결과를 시각적으로 나타낸다.

도 9는 본 발명의 일 실시예에 따른 패킷 프로세싱 프로그램 컴파일 방법을 나타낸다.

도 10은 도 9의 병렬화 단계를 상세하게 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0025] 본 발명과 본 발명의 동작상의 이점 및 본 발명의 실시에 의하여 달성되는 목적을 충분히 이해하기 위해서는 본 발명의 바람직한 실시예를 예시하는 첨부 도면 및 첨부 도면에 기재된 내용을 참조하여야만 한다.

[0026] 이하, 첨부한 도면을 참조하여 본 발명의 바람직한 실시예를 설명함으로써, 본 발명을 상세히 설명한다. 그러나, 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며, 설명하는 실시예에 한정되는 것이 아니다. 그리고, 본 발명을 명확하게 설명하기 위하여 설명과 관계없는 부분은 생략되며, 도면의 동일한 참조부호는 동일한 부재임을 나타낸다.

[0027] 명세서 전체에서, 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. 또한, 명세서에 기재된 "...부", "...기", "모듈", "블록" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어나 소프트웨어 또는 하드웨어 및 소프트웨어의 결합으로 구현될 수 있다.

[0028] 도 1은 본 발명의 일 실시예에 따른 패킷 프로세싱 프로그램 컴파일 장치의 개략적 구조를 나타낸다.

[0029] 도 1을 참조하면, 본 실시예에 따른 패킷 프로세싱 프로그램 컴파일 장치는 소스 코드 획득부(100), 전단부(200) 후단부(300) 및 프로그램 출력부(400)를 포함할 수 있다.

[0030] 소스 코드 획득부(100)는 컴파일되어야 하는 패킷 프로세싱 프로그램의 소스 코드를 획득한다. 여기서 소스 코드는 P4 언어로 작성된 코드일 수 있다. 소스 코드 획득부(100)는 소스 코드가 미리 저장된 메모리 모듈 또는 외부로부터 소스 코드를 인가받는 통신 모듈 등으로 구현될 수 있다.

[0031] 전단부(200)는 소스 코드 획득부(100)에서 획득된 소스 코드를 인가받아 기지정된 방식에 따라 중간 표현(Intermediate Representation: 이하 IR)으로 변환한다. 소스 코드가 P4 언어로 작성된 경우 전단부(200)는 소스 코드를 P4 중간 언어 형식인 P4 IR로 변환할 수 있다.

[0032] 전단부(200)는 전단 변환부(210) 및 타겟 적응 변환부(220)를 포함할 수 있다. 전단 변환부(210)는 인가된 소스 코드를 분석(parse)하고, 타입 체크(type-check), 상수 폴딩(constant folding) 및 데드 코드 제거(dead code elimination)와 같은 일련의 동작을 수행하여 소스 코드를 기지정된 IR 형식으로 변환한다. 전단 변환부(210)는 구문 분석을 수행하는 일종의 파서(parser)로 볼 수 있다.

[0033] 타겟 적응 변환부(220)는 전단 변환부(210)에서 IR 형식으로 변환된 코드를 타겟이 되는 스위치에 대응하는 구조로 종속 변환을 수행하여 P4 IR을 출력한다.

[0034] P4 언어는 초기부터 타겟, 즉 적용 대상 장치에 독립적으로 설계된 언어로서 프로그램의 수정 없이 ASIC, FPGA, CPU, NPU 및 GPU와 같은 다양한 타겟에서 실행되도록 컴파일될 수 있도록 설계되었다. 따라서 패킷 프로세싱 프로그램 컴파일 장치는 P4 언어로 작성된 소스 코드를 컴파일 할 때, 타겟이 되는 장치(여기서는 일예로 프로그래머블 스위치)에서 적용 가능하도록 변환되어야 한다. 또한 소스 코드에는 클래스 형식으로 사전 정의된 각종 객체(object)를 호출하도록 작성될 수도 있다.

[0035] 이에 타겟 적응 변환부(220)는 전단 변환부(210)에서 타겟에 독립적으로 변환된 IR 코드를 타겟에 적합한 형식으로 변환하여 P4 IR을 획득한다. 이때 타겟 적응 변환부(220)는 호출되는 클래스의 객체를 함께 포함하여 변환할 수 있다.

[0036] 여기서 전단부(200)는 일예로 P4 언어를 P4 IR로 컴파일하도록 공개된 P4C 컴파일러로 구현될 수 있다. 그리고 경우에 따라서는 전단부(200)의 전단 변환부(210)를 전단부(frontend)라 하고, 타겟 적응 변환부(220)를 중단부(mid-end)라고 할 수도 있다.

[0037] 전단부(200)의 구성 및 기능은 공지된 기술이므로 여기서는 상세하게 설명하지 않는다.

[0038] 후단부(300)는 전단부(200)에서 생성된 P4 IR을 인가받고 분석하여 P4 IR의 자원 할당을 결정하여 도메인 지정 언어로 변환한다. 여기서 도메인 지정 언어는 컴파일 장치에 의해 지정된 형식의 네트워크 스위치에서 실행 가능한 패킷 프로세싱 프로그램을 의미한다. 그리고 여기서는 일예로 P4 IR을 도메인 지정 언어인 PX 언어로 변환하는 것으로 가정하여 설명한다.

[0039] 특히 본 실시예에서 후단부(300)는 패킷 프로세싱 프로그램의 동작을 베이직 블록 단위로 분해하고, 분해된 베이직 블록 사이의 의존성을 분석하여 병렬화함으로써, 패킷 프로세싱 프로그램의 저지연성을 개선되도록 변환할

수 있다.

[0040] 프로그램 출력부(400)는 후단부(300)에서 획득된 패킷 프로세싱 프로그램을 출력한다. 프로그램 출력부(400)에서 출력되는 패킷 프로세싱 프로그램은 FPGA를 기반으로 하는 스위치 칩으로 제조될 수 있다.

[0041] 도 2는 도 1의 후단부의 상세 구조를 나타내고, 도 3은 소스 코드에서 테이블의 구조를 개념적으로 나타낸 도면이며, 도 4는 테이블을 기본 블록 단위로 구분하는 개념을 나타낸 도면이다. 그리고 도 5는 조건 구문에 의해 분기되는 두개의 테이블 사이의 관계를 시각적으로 나타낸 도면이고, 도 6은 두개의 테이블에서 매치 영역 및 액션 영역 사이의 의존성을 고려하여 병렬화를 수행한 결과를 시각적으로 나타낸 도면이며, 도 7은 엔진들 사이의 의존성에 따른 병렬화를 수행한 결과를 나타낸다. 도 8은 도 2의 결합 변환부가 병렬로 배치된 튜플 엔진을 결합한 결과를 시각적으로 나타낸다.

[0042] 도 2를 참조하면, 후단부(300)는 후단 변환부(310), 테이블 분해부(320), 엔진 연관부(330), 의존성 분석부(340), 병렬화부(350), 결합 변환부(360) 및 코드 생성부(370)를 포함할 수 있다.

[0043] 후단 변환부(310)는 전단부에서 생성된 P4 IR을 인가받고, P4 IR을 구문 분석하여 P4 IR의 각 객체를 PX 언어에서 기지정된 형태의 구성 요소로 변환한다. 후단 변환부(310)는 전단부(200)의 전단 변환부(210)와 유사하게 구현되어 P4 IR의 구문을 분석하여 변환하여 PX 언어의 중간 표현(IR) 형태인 PX IR을 생성할 수 있다. 후단 변환부(310)는 P4 언어 형식으로 표현된 P4 IR을 PX 언어 형식인 PX IR로 변환함에 있어, P4 IR의 헤더 부분에서 헤더 유형에 따른 비트 폭을 획득하여 변환하거나 오류 코드를 생성하는 등의 기지정된 방식으로 변환한다. 그리고 P4 IR에 기술된 각종 상태 동작을 PX 섹션으로 변환한다.

[0044] P4 언어에서 소스 코드는 데이터 평면에서의 각종 동작을 정의하는 테이블(table)의 집합으로 구성된다.

[0045] 도 3에 도시된 바와 같이, 각각의 테이블은 명령 실행의 대상인지 여부를 판별하기 위한 키(key)가 기술되는 매치 영역과 키와 매치되는 경우 또는 매치되지 않는 경우에 수행해야 하는 동작이 기술된 액션 영역을 포함한다.

[0046] 도 3에서 (a)는 하나의 매치 영역(Match)과 하나의 액션 영역(Action)으로 구성된 테이블을 도시한 것으로 매치 영역(Match)에 기술된 키에 따른 조건에 대응하는 경우, 액션 영역(Action)에 기술된 액션을 수행하라는 명령을 나타낸다.

[0047] 한편 (b)는 하나의 매치 영역(Match)과 2개의 액션 영역(Action1, Action2)으로 구성된 테이블로서, 매치 영역(Match)에 기술된 키에 따른 조건에 대응하는 경우, 제1 액션 영역(Action1)에 기술된 액션을 수행하고, 대응하지 않으면, 제2 액션 영역(Action2)에 기술된 액션을 수행하라는 명령을 나타낸다.

[0048] 일예로 매치 영역에 기술되는 키는 컨트롤 평면에 의해 지정되는 목적지 어드레스일 수 있으며, 액션 영역에 기술되는 액션은 패킷이 전송될 포트 번호, 데이터(패킷 헤더, 메타 데이터 등) 변경 또는 패킷 드롭 등일 수 있다.

[0049] 이에 후단 변환부(310)는 P4 IR을 구분 분석하여 다수의 테이블을 구분하고 정렬할 수 있다.

[0050] 한편, 테이블 분해부(320)는 후단 변환부(310)에서 구분된 다수의 테이블 각각에서 매치 영역과 액션 영역 각각을 구분하여 테이블을 분해한다.

[0051] 상기한 바와 같이 다수의 테이블 각각은 매치 영역(Match)과 액션 영역(Action, Action1, Action2)으로 구성되므로, 테이블 분해부(320)는 각각의 테이블로부터 매치 영역(Match)과 적어도 하나의 액션 영역(Action, Action1, Action2)으로 구분할 수 있다.

[0052] 다만 PX 언어에서는 P4 언어와 달리 네트워크 스위치의 프로세서가 수행해야 하는 동작의 기본 단위인 베이직 블록 단위로 기술되어야 하며, 이에 다수의 테이블 각각을 베이직 블록 단위로 동작을 수행하는 튜플 엔진(Tuple Engine)과 룩업 엔진(Lookup Engine)의 구성으로 분해한다. 이때 P4 언어에서 작성된 하나의 테이블은 하나의 룩업 엔진과 둘 이상의 튜플 엔진으로 분해될 수 있다.

[0053] 도 4를 참조하면, 테이블 분해부(320)는 하나의 매치 영역(Match)과 액션 영역(Action)을 포함하는 테이블을 하나의 룩업 엔진(Lookup)과 2개의 튜플 엔진(Tuple1, Tuple2)으로 분해하여 변환할 수 있다. 또한 하나의 매치 영역(Match)과 2개의 액션 영역(Action1, Action2)을 포함하는 테이블을 하나의 룩업 엔진(Lookup)과 3개의 튜플 엔진(Tuple1, Tuple2, Tuple3)으로 분해하여 변환할 수 있다.

[0054] 여기서 제1 튜플 엔진(Tuple1)은 매치 영역(Match)에 기술된 키를 기반으로 룩업 엔진(Lookup)에 요청을 전송하

는 동작을 수행하는 요청 전송(send request) 엔진이고, 룩업 엔진(Lookup)은 컨트롤 평면에서 키에 대응하는 값을 인가받아 매치 여부를 판정하는 판정 엔진이다. 그리고 제2 및 제3 튜플 엔진(Tuple2, Tuple3)은 룩업 엔진(Lookup)의 판정 결과에 대응하는 동작을 수행하는 응답 수신 및 액션 수행(receive Response & perform action) 엔진이다. 즉 룩업 엔진(Lookup)에서 매치되는 것으로 판정되면, 제2 튜플 엔진(Tuple2)이 구동되는 반면, 매치되지 않는 것으로 판정되면 제3 튜플 엔진(Tuple3)이 구동될 수 있다.

[0055] 따라서 제1 튜플 엔진(Tuple1)과 룩업 엔진(Lookup)은 테이블의 매치 영역에 대응하고, 제2 및 제3 튜플 엔진(Tuple2, Tuple3)은 액션 영역에 대응하는 것으로 볼 수 있다.

[0056] 다만 P4 언어에는 조건 명령인 if 구문이 이용될 수 있으며, if 구분이 포함되면, 룩업 엔진(Lookup)의 판정 결과에 따라 제2 또는 제3 튜플 엔진 (Tuple2, Tuple3)이 아닌 다른 룩업 엔진이나 튜플 엔진으로 분기될 수도 있다. 일예로 다른 테이블의 튜플 엔진으로 분기될 수도 있다.

[0057] 엔진 연관부(330)는 테이블 분해부(320)에서 분해된 다수 테이블의 룩업 엔진(Lookup)과 튜플 엔진(Tuple)들 사이의 의존성 관계를 분석한다. 상기한 바와 같이 다수의 테이블 각각에서 분해된 룩업 엔진(Lookup)과 튜플 엔진(Tuple)은 if 구문 등에 의해 다른 테이블과 연관 관계를 가질 수 있다.

[0058] 도 5에서는 두개의 테이블(T1, T2) 사이의 관계를 시각적으로 나타낸 결과로서 (a)에서는 제1 테이블(T1)의 룩업 엔진(Lookup1)에서 매치 결과가 if 구문에 의해 제2 테이블(T2)의 제2 튜플(Tuple22)로 분기되는 과정을 시각적으로 나타내었다. 여기서 if 구문의 조건에 대한 매치 여부 또한 제1 테이블(T1)의 룩업 엔진(Lookup1)에서 수행되는 동작이므로, (a)에서 별도로 도시된 if 구문에 해나 구성은 (b)와 같이 룩업 엔진(Lookup1)에 포함되도록 다시 표현될 수 있다.

[0059] 소프트웨어적으로 구성된 각 구성 요소 사이의 연관 관계를 분석하여 출력하는 엔진 연관부(330)는 이미 공지된 기술이므로 여기서는 상세하게 설명하지 않는다.

[0060] 의존성 분석부(340)는 엔진 연관부(330)에서 연관 관계가 분석된 다수의 테이블에서 룩업 엔진(Lookup)과 튜플 엔진(Tuple) 또는 튜플 엔진(Tuple)과 튜플 엔진(Tuple) 사이의 의존성을 분석한다.

[0061] 도 4에 도시된 바와 같이, 하나의 테이블 내에서 룩업 엔진(Lookup)과 튜플 엔진(Tuple) 사이의 의존성은 매우 단순하게 분석될 수 있다. 즉 룩업 엔진(Lookup)은 제1 튜플 엔진(Tuple1)에 대해 의존성을 가지며, 제2 및 제3 튜플 엔진(Tuple2, Tuple3) 각각은 룩업 엔진(Lookup)에 의존성을 갖는다.

[0062] 그러나 도 5에 도시된 바와 같이, 액션 영역에 대응하는 제2 또는 제3 튜플 엔진(여기서는 일예로 Tuple22)은 다른 테이블의 룩업 엔진(Lookup1)에 대해서도 의존성을 가질 수 있다. 뿐만 아니라 제2 또는 제3 튜플 엔진은 다른 테이블의 제2 또는 제3 튜플 엔진에 대해서도 의존성을 가질 수 있다.

[0063] 일예로 도 6의 (a)에 도시된 바와 같이, 제2 테이블(T2)의 제2 튜플 엔진(Tuple22)은 제1 테이블(T1)의 제2 튜플 엔진(Tuple12)에 대한 의존성을 가질 수도 있다. 이는 제1 테이블(T1)의 제2 튜플 엔진(Tuple12)이 제2 테이블(T2)의 제2 튜플 엔진(Tuple22)에서 이용해야하는 데이터를 변경하는 등의 작업을 수행하는 경우에 발생될 수 있다. 즉 제2 테이블(T2)의 제2 튜플 엔진(Tuple22)은 제1 테이블(T1)의 제2 튜플 엔진(Tuple12)에 대해 데이터 의존성을 가질 수 있다.

[0064] 의존성 분석부(340)는 이와 같이 다수 테이블 사이에서 룩업 엔진(Lookup)과 튜플 엔진(Tuple) 또는 튜플 엔진(Tuple)과 튜플 엔진(Tuple) 사이의 의존성을 분석한다. 다만 테이블들의 매치 영역과 액션 영역 사이와 액션 영역과 액션 영역 사이에는 의존성이 발생될 수 있으나, 조건을 판별하는 매치 영역 사이에는 의존성이 발생되지 않는다. 즉 PX 언어 형식으로 변환된 다수 테이블에서 제1 튜플 엔진(Tuple1)들 사이 및 룩업 엔진들(Lookup) 사이에는 의존성이 발생되지 않는다.

[0065] 병렬화부(350)는 의존성 분석부(340)에서 분석된 엔진들 사이의 의존성을 분석하여 병렬화를 수행한다.

[0066] 도 7을 참조하면, 병렬화부(350)는 우선 의존성이 없는 엔진을 탐색하여 우선 병렬로 배치한다. 상기한 바와 같이 다수의 테이블 각각에서 매치 영역에 대응하는 제1 튜플 엔진(Tuple1)은 의존성이 없으므로, 최우선 구간에 병렬로 배치될 수 있다. 다만 하드웨어적으로 네트워크 스위치의 파이프 라인 개수가 미리 지정된 경우, 지정된 파이프 라인 개수에 대응하는 개수만큼의 엔진을 병렬로 배치할 수 있다.

[0067] 이후 병렬화부(350)는 병렬로 배치된 엔진에서 분기되는 종속성을 모두 해제하여 다시 의존성을 분석한다. 즉 배치된 엔진을 제외하여 배치된 엔진에 의존하는 엔진들의 의존성을 제거한다. 그리고 다시 의존성이 없는 엔

진을 탐색하여 현재 배치된 엔진의 다음 구간에 병렬로 배치한다. 따라서 제1 튜플 엔진(Tuple1)에 의존하는 룩업 엔진들(Lookup)이 다음 구간에 배치될 수 있다.

[0068] 병렬화부(350)는 의존성 분석된 모든 엔진들이 순차적으로 배치될 때까지 이와 같은 작업을 반복적으로 수행한다. 즉 P4 언어의 테이블에서 PX 형식으로 변환된 모든 엔진들이 배치되도록 한다. 도 7에서는 제2 테이블의 제2 튜플 엔진(Tuple22)이 제1 테이블의 제2 튜플 엔진(Tuple12)에 대해 의존성을 가지므로, 제2 테이블의 제2 튜플 엔진(Tuple22)이 제1 테이블의 제2 튜플 엔진(Tuple12)의 다음 구간에 배치되는 것으로 도시되어 있다.

[0069] 그리고 결합 변환부(360)는 병렬화부(350)에 의해 병렬로 배치된 다수의 엔진 중 동일 구간에 배치된 튜플 엔진(Tuple11, Tuple12)를 병합한다. 결합 변환부(360)는 베이직 블록 단위로 동작하는 튜플 엔진들이 동시에 수행될 수 있도록 병합한다.

[0070] 코드 생성부(370)는 병렬화부(350)에 의해 병렬화되고 결합 변환부(360)에 의해 결합된 엔진들의 배치 순서에 기반하여 기지정된 형식의 프로그램 코드를 생성한다. 여기서 코드 생성부(370)는 일예로 PX 기반의 패킷 프로세싱 프로그램 코드를 생성할 수 있다.

[0071] 결과적으로 도 2에 도시된 본 실시예에 따른 패킷 프로세싱 프로그램 컴파일 장치는 소스 코드에 포함된 다수의 테이블을 매치 영역과 액션 영역으로 구분하고, 구분된 매치 영역에 대응하는 튜플 엔진과 룩업 엔진, 그리고 액션 영역에 대응하는 튜플 엔진들 사이의 의존성을 분석하여 병렬화함으로써, 고속 프로세싱이 가능한 패킷 프로세싱 프로그램을 생성할 수 있다. 즉 본 실시예의 패킷 프로세싱 프로그램 컴파일 장치는 다수의 테이블에 대해 병렬 수행을 지정할 수 없는 P4 언어로 작성된 소스 코드를 인가받아 베이직 블록 단위로 분해하고, 분해된 베이직 블록 단위의 의존성에 기초하여 병렬화함으로서 패킷 프로세싱 프로그램을 자동 최적화할 수 있다. 또한 하드웨어로 구현되는 네트워크 스위치의 제약 사항을 반영하여 병렬화 개수를 조절할 수 있으므로, 다양한 제약 조건에서도 최적화된 패킷 프로세싱 프로그램을 생성할 수 있다.

[0072] 기존에도 P4 언어로 작성된 소스 코드의 테이블에 대한 병렬화 기법이 연구된 바가 있으나, 기존 연구에서는 테이블 단위의 의존성을 분석하여 병렬화를 수행함에 따라 매치 영역 또는 액션 영역 중 적어도 하나라도 다른 테이블에 대해 의존성을 갖는 경우, 해당 테이블들은 병렬화가 불가능하였다. 그러나 본 발명에서는 테이블을 베이직 블록 단위로 구분하여 병렬화를 수행함에 따라 다수의 테이블의 매치 영역 또는 액션 영역 사이에 의존성이 존재하더라도, 의존성이 존재하지 않는 나머지 베이직 블록 단위들을 병렬화함으로써 네트워크 스위치의 성능을 더욱 향상시킬 수 있다. 즉 테이블의 일부 구성들이 서로 병렬화될 수 있어 더욱 효율적으로 동작하는 패킷 프로세싱 프로그램을 생성할 수 있다.

[0073] 본 실시예에서 패킷 프로세싱 프로그램 컴파일 장치의 각 구성은 하드웨어로 구현될 수도 있으나, 소프트웨어 모듈과 같은 프로그램으로 구현될 수 있다. 그리고 패킷 프로세싱 프로그램 컴파일 장치는 프로세서와 메모리를 포함하여 소프트웨어 모듈을 실행하는 컴퓨터나 이에 준하는 각종 장치로 구현될 수 있다.

[0074] 도 9는 본 발명의 일 실시예에 따른 패킷 프로세싱 프로그램 컴파일 방법을 나타내고, 도 10은 도 9의 병렬화 단계를 상세하게 나타낸다.

[0075] 도 9를 참조하면, 본 실시예에 따른 패킷 프로세싱 프로그램 컴파일 방법은 컴파일되어야 하는 패킷 프로세싱 프로그램의 소스 코드를 획득한다(S10). 여기서 소스 코드는 상기한 바와 같이, P4 언어로 작성된 코드일 수 있다. 그리고 획득된 소스 코드를 기지정된 방식에 따라 전단 중간 표현으로 변환한다(S20). 소스 코드가 P4 언어로 작성된 경우, 소스 코드는 P4 중간 언어 형식인 P4 IR로 변환될 수 있다. 그리고 소스 코드가 전단 중간 표현으로 변환되면, 전단 중간 표현으로 변환된 P4 IR을 분석하여 동작을 베이직 블록 단위로 분해하고, 분해된 베이직 블록 사이의 의존성을 분석하여 병렬화 및 결합하여 소스 코드에 대응하는 패킷 프로세싱 프로그램 코드를 생성한다(S30).

[0076] 패킷 프로세싱 프로그램 코드를 생성하는 단계(S30)는 구체적으로 우선 전단 중간 표현을 인가받아 전단 중간 표현의 각 객체를 기지정된 후단 중간 표현에 따른 구성 요소로 변환한다(S31). 이때 변환된 후단 중간 표현에서 동작 명령을 정의하는 다수의 테이블을 구분할 수 있다.

[0077] 그리고 구분된 다수의 테이블 각각을 분해한다(S32). 여기서 다수의 테이블 각각은 매치 영역과 적어도 하나의 액션 영역으로 구성되며, 매치 영역과 액션 영역은 프로세서가 수행할 수 있는 동작의 기본 단위인 베이직 블록 단위로 다시 분해될 수 있다. 매치 영역은 하나의 튜플 엔진과 하나의 룩업 엔진(Lookup)으로 분해될 수 있으며, 액션 영역은 각각 대응하는 튜플 엔진으로 구분될 수 있다.

[0078] 여기서 테이블의 매치 영역에서 분해되는 튜플 엔진을 제1 튜플 엔진(Tuple1)이라 하고, 테이블의 액션 영역의 개수에 따라 액션 영역은 제2 튜플 엔진(Tuple2)으로 구분되거나, 제2 및 제3 튜플 엔진(Tuple2, Tuple3)으로 구분될 수 있다.

[0079] 한편, 다수의 테이블 각각이 베이직 블록 단위로 분해되면, 다수의 베이직 블록 사이의 연관 관계를 분석한다(S33). 여기서 베이직 블록 단위 사이의 연관 관계는 기본적으로 하나의 테이블 내에서 룩업 엔진(Lookup)은 제1 튜플 엔진(Tuple1)과 종속되고, 제2 및 제3 튜플 엔진(Tuple2, Tuple3)은 룩업 엔진(Lookup)에 종속된다. 또한 테이블에 조건 구문인 if 구문이 포함되는 경우, 제2 및 제3 튜플 엔진(Tuple2, Tuple3)은 다른 테이블의 룩업 엔진에 종속될 수 있다. 뿐만 아니라, 제2 및 제3 튜플 엔진(Tuple2, Tuple3)은 다른 테이블의 제2 또는 제3 튜플 엔진(Tuple2, Tuple3)에 대해 데이터가 종속될 수도 있다.

[0080] 다수의 베이직 블록 사이의 연관 관계를 분석되면, 분석된 연관 관계로부터 다수의 베이직 블록들 간의 의존성을 분석한다(S34). 그리고 분석된 의존성에 기초하여 다수의 베이직 블록을 병렬화하여 배치한다(S35).

[0081] 도 10을 참조하면 병렬화 시에는 먼저 각각 베이직 블록을 나타내는 다수의 엔진 중 다른 엔진에 의존하지 않는 비의존성 엔진을 탐색한다(S351). 여기서 비 의존성 엔진으로는 우선 제1 튜플 엔진(Tuple1)이 대표적이다. 그리고 탐색된 비의존성 엔진을 이전 시간 구간에 배치된 엔진들 다음 구간에 병렬로 배치한다(S352). 그리고 배치된 엔진들에 대한 의존성을 제거한다(S353). 즉 배치된 엔진에 의존하는 엔진들의 의존성을 제거한다. 그리고 배치되지 않은 엔진이 존재하는지 판별한다(S354). 만일 배치되지 않은 엔진이 존재하면, 다시 배치되지 않은 엔진들 중 비의존성 엔진을 탐색한다(S351). 그러나 모든 엔진이 배치된 것으로 판별되면, 병렬화를 종료하고, 병렬로 배치된 엔진들 중 동일 구간에 배치된 튜플 엔진들을 결합한다(S36).

[0082] 이후 결합된 튜플 엔진들 및 배치된 엔진들의 배치 순서에 기반하여 기지정된 형식의 프로그램 코드를 생성한다(S37). 패킷 프로세싱 프로그램 코드가 생성되면, 생성된 패킷 프로세싱 프로그램 코드를 출력한다(S40).

[0083] 본 발명에 따른 방법은 컴퓨터에서 실행시키기 위한 매체에 저장된 컴퓨터 프로그램으로 구현될 수 있다. 여기서 컴퓨터 판독가능 매체는 컴퓨터에 의해 액세스 될 수 있는 임의의 가용 매체일 수 있고, 또한 컴퓨터 저장 매체를 모두 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현된 휘발성 및 비휘발성, 분리형 및 비분리형 매체를 모두 포함하며, ROM(판독 전용 메모리), RAM(랜덤 액세스 메모리), CD(컴팩트 디스크)-ROM, DVD(디지털 비디오 디스크)-ROM, 자기 테이프, 플로피 디스크, 광데이터 저장장치 등을 포함할 수 있다.

[0084] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다.

[0085] 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 청구범위의 기술적 사상에 의해 정해져야 할 것이다.

부호의 설명

[0086] 100: 소스 코드 획득부 200: 전단부

210: 전단 변환부 220: 타겟 적응 변환부

300: 후단부 310: 후단 변환부

320: 테이블 분해부 330: 엔진 연관부

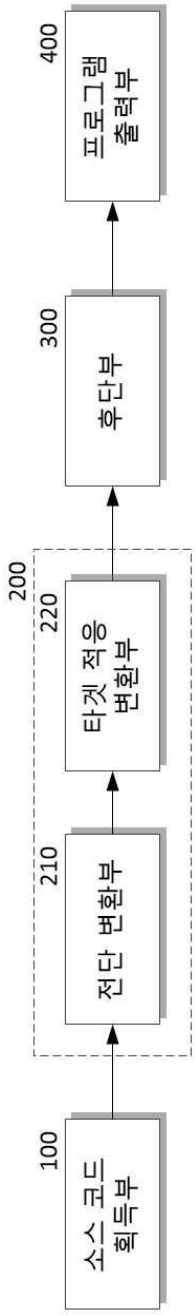
340: 의존성 분석부 350: 병렬화부

360: 결합 변환부 370: 코드 생성부

400: 프로그램 출력부

도면

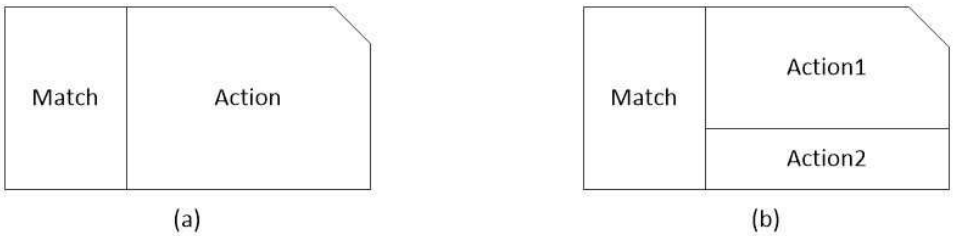
도면1



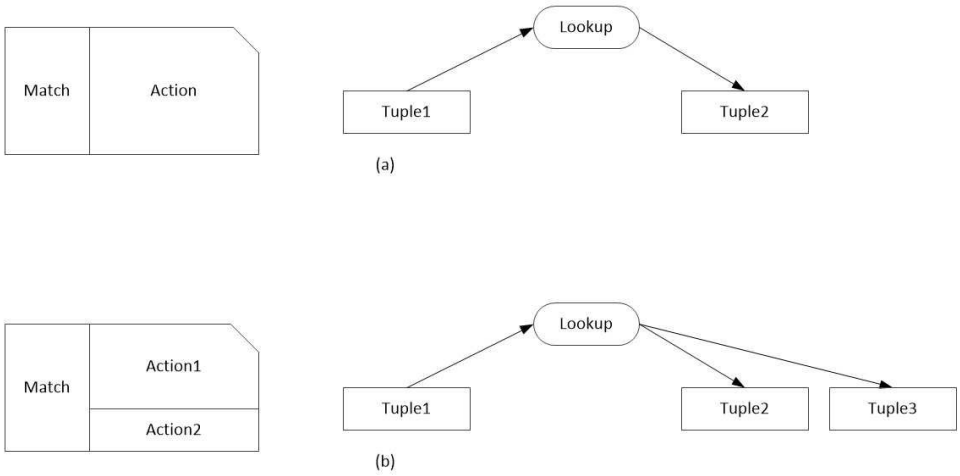
도면2


300
370
코드 생성부
360
결합 변환부
350
병렬화부
340
의존성 분석부
330
엔진 연관부
320
테이블 분해부
310
후단 변환부

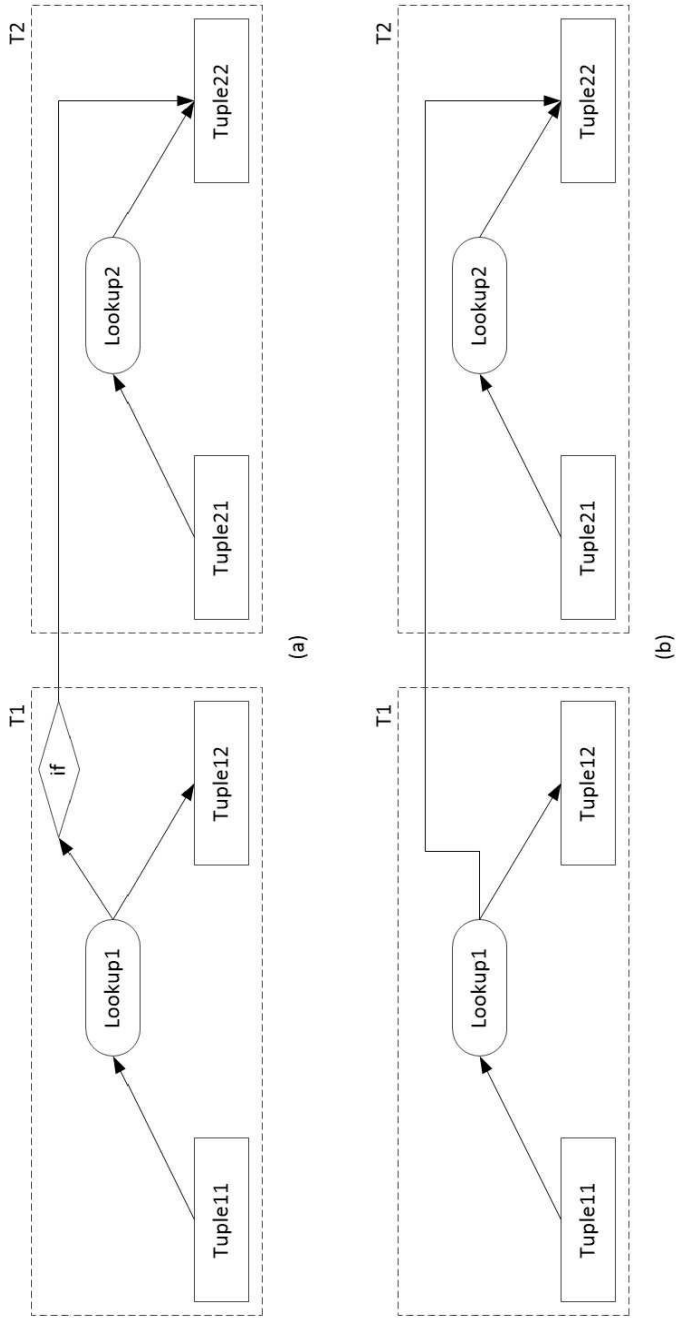
도면3


Match
Action
(a)
Match
Action1
Action2
(b)

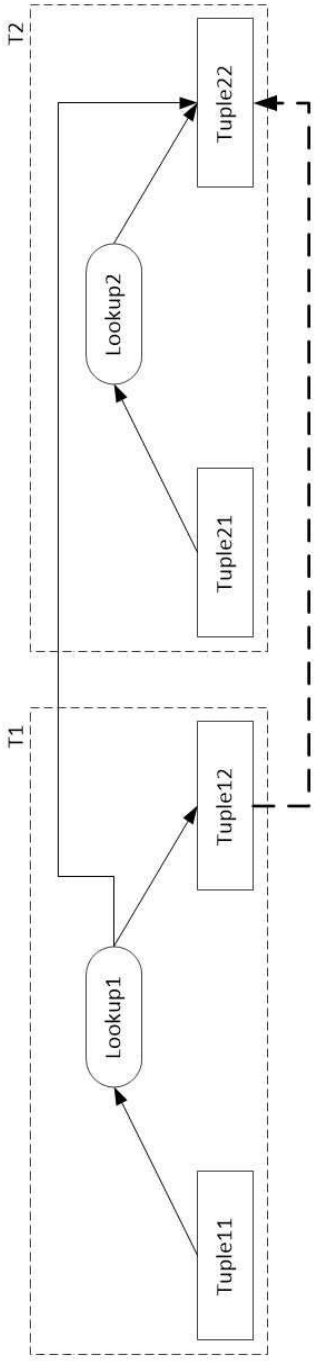
도면4


Match
Action
Lookup
Tuple1
Tuple2
(a)
Match
Action1
Action2
Lookup
Tuple1
Tuple2
Tuple3
(b)

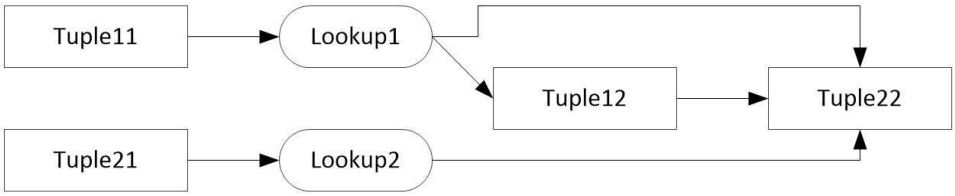
도면5


T2
Tuple22
Lookup2
Tuple21
T1
if
Tuple12
Lookup1
Tuple11
(a)
T2
Tuple22
Lookup2
Tuple21
T1
Tuple12
Lookup1
Tuple11
(b)

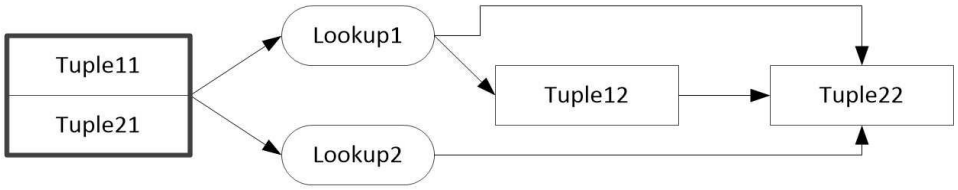
도면6


T2
Tuple22
Lookup2
Tuple21
T1
Tuple12
Lookup1
Tuple11

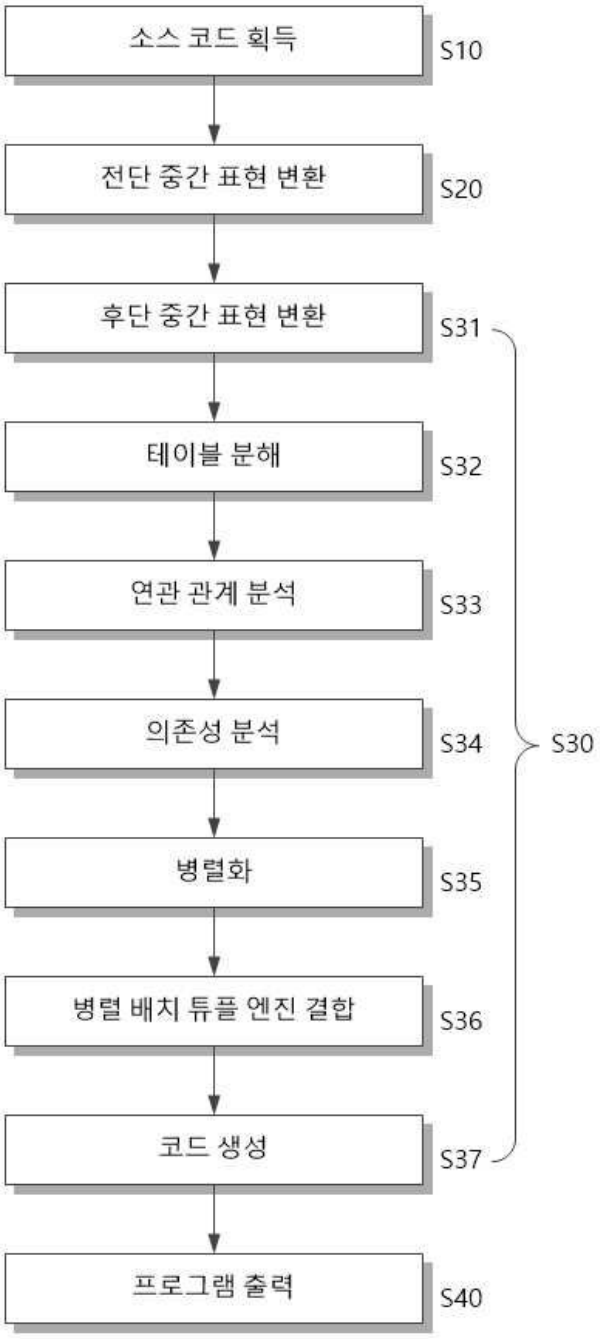
도면7


Tuple11
Lookup1
Tuple12
Tuple22
Tuple21
Lookup2

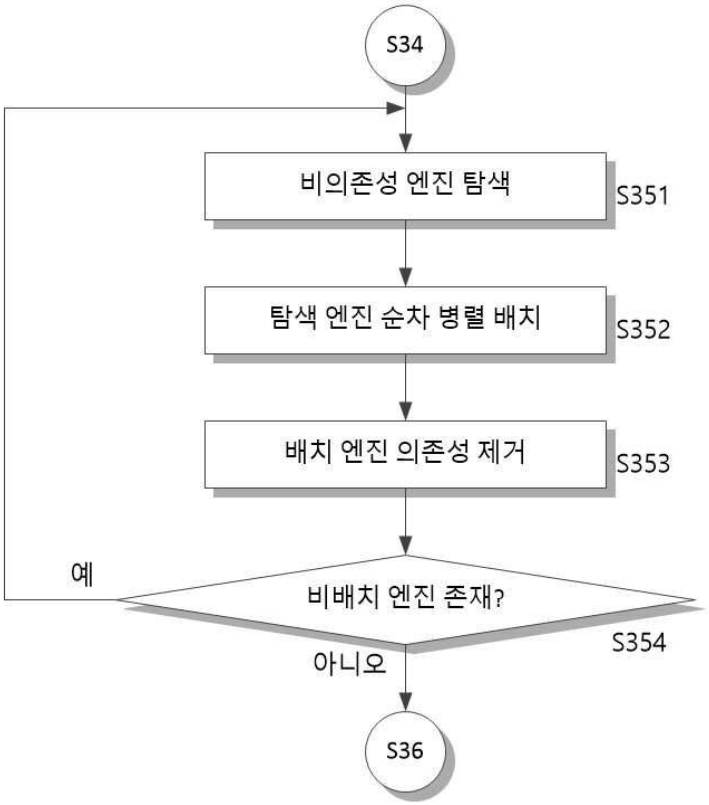
도면8


Tuple11
Tuple21
Lookup1
Lookup2
Tuple12
Tuple22

도면9


소스 코드 획득
S10
전단 중간 표현 변환
S20
후단 중간 표현 변환
S31
테이블 분해
S32
연관 관계 분석
S33
의존성 분석
S34
S30
병렬화
S35
병렬 배치 튜플 엔진 결합
S36
코드 생성
S37
프로그램 출력
S40

도면10



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 18

【변경전】

제11 항에 있어서, 상기 다음 시간 구간에 병렬로 배치하는 단계는

상기 패킷 프로세싱 프로그램 코드가 적용되는 타겟의 하드웨어적 파이프 라인의 개수 이하 개수의 엔진을 동일 시간 구간에 병렬로 배치하는 패킷 프로세싱 프로그램 컴파일 장치.

【변경후】

제11 항에 있어서, 상기 다음 시간 구간에 병렬로 배치하는 단계는

상기 패킷 프로세싱 프로그램 코드가 적용되는 타겟의 하드웨어적 파이프 라인의 개수 이하 개수의 엔진을 동일 시간 구간에 병렬로 배치하는 패킷 프로세싱 프로그램 컴파일 방법.