(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0069353
(43) 공개일자 2020년06월16일

(51) 국제특허분류(Int. Cl.)
G06N 3/063 (2006.01) *G06K 9/00* (2006.01)
G06N 3/04 (2006.01) *G06N 3/08* (2006.01)
G06N 3/10 (2006.01)
(52) CPC특허분류
G06N 3/063 (2013.01)
G06K 9/00986 (2013.01)
(21) 출원번호 10-2020-7013829
(22) 출원일자(국제) 2018년09월26일
심사청구일자 없음
(85) 번역문제출일자 2020년05월14일
(86) 국제출원번호 PCT/US2018/052833
(87) 국제공개번호 WO 2019/079008
국제공개일자 2019년04월25일
(30) 우선권주장
15/785,679 2017년10월17일 미국(US)

(71) 출원인
자일링크스 인코포레이티드
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
(72) 발명자
응 아론
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
제다 인드리히
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
(뒷면에 계속)
(74) 대리인
김태홍, 김진희

전체 청구항 수 : 총 15 항

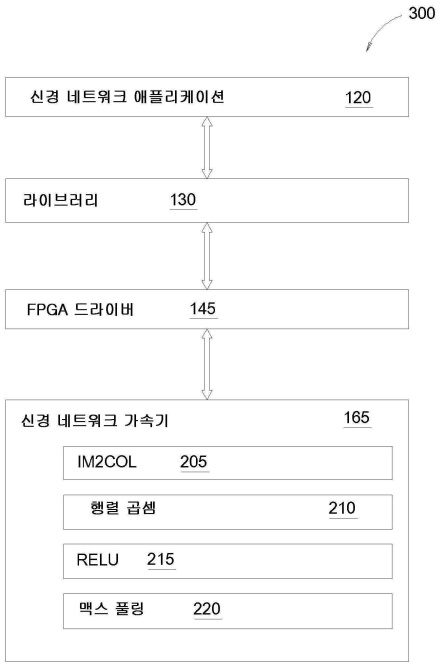
(54) 발명의 명칭 **신경 네트워크 가속화를 위한 머신 러닝 런타임 라이브러리**

(57) 요 약

본 명세서에서의 실시예들은 라이브러리(130)를 사용하여 신경 네트워크 애플리케이션(120)을 신경 네트워크 가속기(165)와 인터페이싱시키기 위한 기술들을 설명한다. 신경 네트워크 애플리케이션(120)은 호스트 컴퓨팅 시스템(105) 상에서 실행될 수 있는 반면, 신경 네트워크 가속기(165)는 대규모 병렬 하드웨어 시스템, 예를 들어,

(뒷면에 계속)

대 표 도 - 도3



FPGA(150) 상에서 실행된다. 라이브러리(130)는 신경 네트워크 애플리케이션(120)으로부터 수신되는 태스크들을 신경 네트워크 가속기(165)에 제출하기 위한 파이프라인(500)을 동작시킨다. 일 실시예에서, 파이프라인(500)은 각각이 상이한 스레드들에 대응하는 프리-프로세싱 스테이지, FPGA 실행 스테이지, 및 포스트-프로세싱 스테이지(135)를 포함한다. 신경 네트워크 애플리케이션(120)으로부터 태스크를 수신할 때, 라이브러리(130)는 태스크들을 수행하기 위해 파이프라인 내의 상이한 스테이지들에 대해 요구되는 정보를 포함하는 패킷을 생성한다(410). 스테이지들이 상이한 스레드들에 대응하기 때문에(415), 라이브러리(130)는 다수의 패킷들을 병렬로 프로세싱할 수 있으며 이는 하드웨어 시스템 상의 신경 네트워크 가속기(165)의 이용률을 증가시킬 수 있다.

(52) CPC특허분류
G06N 3/04 (2013.01)
G06N 3/08 (2013.01)
G06N 3/10 (2013.01)

(72) 발명자
델라예 엘리엇
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
탱 샤오
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
산탄 소날
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
소이 소렌 티
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
시라사오 애쉬시
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
가세미 에산
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100
세틀 션
미합중국 95124 캘리포니아 산 호세 로직 드라이브 2100

명 세 서

청구범위

청구항 1

신경 네트워크 가속기에 제출된 태스크들을 파이프라이닝하기 위한 방법으로서,

상기 신경 네트워크 가속기에 의해 프로세싱될 제1 태스크를 신경 네트워크 애플리케이션으로부터 수신하는 단계;

하나 이상의 컴퓨터 프로세서를 사용하여, 파이프라인 내의 다수의 스테이지들에 의해 사용되는 정보를 포함하는 패킷을 생성하는 단계;

상기 다수의 스테이지들에서 상기 패킷을 프로세싱하는 단계 - 상기 다수의 스테이지들 중 적어도 하나는 상기 신경 네트워크 가속기를 실행하는 하드웨어 시스템에 대한 호출을 수행하며, 상기 파이프라인은 상기 패킷을 프로세싱하는 것과 병렬로 제2 태스크에 대응하는 적어도 하나의 다른 패킷을 프로세싱함 -; 및

상기 파이프라인을 사용하여 상기 패킷을 프로세싱한 결과들을 상기 신경 네트워크 애플리케이션에 반환하는 단계

를 포함하는, 방법.

청구항 2

제1항에 있어서,

상기 다수의 스테이지들에서 상기 패킷을 프로세싱하는 단계는:

프리-프로세싱(pre-processing) 스테이지에서 상기 패킷을 프로세싱하는 단계;

상기 프리-프로세싱 스테이지 이후에 발생하는 실행 스테이지에서 상기 패킷을 프로세싱하는 단계 - 상기 하드웨어 시스템에 대한 호출은 상기 실행 스테이지 동안 발생함 -; 및

상기 실행 스테이지 이후의 포스트-프로세싱(post-processing) 스테이지에서 상기 패킷을 프로세싱하는 단계

를 포함하는 것인, 컴퓨터-구현 방법.

청구항 3

제2항에 있어서,

상기 프리-프로세싱 스테이지에서 상기 패킷을 프로세싱하는 단계는:

제1 태스크에 대응하는 데이터를 상기 신경 네트워크 애플리케이션에 의해 사용되는 제1 포맷으로부터 상기 하드웨어 시스템에 의해 사용되는 제2 포맷으로 변환하는 단계를 포함하고,

상기 포스트-프로세싱 스테이지에서 상기 패킷을 프로세싱하는 단계는,

상기 결과들을 상기 제2 포맷으로부터 상기 제1 포맷으로 변환하는 단계를 포함하는 것인, 컴퓨터-구현 방법.

청구항 4

제1항에 있어서,

상기 다수의 스테이지들 각각은 다른 스레드(thread)들과 독립적으로 상기 패킷을 프로세싱하는 각자의 스레드를 포함하는 것인, 컴퓨터-구현 방법.

청구항 5

제1항에 있어서,

상기 신경 네트워크 애플리케이션을 위한 할당된 메모리 블록들을 상기 하드웨어 시스템 내의 상기 신경 네트워

크 가속기를 위한 할당된 메모리 블록들에 매핑하는 메모리 맵을 생성하는 단계; 및

상기 신경 네트워크 애플리케이션으로부터 수신되는 제1 메모리 어드레스들을 상기 메모리 맵에 기초하여 상기 하드웨어 시스템 내의 메모리 블록들에 대한 제2 메모리 어드레스들로 변환하는 단계

를 더 포함하는, 컴퓨터-구현 방법.

청구항 6

제1항에 있어서,

신경 네트워크의 다수의 계층들을 수행하는 데 사용되는 가중치들을 행렬 포맷으로 상기 하드웨어 시스템에게 전송하는 단계; 및

새로운 태스크에 대응하는 상기 가중치들의 서브세트를 식별하는 단계; 및

상기 패킷을 프로세싱할 때 사용될 상기 가중치들의 서브세트를 지시하는 오프셋을 상기 하드웨어 시스템에게 전송하는 단계

를 더 포함하는, 컴퓨터-구현 방법.

청구항 7

제1항에 있어서,

상기 하드웨어 시스템 상의 상기 신경 네트워크 가속기의 실행에 관한 메트릭(metric)을 획득하는 단계;

상기 메트릭의 시각적 표현을 디스플레이를 위해 출력하는 단계; 및

상기 신경 네트워크 가속기의 이용률을 증가시키기 위해 상기 파이프라인 내의 상기 다수의 스테이지들을 실행하는 하드웨어 자원들을 조정하는 단계

를 더 포함하는, 컴퓨터-구현 방법.

청구항 8

제1항에 있어서,

상기 파이프라인 내의 상기 다수의 스테이지들은 라이브러리에 정의되어 있고, 상기 라이브러리는 상이한 유형들의 신경 네트워크 애플리케이션들이 상기 파이프라인 내의 상기 다수의 스테이지들을 사용하여 태스크들을 상기 신경 네트워크 가속기에 제출할 수 있게 해주도록 구성되는 애플리케이션 프로그램 인터페이스(application program interface; API)를 포함하는 것인, 컴퓨터-구현 방법.

청구항 9

컴퓨팅 시스템으로서,

프로세서; 및

라이브러리를 포함하는 메모리

를 포함하며, 상기 라이브러리는, 상기 프로세서에 의해 실행될 때, 동작을 수행하고, 상기 동작은:

신경 네트워크 가속기에 의해 프로세싱될 제1 태스크를 신경 네트워크 애플리케이션으로부터 수신하는 것;

하나 이상의 컴퓨터 프로세서를 사용하여, 파이프라인 내의 다수의 스테이지들에 의해 사용되는 정보를 포함하는 패킷을 생성하는 것;

상기 다수의 스테이지들에서 상기 패킷을 프로세싱하는 것 - 상기 다수의 스테이지들 중 적어도 하나는 상기 신경 네트워크 가속기를 실행하는 하드웨어 시스템에 대한 호출을 수행하며, 상기 파이프라인은 상기 패킷을 프로세싱하는 것과 병렬로 제2 태스크에 대응하는 적어도 하나의 다른 패킷을 프로세싱함 -; 및

상기 파이프라인을 사용하여 상기 패킷을 프로세싱한 결과들을 상기 신경 네트워크 애플리케이션에 반환하는 것을 포함하는, 컴퓨팅 시스템.

청구항 10

제9항에 있어서,

상기 다수의 스테이지들에서 상기 패킷을 프로세싱하는 동작은:

프리-프로세싱 스테이지에서 상기 패킷을 프로세싱하는 동작;

상기 프리-프로세싱 스테이지 이후에 발생하는 실행 스테이지에서 상기 패킷을 프로세싱하는 동작 - 상기 하드웨어 시스템에 대한 호출은 상기 실행 스테이지 동안 발생함 - ; 및

상기 실행 스테이지 이후의 포스트-프로세싱 스테이지에서 상기 패킷을 프로세싱하는 동작

을 포함하는 것인, 컴퓨팅 시스템.

청구항 11

제10항에 있어서,

상기 프리-프로세싱 스테이지에서 상기 패킷을 프로세싱하는 동작은:

제1 태스크에 대응하는 데이터를 상기 신경 네트워크 애플리케이션에 의해 사용되는 제1 포맷으로부터 상기 하드웨어 시스템에 의해 사용되는 제2 포맷으로 변환하는 동작을 포함하고,

상기 포스트-프로세싱 스테이지에서 상기 패킷을 프로세싱하는 동작은:

상기 결과들을 상기 제2 포맷으로부터 상기 제1 포맷으로 변환하는 동작을 포함하는 것인, 컴퓨팅 시스템.

청구항 12

제9항에 있어서,

상기 동작은:

상기 신경 네트워크 애플리케이션을 위한 할당된 메모리 블록들을 상기 하드웨어 시스템 내의 상기 신경 네트워크 가속기를 위한 할당된 메모리 블록들에 매핑하는 메모리 맵을 생성하는 동작; 및

상기 신경 네트워크 애플리케이션으로부터 수신되는 제1 메모리 어드레스들을 상기 메모리 맵에 기초하여 상기 하드웨어 시스템 내의 메모리 블록들에 대한 제2 메모리 어드레스들로 변환하는 동작

을 더 포함하는 것인, 컴퓨팅 시스템.

청구항 13

제9항에 있어서,

상기 동작은:

신경 네트워크의 다수의 계층들을 수행하는 데 사용되는 가중치들을 행렬 포맷으로 상기 하드웨어 시스템에게 전송하는 동작; 및

새로운 태스크에 대응하는 상기 가중치들의 서브세트를 식별하는 동작; 및

상기 패킷을 프로세싱할 때 사용될 상기 가중치들의 서브세트를 지시하는 오프셋을 상기 하드웨어 시스템에게 전송하는 동작

을 더 포함하는 것인, 컴퓨팅 시스템.

청구항 14

제9항에 있어서,

상기 동작은:

상기 하드웨어 시스템 상의 상기 신경 네트워크 가속기의 실행에 관한 메트릭을 획득하는 동작;

상기 메트릭의 시각적 표현을 디스플레이를 위해 출력하는 동작; 및

상기 신경 네트워크 가속기의 이용률을 증가시키기 위해 상기 파이프라인 내의 상기 다수의 스테이지들을 실행하는 하드웨어 자원들을 조정하는 동작

을 더 포함하는 것인, 컴퓨팅 시스템.

청구항 15

제9항에 있어서,

상기 파이프라인 내의 상기 다수의 스테이지들은 상기 라이브러리에 정의되어 있고, 상기 라이브러리는 상이한 유형들의 신경 네트워크 애플리케이션들이 상기 파이프라인 내의 상기 다수의 스테이지들을 사용하여 태스크들을 상기 신경 네트워크 가속기에 제출할 수 있게 해주도록 구성되는 API를 포함하는 것인, 컴퓨팅 시스템.

발명의 설명

기 술 분 야

[0001] 본 개시내용의 예들은 일반적으로 호스트 컴퓨팅 시스템 상에서 실행되는 신경 네트워크 애플리케이션과 신경 네트워크 가속기 사이의 통신에 관한 것이다.

배 경 기 술

[0002] 머신 러닝은 컴퓨팅 시스템들이 명시적으로 프로그래밍됨이 없이 작동하도록 유도하는 과학이다. 고전적인 머신 러닝은, K-평균 클러스터링, 선형 및 로지스틱 회귀, 확률적 경사 하강법, 연관 규칙 학습 등을 포함한, 다양한 클러스터링 및 분류 기술들을 포함한다. 딥 러닝은 머신 러닝에서의 더 새로운 미개척 분야이다. 딥 러닝은 특징 추출 및 변환을 위해 다수의 계층들의 비선형 프로세싱 유닛들을 사용하는 머신 러닝 알고리즘들의 한 부류이다. 딥 러닝 알고리즘들은 비지도(unsupervised)(예를 들어, 패턴 분석) 또는 지도(supervised)(예를 들어, 분류)일 수 있다. 딥 러닝 알고리즘은 인공 신경 네트워크(artificial neural network; ANN)(본 명세서에서 "신경 네트워크"라고 지칭됨)의 계층들을 사용하여 구현될 수 있다.

[0003] 일반적으로, 신경 네트워크는 그래프로 연결되는 노드들(즉, "뉴런들")의 집합체이다. 신경 네트워크 내의 노드는 가중 입력들의 합을 계산하고 합에 선택적인 바이어스를 가산한다. 노드의 출력은 최종 합(final sum)의 함수("활성화 함수"라고 지칭됨)이다. 예시적인 활성화 함수들은 시그모이드 함수, 쌍곡 탄젠트(hyperbolic tangent; tanh) 함수, 정류 선형 유닛(Rectified Linear Unit; ReLU) 함수, 및 항등 함수를 포함한다. 신경 네트워크 모델들은 종종, 특정 토폴로지를 정의하는 노드들의 계층들, 및 대응하는 가중치들 및 바이어스들로 구성된다. 가중치들 및 바이어스들은 네트워크 파라미터들이라고 지칭된다.

[0004] 일반적으로, 신경 네트워크는 입력 계층 및 출력 계층을 포함하고, 선택적으로 입력 계층과 출력 계층 사이에 하나 이상의 은닉 계층을 포함할 수 있다. 딥 러닝 애플리케이션들에서 사용되는 신경 네트워크는 전형적으로 많은 은닉 계층들을 포함하며, 이로부터 딥 신경 네트워크(deep neural network; DNN)라는 용어가 생긴다. 신경 네트워크의 계층들은 밀집 연결될(densely connected) 수 있거나(예를 들어, 계층에서의 각각의 노드가 이전 계층에서의 모든 노드들에 완전 연결됨(fully connected)) 또는 희소 연결될(sparsely connected) 수 있다(예를 들어, 계층에서의 각각의 노드가 이전 계층에서의 노드들 중의 일 부분에만 연결됨). 컨볼루션 신경 네트워크(convolutional neural network; CNN)는, 컨볼루션 계층들이라고 지칭되는, 하나 이상의 희소 연결 계층을 포함하는 DNN의 한 유형이다. CNN은 이미지 또는 비디오 데이터를 프로세싱하는 데 아주 적합하다. 다른 유형들의 DNN들은 음성 및 텍스트 데이터를 프로세싱하는 데 아주 적합한 순환 신경 네트워크(recurrent neural network; RNN)를 포함한다.

[0005] 최신의 필드 프로그래밍가능 게이트 어레이(field programmable gate array; FPGA)는 대규모 병렬 하드웨어 시스템들을 생성하는 데 이용될 수 있는 수백만개의 룩업 테이블 및 수천개의 디지털 신호 프로세싱(DSP) 및 랜덤 액세스 메모리 블록(BRAM)을 제공한다. FPGA 내의 프로그래밍가능 로직은 병렬 하드웨어 시스템들을 사용하여 신경 네트워크 가속기들(일반적으로 가속 회로들이라고 지칭됨)을 실행할 수 있는 하나 이상의 커널을 형성할 수 있다.

[0006] FPGA의 이용률(utilization)을 증가시키는 것은 신경 네트워크 애플리케이션의 성능을 개선시킬 수 있다. 따라

서, FPGA가 신경 네트워크 애플리케이션에 의해 제공되는 태스크들을 실행하는 데 많은 시간을 들일수록, 신경 네트워크는 결과들을 빠르게 프로세싱 및 제공할 수 있다. 그렇지만, 신경 네트워크 설계자들이 FPGA 상의 신경 네트워크 가속기들을 완전히 이용하는 데 필요한 요구된 역량(skill) 및 전문지식을 갖지 않을 수 있다.

발명의 내용

[0007] 신경 네트워크를 스케줄링하기 위한 기술들이 설명된다. 하나의 예는 신경 네트워크 가속기에 제출된 태스크들을 파이프라이닝하기 위한 방법이다. 이 방법은 신경 네트워크 가속기에 의해 프로세싱될 제1 태스크를 신경 네트워크 애플리케이션으로부터 수신하는 단계, 파이프라인 내의 다수의 스테이지들에 의해 사용되는 정보를 포함하는 패킷을 생성하는 단계, 및 다수의 스테이지들에서 패킷을 프로세싱하는 단계를 포함하고, 여기서 다수의 스테이지들 중 적어도 하나는 신경 네트워크 가속기를 실행하는 하드웨어 시스템에 대한 호출을 수행하며, 여기서 파이프라인은 패킷을 프로세싱하는 것과 병렬로 제2 태스크에 대응하는 적어도 하나의 다른 패킷을 프로세싱한다. 이 방법은 파이프라인을 사용하여 패킷을 프로세싱한 결과들을 신경 네트워크 애플리케이션에 반환하는 단계를 또한 포함한다.

[0008] 일부 실시예들에서, 다수의 스테이지들에서 패킷을 프로세싱하는 단계는 프리-프로세싱(pre-processing) 스테이지에서 패킷을 프로세싱하는 단계, 프리-프로세싱 스테이지 이후에 발생하는 실행 스테이지에서 패킷을 프로세싱하는 단계 - 하드웨어 시스템에 대한 호출은 실행 스테이지 동안 발생할 수 있음 -, 및 실행 스테이지 이후의 포스트-프로세싱(post-processing) 스테이지에서 패킷을 프로세싱하는 단계를 포함할 수 있다.

[0009] 일부 실시예들에서, 프리-프로세싱 스테이지에서 패킷을 프로세싱하는 단계는 제1 태스크에 대응하는 데이터를 신경 네트워크 애플리케이션에 의해 사용되는 제1 포맷으로부터 하드웨어 시스템에 의해 사용되는 제2 포맷으로 변환하는 단계를 포함할 수 있다. 포스트-프로세싱 스테이지에서 패킷을 프로세싱하는 단계는 결과들을 제2 포맷으로부터 제1 포맷으로 변환하는 단계를 포함할 수 있다.

[0010] 일부 실시예들에서, 다수의 스테이지들 각각은 다른 스레드들과 독립적으로 패킷을 프로세싱하는 각자의 스레드를 포함할 수 있다.

[0011] 일부 실시예들에서, 이 방법은 신경 네트워크 애플리케이션을 위한 할당된 메모리 블록들을 하드웨어 시스템 내의 신경 네트워크 가속기를 위한 할당된 메모리 블록들에 매핑하는 메모리 맵을 생성하는 단계 및 신경 네트워크 애플리케이션으로부터 수신되는 제1 메모리 어드레스들을 메모리 맵에 기초하여 하드웨어 시스템 내의 메모리 블록들에 대한 제2 메모리 어드레스들로 변환하는 단계를 추가로 포함할 수 있다.

[0012] 일부 실시예들에서, 이 방법은 신경 네트워크의 다수의 계층들을 수행하는 데 사용되는 가중치들을 행렬 포맷으로 하드웨어 시스템에게 전송하는 단계, 새로운 태스크에 대응하는 가중치들의 서브세트를 식별하는 단계 및 패킷을 프로세싱할 때 사용될 가중치들의 서브세트를 지시하는 오프셋을 하드웨어 시스템에게 전송하는 단계를 추가로 포함할 수 있다.

[0013] 일부 실시예들에서, 이 방법은 하드웨어 시스템 상의 신경 네트워크 가속기의 실행에 관한 메트릭을 획득하는 단계, 메트릭의 시각적 표현을 디스플레이를 위해 출력하는 단계 및 신경 네트워크 가속기의 이용률을 증가시키기 위해 파이프라인 내의 다수의 스테이지들을 실행하는 하드웨어 자원들을 조정하는 단계를 추가로 포함할 수 있다.

[0014] 일부 실시예들에서, 파이프라인 내의 다수의 스테이지들은 라이브러리에 정의되어 있을 수 있다. 라이브러리는 상이한 유형들의 신경 네트워크 애플리케이션들이 파이프라인 내의 다수의 스테이지들을 사용하여 태스크들을 신경 네트워크 가속기에 제출할 수 있게 해주도록 구성되는 애플리케이션 프로그램 인터페이스(application program interface; API)를 포함할 수 있다.

[0015] 일부 실시예들에서, 이 방법은 신경 네트워크 가속기에 정보를 제공하는 데 사용되는 패킷 내의 복수의 필드들을 커스터마이징하는 단계를 추가로 포함할 수 있다. 커스터마이징된 복수의 필드들은 신경 네트워크 가속기의 유형에 따라 달라질 수 있으며, 상이한 유형들의 신경 네트워크 가속기들은 상이한 필드들을 사용할 수 있다.

[0016] 일부 실시예들에서, 이 방법은 디버깅 함수가 활성인지 여부를 결정하고 신경 네트워크 가속기에 패킷을 제출하는 것과 신경 네트워크 애플리케이션을 실행하는 호스트 내의 하나 이상의 컴퓨터 프로세서에 패킷을 제출하는 것 사이에서 전환하는 단계를 추가로 포함할 수 있다.

[0017] 다른 예는, 하나 이상의 프로세싱 디바이스 상에서 실행될 때, 신경 네트워크 가속기에 제출된 태스크들을 파이

프라이닝하기 위한 동작을 수행하는 명령어들을 저장하는 비일시적 컴퓨터 판독가능 저장 매체이다. 이 동작은 신경 네트워크 가속기에 의해 프로세싱될 제1 태스크를 신경 네트워크 애플리케이션으로부터 수신하는 것, 파이프라인 내의 다수의 스테이지들에 의해 사용되는 정보를 포함하는 패킷을 생성하는 것, 및 다수의 스테이지들에서 패킷을 프로세싱하는 것을 포함하고, 여기서 다수의 스테이지들 중 적어도 하나는 신경 네트워크 가속기를 실행하는 하드웨어 시스템에 대한 호출을 수행하며, 여기서 파이프라인은 패킷을 프로세싱하는 것과 병렬로 제2 태스크에 대응하는 적어도 하나의 다른 패킷을 프로세싱한다. 이 동작은 파이프라인을 사용하여 패킷을 프로세싱한 결과들을 신경 네트워크 애플리케이션에 반환하는 것을 또한 포함한다.

[0018] 다른 예는 프로세서 및 메모리를 포함하는 컴퓨팅 시스템이다. 메모리는, 프로세서에 의해 실행될 때, 동작을 수행하는 라이브러리를 포함한다. 이 동작은 신경 네트워크 가속기에 의해 프로세싱될 제1 태스크를 신경 네트워크 애플리케이션으로부터 수신하는 것, 파이프라인 내의 다수의 스테이지들에 의해 사용되는 정보를 포함하는 패킷을 생성하는 것, 및 다수의 스테이지들에서 패킷을 프로세싱하는 것을 포함하고, 여기서 다수의 스테이지들 중 적어도 하나는 신경 네트워크 가속기를 실행하는 하드웨어 시스템에 대한 호출을 수행하며, 여기서 파이프라인은 패킷을 프로세싱하는 것과 병렬로 제2 태스크에 대응하는 적어도 하나의 다른 패킷을 프로세싱한다. 이 동작은 파이프라인을 사용하여 패킷을 프로세싱한 결과들을 신경 네트워크 애플리케이션에 반환하는 것을 또한 포함한다.

[0019] 일부 실시예들에서, 다수의 스테이지들에서 패킷을 프로세싱하는 것은 프리-프로세싱 스테이지에서 패킷을 프로세싱하는 것, 프리-프로세싱 스테이지 이후에 발생하는 실행 스테이지에서 패킷을 프로세싱하는 것 - 하드웨어 시스템에 대한 호출은 실행 스테이지 동안 발생할 수 있음 -, 및 실행 스테이지 이후의 포스트-프로세싱 스테이지에서 패킷을 프로세싱하는 것을 포함할 수 있다.

[0020] 일부 실시예들에서, 프리-프로세싱 스테이지에서 패킷을 프로세싱하는 것은 제1 태스크에 대응하는 데이터를 신경 네트워크 애플리케이션에 의해 사용되는 제1 포맷으로부터 하드웨어 시스템에 의해 사용되는 제2 포맷으로 변환하는 것을 포함할 수 있고, 포스트-프로세싱 스테이지에서 패킷을 프로세싱하는 것은 결과들을 제2 포맷으로부터 제1 포맷으로 변환하는 것을 포함할 수 있다.

[0021] 일부 실시예들에서, 이 동작은 신경 네트워크 애플리케이션을 위한 할당된 메모리 블록들을 하드웨어 시스템 내의 신경 네트워크 가속기를 위한 할당된 메모리 블록들에 매핑하는 메모리 맵을 생성하는 것 및 신경 네트워크 애플리케이션으로부터 수신되는 제1 메모리 어드레스들을 메모리 맵에 기초하여 하드웨어 시스템 내의 메모리 블록들에 대한 제2 메모리 어드레스들로 변환하는 것을 추가로 포함할 수 있다.

[0022] 일부 실시예들에서, 이 동작은 신경 네트워크의 다수의 계층들을 수행하는 데 사용되는 가중치들을 행렬 포맷으로 하드웨어 시스템에게 전송하는 것, 새로운 태스크에 대응하는 가중치들의 서브세트를 식별하는 것 및 패킷을 프로세싱할 때 사용될 가중치들의 서브세트를 지시하는 오프셋을 하드웨어 시스템에게 전송하는 것을 추가로 포함할 수 있다.

[0023] 일부 실시예들에서, 이 동작은 하드웨어 시스템 상의 신경 네트워크 가속기의 실행에 관한 메트릭을 획득하는 것, 메트릭의 시각적 표현을 디스플레이를 위해 출력하는 것 및 신경 네트워크 가속기의 이용률을 증가시키기 위해 파이프라인 내의 다수의 스테이지들을 실행하는 하드웨어 자원들을 조정하는 것을 추가로 포함할 수 있다.

[0024] 일부 실시예들에서, 파이프라인 내의 다수의 스테이지들은 라이브러리에 정의되어 있을 수 있다. 라이브러리는 상이한 유형들의 신경 네트워크 애플리케이션들이 파이프라인 내의 다수의 스테이지들을 사용하여 태스크들을 신경 네트워크 가속기에 제출할 수 있게 해주도록 구성되는 API를 포함할 수 있다.

[0025] 일부 실시예들에서, 이 동작은 신경 네트워크 가속기에 정보를 제공하는 데 사용되는 패킷 내의 복수의 필드들을 커스터마이징하는 것을 추가로 포함할 수 있다. 커스터마이징된 복수의 필드들은 신경 네트워크 가속기의 유형에 따라 달라질 수 있으며, 상이한 유형들의 신경 네트워크 가속기들은 상이한 필드들을 사용할 수 있다.

[0026] 일부 실시예들에서, 컴퓨팅 시스템은 디버깅 함수가 활성이라고 결정하고 신경 네트워크 가속기에 패킷을 제출하는 것과 실행을 위해 컴퓨팅 시스템 내의 프로세서에 패킷을 제출하는 것 사이에서 전환하는 것을 추가로 포함할 수 있다.

도면의 간단한 설명

[0027] 앞서 언급된 특징들이 상세하게 이해될 수 있도록, 앞서 간략하게 요약된 더 자세한 설명이 예시적인 구현들 - 그 일부가 첨부 도면들에 예시되어 있음 - 을 참조하여 이루어질 수 있다. 그렇지만, 첨부 도면들이 전형적인

예시적인 구현들만을 예시하고 따라서 그의 범위를 제한하는 것으로 간주되어서는 안된다는 것에 유의해야 한다.

도 1은 예에 따른, 다중 계층 신경 네트워크를 예시한다.

도 2는 예에 따른, 신경 네트워크 가속기를 신경 네트워크 어플리케이션과 인터페이싱시키기 위한 시스템이다.

도 3은 예에 따른, 신경 네트워크 가속기와 신경 네트워크 어플리케이션 사이의 통신 흐름을 예시한다.

도 4는 예에 따른, 신경 네트워크 가속기에서 실행하기 위해 신경 네트워크 애플리케이션으로부터 수신되는 태스크들을 파이프라이닝하는 것에 대한 플로차트이다.

도 5는 예에 따른, 신경 네트워크 애플리케이션에 의해 제출된 태스크들에 대한 파이프라인을 예시한다.

도 6은 예에 따른, 신경 네트워크 애플리케이션에 의해 제출된 태스크들에 대한 파이프라인의 실행을 조정하는 것에 대한 플로차트이다.

도 7은 예에 따른, 신경 네트워크 애플리케이션에 의해 제출된 태스크들을 파이프라이닝하는 것에 대응하는 타이밍 차트이다.

도 8은 예에 따른 신경 네트워크들을 구현하기 위한 시스템을 묘사하는 블록 다이어그램이다.

도 9는 예에 따른 컴퓨팅 시스템을 묘사하는 블록 다이어그램이다.

도 10은 예에 따른 가속 회로를 묘사하는 블록 다이어그램이다.

도 11은 예에 따른 프로그래밍가능 집적 회로(IC)를 묘사하는 블록 다이어그램이다.

도 12는 예에 따른 프로그래밍가능 IC의 FPGA(field programmable gate array) 구현을 예시한다.

이해를 용이하게 하기 위해, 가능한 경우, 도면들에 공통인 동일한 요소들을 표시하기 위해 동일한 참조 번호들이 사용되었다. 일 예의 요소들이 유리하게는 다른 예들에 포함될 수 있는 것이 생각되고 있다.

발명을 실시하기 위한 구체적인 내용

[0028] 다양한 특징들이 도면들을 참조하여 이하에서 설명된다. 도면들이 축척대로 그려져 있을 수 있거나 그렇지 않을 수 있다는 점과, 유사한 구조들 또는 기능들의 요소들이 도면들 전체에 걸쳐 비슷한 참조 번호들로 표현된다는 점에 유의해야 한다. 도면들이 특징들의 설명을 용이하게 하는 것으로만 의도된다는 점에 유의해야 한다. 이들은 이 설명의 총망라적 설명으로서 또는 청구항들의 범위에 대한 제한으로서 의도되지 않는다. 그에 부가하여, 예시된 예는 도시된 모든 양태들 또는 장점들을 가질 필요는 없다. 특정 예와 관련하여 설명된 양태 또는 장점이 반드시 그 예로 제한되는 것은 아니며, 그렇게 예시되지 않았더라도 또는 그렇게 명시적으로 설명되지 않았더라도 임의의 다른 예들에서 실시될 수 있다.

[0029] 본 명세서에서의 실시예들은 라이브러리를 사용하여 신경 네트워크 애플리케이션을 신경 네트워크 가속기와 인터페이싱시키기 위한 기술들을 설명한다. 예를 들어, 신경 네트워크 애플리케이션은 호스트 컴퓨팅 시스템 상에서 실행될 수 있는 반면, 신경 네트워크 가속기는 대규모 병렬 하드웨어 시스템, 예를 들어, FPGA, 그래픽 프로세싱 유닛(graphics processing unit; GPU), 또는 특수 설계된 애플리케이션 특정 집적 회로(ASIC) 상에서 실행된다. 일 실시예에서, (가속기와 신경 네트워크 애플리케이션 사이의 인터페이스 엔진 또는 어댑터라고도 지칭될 수 있는) 라이브러리는 신경 네트워크 애플리케이션으로부터 수신되는 태스크들을 신경 네트워크 가속기에 제출하기 위한 파이프라인을 동작시키는 코드를 포함한다. 일 실시예에서, 파이프라인은 각각이 상이한 스레드들에 대응하는 프리-프로세싱 스테이지, FPGA 실행 스테이지, 및 포스트-프로세싱 스테이지를 포함한다. 신경 네트워크 애플리케이션으로부터 태스크를 수신할 때, 라이브러리는 태스크들을 수행하기 위해 파이프라인 내의 상이한 스테이지들에 대해 요구되는 정보를 포함하는 패킷을 생성한다. 스테이지들이 상이한 스레드들에 대응하기 때문에, 라이브러리는 다수의 패킷들을 병렬로 프로세싱할 수 있다. 즉, 라이브러리는 프리-프로세싱 스테이지에서 제1 패킷을 프로세싱할 수 있는 반면, 제2 패킷은 실행 스테이지에 있고, 제3 패킷은 포스트-프로세싱 스테이지에 있다. 그렇게 하는 것은 대규모 병렬 하드웨어 시스템 상의 신경 네트워크 가속기의 이용률을 증가시킬 수 있다. 즉, 각각의 프로세싱 사이클 동안, 라이브러리는 실행 스테이지에서 상이한 패킷(또는 태스크)을 제출하여, 이에 의해 신경 네트워크 가속기가 새로운 데이터를 기다리는 또는 신경 네트워크 가속기가 라이브러리가 이미 프로세싱된 데이터를 검색하기를 기다리는 다운시간을 최소화할 수 있다.

[0030] 일 실시예에서, 라이브러리는 신경 네트워크 애플리케이션에 할당된 호스트 컴퓨팅 시스템 내의 메모리 블록들과 신경 네트워크 가속기에 할당된 대규모 병렬 하드웨어 시스템 내의 메모리 블록들 사이의 메모리 맵을 유지한다. 라이브러리는 호스트 컴퓨팅 시스템 내의 메모리 블록들에 대한 변경들을 검출하고 그 변경들을 대규모 병렬 하드웨어 시스템 내의 메모리 블록들에 자동으로 미러링할 수 있다. 다른 예에서, 라이브러리는 대규모 병렬 하드웨어 시스템 내의 메모리 블록들에 대한 하나의 기입을 수행하고 이어서 2개의 시스템 사이에서 전송되는 데이터의 양을 감소시키기 위해 메모리 맵에 따른 오프셋 어드레스를 사용할 수 있다.

[0031] 다른 실시예에서, 라이브러리는 신경 네트워크 가속기의 이용률에 관한 메트릭들을 제공한다. 이러한 메트릭들은 신경 네트워크의 조작자에게 실시간으로(또는 지연을 갖고) 시각적으로 출력될 수 있고, 조작자는 그러면 파이프라인에 대한 조정들을 식별할 수 있다. 예를 들어, 다음 패킷이 프리-프로세싱 스테이지로부터 이용가능하기 전에 신경 네트워크 가속기가 실행 스테이지 동안 패킷을 프로세싱하는 것을 완료할 수 있다면, 조작자는 파이프라인의 전체적인 실행을 가속시키기 위해 프리-프로세싱 스테이지를 실행하는 스레드에 부가의 하드웨어 자원들(예를 들어, 더 많은 프로세싱 능력)을 할당할 수 있다.

[0032] 도 1은 예에 따른, 다중 계층 신경 네트워크(100)를 예시한다. 본 명세서에서 사용되는 바와 같이, 신경 네트워크(100)는 머신 러닝에서 사용되는 계산 모듈이고, 인공 뉴런들이라고 불리는 연결된 유닛들의 대규모 집합체에 기초하며 여기서 뉴런들 사이의 연결들은 다양한 강도의 활성화 신호를 운반한다. 신경 네트워크(100)는 명시적으로 프로그래밍되는 것보다는 예들로부터 트레이닝될 수 있다. 일 실시예에서, 신경 네트워크(100)에서의 뉴런들은 계층들 - 예를 들어, 계층 1, 계층 2, 계층 3 등 - 을 이루어 연결되며, 여기서 데이터는 제1 계층 - 예를 들어, 계층 1 - 으로부터 마지막 계층 - 예를 들어, 계층 7 - 으로 이동한다. 비록 도 1에 7개의 계층이 도시되어 있지만, 신경 네트워크(100)는 수백 또는 수천개의 상이한 계층을 포함할 수 있다.

[0033] 신경 네트워크들은 컴퓨터 비전, 특징 검출, 음성 인식 등과 같은 임의의 개수의 태스크들을 수행할 수 있다. 도 1에서, 신경 네트워크(100)는 이미지 내의 객체들을 분류하는 것, 얼굴 인식을 수행하는 것, 텍스트를 식별하는 것 등과 같이 디지털 이미지에서 특징들을 검출한다. 그렇게 하기 위해, 이미지 데이터(101)는 이미지 데이터(101)에 대해 대응하는 함수, 이 예에서, 10x10 컨볼루션을 수행하는 신경 네트워크에서의 제1 계층에 공급된다. 그 함수의 결과들은 이어서 프로세싱된 이미지 데이터를 다음 레벨에 전달하기 전에 자신의 함수를 수행하는 다음 계층 - 예를 들어, 계층 2 - 에 전달되고, 이하 마찬가지이다. 계층들에 의해 프로세싱된 후에, 데이터는 이미지 데이터에서 특징들을 검출할 수 있는 이미지 분류기(102)에 수신된다.

[0034] 계층 1이 계층 2 이전에 수행되고, 계층 2가 계층 3 이전에 수행되며, 이하 마찬가지이도록 계층들이 순차적 순서로 정의된다. 따라서, 하위 계층들과 상위 계층(들) 사이에 데이터 의존성이 존재한다. 비록 계층 2가 계층 1로부터 데이터를 수신하기 위해 기다리지만, 일 실시예에서, 신경 네트워크(100)는 각각의 계층이 동시에 동작할 수 있도록 병렬화될 수 있다. 즉, 각각의 클록 사이클 동안, 계층들은 새로운 데이터를 수신하고 프로세싱된 데이터를 출력할 수 있다. 예를 들어, 각각의 클록 사이클 동안, 새로운 이미지 데이터(101)가 계층 1에 제공될 수 있다. 간략함을 위해, 각각의 클록 사이클 동안 새로운 이미지가 계층 1에 제공되고 각각의 계층이 이전 클록 사이클에서 수신된 이미지 데이터에 대한 프로세싱된 데이터를 출력할 수 있다고 가정한다. 병렬화된 파이프라인을 형성하기 위해 계층들이 하드웨어로 구현되는 경우, 7개의 클록 사이클 후에, 계층들 각각은 (7개의 상이한 이미지에 대해서라도) 이미지 데이터를 프로세싱하기 위해 동시에 동작한다. 따라서, 병렬 파이프라인을 형성하기 위해 계층들을 하드웨어로 구현하는 것은 계층들을 한 번에 하나씩 동작시키는 것과 비교할 때 신경 네트워크의 처리량을 크게 증가시킬 수 있다. 신경 네트워크(100)에서의 계층들의 개수가 증가함에 따라 대규모 병렬 하드웨어 시스템에서 계층들을 스케줄링하는 것의 타이밍 이점들이 더욱 개선된다.

[0035] 도 2는 예에 따른, 신경 네트워크 가속기(165)를 신경 네트워크 어플리케이션(120)과 인터페이싱시키기 위한 시스템(200)이다. 시스템(200)은 호스트(105)(예를 들어, 호스트 컴퓨팅 시스템) 및 FPGA(150)를 포함한다. 비록 FPGA가 구체적으로 도시되어 있지만, 본 명세서에서의 실시예들은 라이브러리(130)를 사용하여 임의의 유형의 하드웨어 시스템 - 예를 들어, GPU 또는 ASIC - 상에서 호스팅되는 신경 네트워크 가속기(165)(예를 들어, 가속 회로 또는 커널 가속기 회로)를 신경 네트워크 애플리케이션(120)과 인터페이싱시키는 데 사용될 수 있다.

[0036] 호스트(105)는 프로세서(110) 및 메모리(115)를 포함한다. 프로세서(110)는 각각이 임의의 개수의 프로세싱 코어들을 포함할 수 있는 임의의 개수의 프로세싱 요소들을 나타낸다. 메모리(115)는 휘발성 메모리 요소들, 비휘발성 메모리 요소들, 및 이들의 조합들을 포함할 수 있다. 더욱이, 메모리(115)는 상이한 매체들(예를 들어, 네트워크 스토리지 또는 외부 하드 드라이브들)에 걸쳐 분산될 수 있다.

[0037] 메모리(115)는 일 실시예에서 프로세서(110)에 의해 실행되는 소프트웨어 애플리케이션인 신경 네트워크 애플리

케이션(120)을 포함하지만; 다른 예들에서, 신경 네트워크 애플리케이션(120)은 하드웨어 요소들을 포함할 수 있다. 신경 네트워크 애플리케이션(120)은 상이한 함수들 - 예를 들어, 컨볼루션, 맥스 풀링(Max-pooling), im2col, 행렬 곱셈 등 - 을 수행하는 임의의 개수의 계층들을 가질 수 있는 신경 네트워크 - 예를 들어, 도 1에 도시된 신경 네트워크(100) - 를 구축한다. 비록 도시되어 있지는 않지만, 신경 네트워크 애플리케이션(120)은 메모리(115)에 저장된 또는 외부 소스들로부터의 데이터(예를 들어, 이미지 또는 오디오 데이터)를 프로세싱하기 위해 신경 네트워크를 사용할 수 있다. 예를 들어, 호스트(105)는 사용자들이 이미지들을 제출하도록 허용하는 웹 포털에 통신가능하게 커플링될 수 있고 이미지들은 이어서 신경 네트워크 애플리케이션(120)에 의해 프로세싱된다.

[0038] 이하의 실시예들에서, 신경 네트워크 애플리케이션(120)은 신경 네트워크의 성능을 개선시킬 수 있는 - 예를 들어, 신경 네트워크가 프로세서(110)에만 의존하는 것보다 복수의 계층들을 더 빨리 실행할 수 있게 해줄 수 있는 - FPGA(150) 상의 신경 네트워크 가속기들(165)에 통신가능하게 커플링된다. 그렇지만, 신경 네트워크 어플리케이션(120)은 신경 네트워크 가속기(165)와 상이한 포맷을 사용하여 데이터를 프로세싱할 수 있다. 게다가, 호스트(105) 및 FPGA(150)의 메모리들은 상이한 비-코히런트(non-coherent) 메모리들일 수 있다.

[0039] 라이브러리(130)는 신경 네트워크 애플리케이션(120)을 신경 네트워크 가속기(165)에 통신가능하게 커플링시키기 위한 방법들 및 동작들을 제공한다. 라이브러리는 신경 네트워크 애플리케이션(120)을 위한 할당된 메모리 블록들(125)을 FPGA(150) 내의 신경 네트워크 가속기들(165)을 위한 할당된 메모리 블록들(175)에 매핑하는 메모리 맵(140)(예를 들어, 데이터 구조 또는 데이터베이스)을 포함한다. 일 실시예에서, 호스트(105)는 프로세싱될 상이한 이미지들을 저장하는 신경 네트워크 애플리케이션(120)을 위한 큰 메모리 청크(즉, 할당된 블록들(125))를 할당할 수 있다. 예를 들어, 상이한 이미지들을 프로세싱할 때, 신경 네트워크 애플리케이션(120)은 특정의 이미지가 저장되는 할당된 블록들(125)에 대한 오프셋을 라이브러리(130)에게 전송할 수 있다. 메모리 맵(140)을 사용하여, 라이브러리는 FPGA(150) 내의 대응하는 할당된 블록 또는 블록들(175)을 식별할 수 있다. 일 실시예에서, 동일한 이미지 또는 오디오 파일에 대응하는 데이터가 호스트(105) 내의 메모리(115)와 FPGA(150) 내의 메모리(170) 사이에서 올바르게 상관될 수 있도록, 라이브러리(130)는 호스트(105) 및 FPGA(150)가 블록들(125) 및 블록들(175)을 할당한 후에 메모리 맵(140)을 생성한다.

[0040] 그에 부가하여, 라이브러리(130)는 신경 네트워크 가속기(165)에 의해 완료되도록 신경 네트워크 애플리케이션(120)에 의해 제출된 태스크들을 프로세싱하는 파이프라인 스테이지들(135)을 포함한다. 즉, 신경 네트워크 애플리케이션(120)으로부터 태스크를 수신하고, 태스크를 신경 네트워크 가속기(165)에 제출하며, 결과들을 기다리는 대신에, 라이브러리(130)는 상이한 스테이지들에서 다수의 태스크들을 병렬로 프로세싱하기 위해 파이프라인 스테이지들(135)을 사용한다. 일 실시예에서, 새로운 태스크를 수신할 때, 라이브러리(130)는 태스크를 완료하는 데 사용되는 데이터를 포함하는 패킷을 생성한다. 일 실시예에서, 각각의 패킷이 파이프라인에서의 다른 패킷들에 대한 데이터 의존성을 갖지 않으면서 스테이지들(135)에서 개별적으로 프로세싱될 수 있도록 패킷들은 자립적(self-contained)이다. 라이브러리(130)가 3개의 스테이지(135)(예를 들어, 프리-프로세싱 스테이지, FPGA 실행 스테이지, 및 포스트-프로세싱 스테이지)를 갖는 파이프라인을 형성하는 경우, 라이브러리(130)는 3개의 스테이지를 사용하여 병렬로 (각각이 애플리케이션(120)에 의해 제출된 상이한 태스크에 대응할 수 있는) 3개의 패킷을 프로세싱할 수 있다. 그렇게 하는 것은 신경 네트워크 가속기의 이용률 및 신경 네트워크의 전체적인 런타임을 증가시킬 수 있다.

[0041] 일 실시예에서, 라이브러리(130)는 신경 네트워크 조작자가 FPGA(150) 내의 신경 네트워크 가속기(165)를 구성하는 방법 또는 효율적으로 실행하는 방법을 알 필요 없이 조작자가 신경 네트워크 가속기(165)를 사용할 수 있게 해준다. 즉, 조작자는 FPGA(150) 내의 프로그래밍가능 로직(155)을 구성하는 데 전형적으로 사용되는 레지스터 전송 로직(register transfer logic; RTL)을 이해할 필요가 없다. 그 대신에, 라이브러리(130)는 파이프라인 스테이지들(135) 및 메모리 맵(140)을 사용하여 신경 네트워크 애플리케이션(120)과 신경 네트워크 가속기(165) 사이의 통신을 추상화한다. 더욱이, 라이브러리(130)는 신경 네트워크 가속기들(165)과 통신하기 위해 상이한 유형들의 신경 네트워크들(및 신경 네트워크 애플리케이션들)과 함께 사용될 수 있는 일반(generic) 애플리케이션 프로그램 인터페이스(API)를 제공할 수 있다.

[0042] 메모리(115)는 호스트(105)와 FPGA(150) 사이의 통신을 가능하게 해주는 FPGA 드라이버(145)를 또한 포함한다. 일 실시예에서, FPGA 드라이버(145)는 라이브러리(130) 및 그의 대응하는 함수들 및 연산자들이 FPGA(150)와 통신할 수 있게 해준다. 더욱이, FPGA 드라이버(145)는 프로그래밍가능 로직(155) 및 메모리(170)의 이용률과 같은 FPGA(150) 내의 하드웨어에 관한 메트릭들을 수신(또는 요청)할 수 있다. 일 실시예에서, 라이브러리(130)는 처리량을 증가시키도록 파이프라인 스테이지들(135)을 조정할 때 조작자에 도움을 줄 수 있는 신경 네트워크

가속기(165)의 이용률의 시각적 표현을 출력하기 위해 이러한 메트릭들을 사용할 수 있다.

[0043] FPGA(150)는 프로그래밍가능 로직(155) 및 메모리(170)를 포함한다. 프로그래밍가능 로직(155)은 프로그래밍가능 로직 블록들의 어레이 및 로직 블록들이 통신가능하게 커플링될 수 있게 해주는 재구성가능 인터커넥트들의 계층구조를 포함할 수 있다. 도 2에서, 프로그래밍가능 로직(155)은 각각이 하나 이상의 신경 네트워크 가속기(165)를 실행할 수 있는 하나 이상의 커널(160)을 형성한다. 일 예에서, 신경 네트워크 가속기들(165)은 신경 네트워크에 대한 컨볼루션들을 수행할 때 유용한 DSP 블록들을 포함한다. 다른 실시예에서, 컨볼루션을 수행하기 위해 행렬 곱셈이 사용될 수 있도록 가속기(165)는 수신된 이미지 데이터를 2D 행렬로 변환한다(im2col이라고 지칭됨). 그렇지만, 신경 네트워크 애플리케이션(120)은, 이미지가 스케일링될 때 특징들이 손실되지 않도록 이미지에서의 특징들을 증폭하는 맥스 풀링, 활성화 함수 또는 램프 함수(ramp function)인 정류 선형 유닛(ReLU) 등과 같은, 다른 유형들의 신경 네트워크 함수를 신경 네트워크 가속기(165)에 오프로딩할 수 있다.

[0044] 메모리(170)는 DDR RAM과 같은 휘발성 및 비휘발성 메모리 요소들을 포함할 수 있다. 신경 네트워크 애플리케이션(120)과의 통신을 구축할 때, FPGA(150)는 할당된 블록들(175)을 신경 네트워크 가속기들(165)에 할당한다. 그렇지만, FPGA(150) 내의 메모리(170)는 호스트(105) 내의 메모리(115)와 공유되지 않을 수 있기 때문에, 할당된 블록들(175)에 대한 어드레스들은 호스트(105) 내의 할당된 블록들(125)의 어드레스들과 부합(correspond)하지 않는다. 더욱이, 할당된 블록들(125 및 175)은 메모리에서 연속적인 블록들이 아닐 수 있거나 또는 상이한 때에 할당될 수 있다. 위에서 언급된 바와 같이, 라이브러리(130)는 할당된 블록들(125 및 175) 내의 개별 블록들을 서로에 매핑할 수 있는 메모리 맵(140)을 포함한다. 따라서, 라이브러리(130)가 할당된 블록들(125) 내의 어드레스 A에 위치된 이미지에 대한 태스크를 수신할 때, 라이브러리(130)는 그 태스크를 수행하기 위해 그 어드레스를 할당된 블록들(175) 내의 어드레스 B로 변환할 수 있다. 유사하게, 할당된 블록들(175)로부터 결과들을 판독할 때, 라이브러리(130)는 결과들을 할당된 블록들(125)에 저장하기 위해 어드레스를 대응하는 목적지 어드레스에 매핑할 수 있다.

[0045] 도 3은 예에 따른, 신경 네트워크 애플리케이션(120)과 신경 네트워크 가속기(165) 사이의 통신 흐름(300)을 예시한다. 도시된 바와 같이, 라이브러리(130) 및 FPGA 드라이버(145)는 통신 흐름(300)에서 신경 네트워크 애플리케이션(120)과 신경 네트워크 가속기(165) 사이에 있다. 따라서, 신경 네트워크 애플리케이션(120)은 태스크들을 라이브러리(130)에 제출하고, 라이브러리(130)는 차례로 FPGA 드라이버(145)를 사용하여 판독/기입 커맨드들을 생성하고 데이터를 신경 네트워크 가속기(165)에게 전송한다.

[0046] 신경 네트워크 가속기(165)는 신경 네트워크 애플리케이션(120)에 의해 할당된 태스크를 완료하기 위해 im2col(205), 행렬 곱셈(210), ReLU(215), 및 맥스 풀링(220)과 같은 다양한 연산들을 수행할 수 있다. 일 실시예에서, 신경 네트워크 애플리케이션(120)은 신경 네트워크의 단일 계층을 수행 - 예를 들어, 10x10 컨볼루션을 수행하거나 맥스 풀링을 수행 - 하기 위해 신경 네트워크 가속기(165)에 대한 태스크를 제출한다. 네트워크 가속기(165)는 또한 별도의 im2col/행렬 곱셈 단계들을 거치지 않고 직접적으로 컨볼루션과 같은 연산들을 수행할 수 있다. 다른 실시예에서, 신경 네트워크 애플리케이션(120)은 이미지별로 태스크를 제출할 수 있고, 이 경우에 라이브러리(130) 및 FPGA 드라이버(145)는 신경 네트워크 내의 계층들 중 선택된 하나의 계층보다는 신경 네트워크 내의 모든 계층들을 수행하여 이미지를 프로세싱하도록 신경 네트워크 가속기(165)에 지시한다.

[0047] 신경 네트워크 가속기(165)는 im2col(205), 행렬 곱셈(MM)(210), ReLU(215), 맥스 풀링(220) 등과 같은 하나 이상의 함수를 수행하기 위한 로직(예를 들어, FPGA 상에서 구현되는 경우 프로그래밍가능 로직)을 포함한다. 일 실시예에서, 신경 네트워크 가속기(165)를 형성하는 로직 블록들이 병렬로 실행될 수 있도록 이러한 함수들이 파이프라이닝될 수 있다. 즉, 신경 네트워크 가속기(165) 내의 하드웨어 로직은 라이브러리(130)에 의해 제공되는 함수들과 함께 병렬화될 수 있다.

[0048] 역방향으로 이동하면서, 태스크들을 프로세싱한 후에, 신경 네트워크 가속기(165)는 프로세싱된 데이터를 FPGA 드라이버(145)에게 전송하고, FPGA 드라이버(145)는 이 데이터를 라이브러리(130)에게 포워딩한다. 하나 이상의 파이프라인 스테이지를 사용하여, 라이브러리(130)는 결과들을 프로세싱하고 결과들을 신경 네트워크 애플리케이션(120)에게 전송한다. 일 실시예에서, 라이브러리(130)는 신경 네트워크 애플리케이션(120)으로부터 가속기(165)로 데이터를 전송할 때는 물론 가속기(165)로부터 애플리케이션(120)으로 결과들을 전송할 때 포맷을 변경한다. 라이브러리(130)는 다수의 태스크들을 동시에 프로세싱하기 위해 파이프라인 스테이지들을 사용할 수 있다.

[0049] 도 4는 예에 따른, 신경 네트워크 가속기에서 실행하기 위해 신경 네트워크 애플리케이션으로부터 수신되는 태스크들을 파이프라이닝하기 위한 방법(400)의 플로차트이다. 블록(405)에서, 라이브러리는 신경 네트워크 애플

리케이션으로부터 새로운 태스크를 수신한다. 일 실시예에서, 태스크는 신경 네트워크 애플리케이션이 신경 네트워크 가속기가 프로세싱하기를 원하는 데이터를 저장하는 호스트 메모리에서의 메모리 어드레스를 포함한다. 메모리 어드레스는 신경 네트워크 애플리케이션에 할당된 메모리 블록에 대한 시작 메모리 어드레스에 대한 오프셋일 수 있다. 예를 들어, 호스트는 큰 메모리 블록을 신경 네트워크 애플리케이션에 할당하고 이어서 메모리 오프셋을 사용하여 메모리 내의 서브블록들을 참조할 수 있다.

[0050] 일 실시예에서, 라이브러리 내의 메모리 맵은 호스트 내의 할당된 메모리의 서브블록들을 신경 네트워크 가속기를 실행하는 FPGA 내의 대응하는 메모리 블록들에 매핑하는 포인터들을 저장한다. 예를 들어, 신경 네트워크 애플리케이션은 서브블록들 중 하나에 저장된 특정 이미지를 참조할 수 있다. 메모리 맵을 사용하여, 라이브러리는 FPGA 내의 대응하는 어드레스 블록을 식별할 수 있다. 따라서, 메모리 맵은 신경 네트워크 애플리케이션에 할당된 호스트 내의 메모리를 신경 네트워크 가속기에 할당되는 FPGA 내의 메모리에 동기화시킬 수 있다. 일 실시예에서, 라이브러리는 신경 네트워크 애플리케이션에 할당된 메모리에서 이루어진 임의의 변경들을 신경 네트워크 가속기에 할당된 메모리에 미러링할 수 있다. 그렇지만, 라이브러리는 메모리에서의 변경들이 상대방 플랫폼으로 전파되는시기와 경우를 선택할 수 있다. DDR 메모리 전송은 비용이 많이 들기 때문에, 라이브러리는 호스트로의 DDR 전송들을 최소화하고 데이터를 가능한 한 FPGA 상에 유지할 수 있다. 예를 들어, 신경 네트워크 가속기가 FPGA 상에서 4개의 컨볼루션을 순차적으로(예를 들어, conv1->conv2->conv3->conv4) 실행하는 경우, 라이브러리는 컨볼루션 연산들의 모든 입력들/출력들을 자동으로 동기화시키지 않고 그 대신에 컨볼루션 연산들 중 첫 번째 컨볼루션 연산과 마지막 컨볼루션 연산만을 호스트 내의 메모리에 동기화시킬 수 있다. 즉, 호스트 내의 메모리에서 이루어진 임의의 변경들이 FPGA 내의 대응하는 메모리로 자동으로 전파되며 그 반대도 마찬가지이다. 일 실시예에서, 라이브러리는 호스트 및 FPGA에서 메모리들이 할당될 때 메모리들을 매핑하는 데 사용되는 메모리 맵 및 포인터들을 채울 수 있다.

[0051] 더욱이, 라이브러리는 호스트와 FPGA 사이에서 데이터를 전송하는 데 상당한 양의 시간을 소비할 수 있다. 신경 네트워크들은 네트워크 내의 뉴런들 사이의 연결들의 진폭을 특성화하는 데 가중치들을 사용한다. 신경 네트워크의 각각의 계층에 사용되는 가중치들은 상이할 수 있다. 따라서, 태스크를 신경 네트워크 가속기에게 전송할 때, 신경 네트워크 애플리케이션은 계층 또는 계층들에 대한 가중치들도 전송할 수 있다. 신경 네트워크 가속기는 신경 네트워크를 여러 번(예를 들어, 새로운 이미지가 수신될 때마다) 실행할 수 있으며, 이는 계층 또는 계층들이 실행될 때마다 라이브러리가 가중치들을 FPGA에게 전송한다는 것을 의미할 수 있다. 그 대신에, 일 실시예에서, 라이브러리는 신경 네트워크 가속기에 의해 수행되는 계층들에 대한 가중치들을 하나의 전송에서 송신한다. 일 실시예에서, 라이브러리는 가중치들을 큰 행렬로 FPGA에게 송신한다. 라이브러리가 상이한 가중치들을 갖는 상이한 계층을 요구하는 새로운 태스크를 수신할 때, 가중치들을 전송하는 대신에, 라이브러리는 가중치들의 어느 서브세트가 태스크를 수행하는 데 사용되어야 하는지를 식별해주는 행렬에 대한 오프셋을 전송할 수 있다. 이러한 방식으로, 라이브러리는 하나의 전송에서 가중치들을 전송하고 이어서 오프셋 메모리 어드레스들을 사용하여 특정의 태스크에 대한 행렬 내의 관련 가중치들을 식별할 수 있다.

[0052] 일 실시예에서, 라이브러리는 메모리 맵에서의 플래그들을 사용하여 어떤 데이터가 FPGA에게 전송되었는지를 식별한다. 예를 들어, 가중치들을 FPGA에게 전송한 후에, 라이브러리는 가중치들을 저장하는 호스트 내의 메모리 어드레스를 플래깅할 수 있다. 따라서, 신경 네트워크 어플리케이션이 플래깅된 메모리 어드레스를 라이브러리에게 송신할 때마다, 라이브러리는 그 메모리 어드레스에 저장된 데이터가 FPGA에게 이전에 송신되었다고 결정할 수 있다. 데이터를 재송신하는 대신에, 라이브러리는 FPGA에서 데이터가 저장되어 있는 대응하는 어드레스만을 송신할 수 있다. 가중치들에 부가하여, 라이브러리는 FPGA에게 이전에 전송된 이미지 데이터 또는 오디오 데이터를 표시하기 위해 플래그들을 사용할 수 있다.

[0053] 일 실시예에서, 메모리가 할당될 때, 라이브러리는 새로운 메모리 블록들을 사용된 세트(used set)에 저장하고, 메모리를 해제(free)하도록 지시받을 때, 라이브러리는 블록들을 사용된 세트로부터 사용되지 않은 세트(unused set)로 이동시킨다. 가능할 때마다, 라이브러리는, 상이한 메모리를 할당하기 전에, 사용되지 않은 메모리 블록들을 재사용하려고 시도한다. 달리 말하면, 메모리를 할당하기 위한 요청을 수신할 때, 라이브러리는 먼저 사용되지 않은 세트로부터의 메모리를 할당하려고 시도하는데 그 이유는 이것이 정렬된 메모리 블록들로부터의 메모리 단편화(memory fragmentation)의 발생을 감소시킬 수 있기 때문이다. 더욱이, 그렇게 하는 것은 각각의 순방향 전파(forward propagation)에서 할당된 메모리의 크기 및 패턴이 동일할 수 있는 딥 신경 네트워크들의 성질을 이용한다.

[0054] 블록(410)에서, 라이브러리는 태스크를 수행하기 위해 파이프라인 스테이지들에 의해 사용되는 정보를 포함하는 제1 데이터 패킷을 생성한다. 일 실시예에서, 데이터 패킷은 파이프라인 스테이지들 각각이 자신들의 작업들을

수행하는 데 필요한 모든 정보를 포함한다. 예를 들어, 각각의 스테이지는 패킷을 프로세싱할 때 패킷 내의 상이한 필드 또는 필드들을 사용할 수 있다. 따라서, 하나의 스테이지는 패킷 내의 제1 필드는 사용하지만 제2 필드는 사용할 수 없는 반면 다른 스테이지는 제2 필드는 사용하지만 제1 필드는 사용하지 않는다. 라이브러리가 채울 수 있는 패킷 내의 필드들의 비제한적인 예는 행렬 차원들, 가중치들에 대한 메모리 오프셋, 프로세싱될 데이터의 메모리 어드레스, 결과들이 저장되어야 하는 메모리 어드레스, 호스트 메모리에 대한 메모리 오프셋들 등을 포함한다. 일 실시예에서, 라이브러리는 상이한 신경 네트워크 가속기들(예를 들어, 상이한 커널들)에 대한 필드들을 커스터마이징할 수 있다. 즉, 상이한 유형들의 가속기들 또는 상이한 커널들은 상이한 필드들을 사용할 수 있다.

[0055] 블록(415)에서, 라이브러리는 파이프라인 내의 각각의 스테이지에 대응하는 스레드를 사용하여 제1 데이터 패킷을 프로세싱한다. 일 실시예에서, 각각의 스레드(즉, 각각의 스테이지)는 다른 스레드들이 다른 패킷들을 프로세싱하는 것과 동시에 패킷을 프로세싱할 수 있다. 이러한 방식으로, 파이프라인은 상이한 태스크들에 대응하는 상이한 패킷들을 병렬로 프로세싱할 수 있다. 더욱이, 하나의 스트림으로부터의 패킷들이 다른 스트림들로부터의 패킷들과 병렬로 프로세싱될 수 있다. 예를 들어, 신경 네트워크 애플리케이션은 웹 포털로부터의 제1 이미지 스트림 및 로컬 메모리에 저장된 제2 이미지 스트림을 수신할 수 있다. 라이브러리는 파이프라인 스테이지들에 의해 병렬로 프로세싱될 수 있는 2개의 스트림 내의 이미지들로부터 독립적인 패킷들을 생성할 수 있다. 즉, 패킷들이 자립적이거나 서로 독립적이기 때문에, 파이프라인은 상이한 소스들로부터의 패킷들을 병렬로 프로세싱할 수 있다.

[0056] 도 5는 예에 따른, 신경 네트워크 애플리케이션에 의해 제출된 태스크들에 대한 파이프라인(500)을 예시한다. 파이프라인(500)은 3개의 스테이지: 프리-프로세싱 스테이지, FPGA 실행 스테이지, 및 포스트-프로세싱 스테이지를 포함한다. 각각의 패킷은 각각의 스테이지를 통과하지만, 다른 실시예들에서, 일부 패킷들은 스테이지들 중 하나 이상의 스테이지를 스킵할 수 있다. 예를 들어, 라이브러리가 프리-프로세싱 스테이지 및 포스트-프로세싱 스테이지를 수행할 필요가 없도록 하나의 패킷 내의 데이터가 포맷팅될 수 있다.

[0057] 도 5에서, 4개의 패킷이 시간 기간 A 내지 시간 기간 F 동안 파이프라인(500)에 의해 프로세싱된다. 시간 기간 A 동안, 패킷 A는 프리-프로세싱 스테이지에 대응하는 스레드에 의해 프로세싱된다. 시간 기간 B 동안, (패킷 A를 프로세싱하는 것을 완료한) 프리-프로세싱 스테이지는 패킷 B를 프로세싱하는 반면 FPGA 실행 스테이지는 패킷 A를 프로세싱한다. 시간 기간 C 동안, FPGA 실행 스테이지가 패킷 B를 프로세싱하는 것 및 포스트-프로세싱 스테이지가 패킷 C를 프로세싱하는 것과 병렬로 프리-프로세싱 스테이지는 패킷 C를 프로세싱한다. 새로운 태스크들이 신경 네트워크 애플리케이션으로부터 수신되는 한, 파이프라인(500)은 각각의 시간 기간 동안 3개의 패킷을 병렬로 프로세싱할 수 있다.

[0058] 일 실시예에서, 시간 기간의 지속시간은 가장 오랫 동안 실행되는 스테이지에 따라 설정된다. 달리 말하면, 라이브러리는, 패킷들을 파이프라인 내의 다음 스테이지로 전달하고 후속 시간 기간을 시작하기 전에, 그 시간 기간 동안의 모든 스테이지들이 완료될 때까지 대기한다. 예를 들어, 프리-프로세싱 스테이지가 완료하는 데 가장 오래 걸리는 경우, 라이브러리는 다음 시간 기간으로 이동하기 전에 이 스테이지가 완료될 때까지 대기하며, 이는 FPGA 실행 스테이지 및 포스트-프로세싱 스테이지가 이미 완료되어 유휴 상태(idle)임을 의미한다. 일 실시예에서, 스테이지들의 지속시간이 달라질 수 있으며, 이는 시간 기간들의 지속시간들을 변화시킬 수 있다. 즉, 시간 기간 C 동안에는, FPGA 실행 스테이지가 실행되는 데 가장 오래 걸릴 수 있지만, 시간 기간 D 동안에는 포스트-프로세싱 스테이지가 실행되는 데 가장 오래 걸린다.

[0059] 일 실시예에서, 라이브러리는 잠금(locking) 함수들을 사용하여 스테이지들 사이의 패킷들의 흐름 및 새로운 시간 기간들이 시작되는 시기를 제어한다. 예를 들어, 프리-프로세싱 스레드가 패킷을 프로세싱하는 것을 완료할 때, 이 스레드는 FPGA 실행 스레드에 대한 입력 큐를 잠금할 수 있다. 잠금되어 있는 동안, 이 스레드는 프리-프로세싱 스레드의 출력 버퍼로부터의 패킷들을 프로세싱한 것으로부터의 결과들을 FPGA 실행 스테이지의 입력 큐로 이동시키고, 이 큐를 잠금해제하며, 다음 패킷이 프로세싱될 준비가 되었다는 것을 FPGA 실행 스테이지의 스레드에 시그널링할 수 있다. 스테이지를 잠금하는 것은 스테이지들 사이에서 패킷들을 넘겨줄 때 데이터가 손상될 수 있는 가능성을 경감시킨다.

[0060] 일 실시예에서, 프리-프로세싱 스테이지는 신경 네트워크 애플리케이션으로부터 수신되는 데이터를 신경 네트워크 가속기를 실행하는 하드웨어에 의해 프로세싱될 수 있는 포맷으로 변경한다. 일 예에서, 신경 네트워크 애플리케이션은 부동 소수점 포맷(예를 들어, 32-비트 부동 소수점)의 데이터를 프로세싱할 수 있지만 FPGA 내의 하드웨어 로직은 고정 소수점 값들(예를 들어, 16-비트 또는 8-비트 고정 소수점)에 대해 연산을 한다. 프리-

프로세싱 스테이지는 부동 소수점 값들을 신경 네트워크 가속기에 의해 프로세싱될 수 있는 고정 소수점 값들로 변환한다.

[0061] 더욱이, 프리-프로세싱 스테이지는 신경 네트워크 애플리케이션으로부터 수신되는 데이터를 행렬 포맷(예를 들어, 32x64 데이터 블록)으로 변환할 수 있다. 데이터의 행렬은 FPGA 실행 스테이지 동안 신경 네트워크 가속기의 연산들을 병렬로 수행하는 데 도움이 될 수 있다. 일단 프리-프로세싱 스테이지가 완료되면, 패킷 내의 데이터는 FPGA 내의 하드웨어 로직에 의해 더 쉽게 프로세싱될 수 있는 포맷으로 배열된다.

[0062] FPGA 실행 스테이지 동안, 대응하는 스레드는 패킷(또는 그 안의 데이터의 일 부분)을 FPGA에게 전송한다. 예를 들어, 실행 스테이지에 대한 스레드는 대응하는 데이터 및 가중치들이 메모리에서 어디에 저장되어 있는지를 나타내는 패킷들의 필드들(예를 들어, 메모리 어드레스들)을 전송할 수 있다. 일 실시예에서, FPGA 실행 스테이지에 대한 스레드는 (메모리 맵을 사용하여) FPGA 내의 적절한 메모리에 패킷들의 일 부분을 기입하고, 신경 네트워크 가속기를 실행하는 커널을 모니터링하며, FPGA 내의 메모리로부터 프로세싱된 결과들을 검색하기 위해 판독을 수행한다. 일 실시예에서, 스레드는 데이터를 FPGA로 이동시키고, 커널 또는 커널들에 실행되도록 지시하며, 커널에 대응하는 버퍼들로부터 결과들을 판독하기 위해 인큐잉(enqueue) 커맨드들을 사용한다. 이러한 방식으로, (호스트 상에서 동작하는) FPGA 실행 스테이지의 스레드는 신경 네트워크 가속기를 제어 및 모니터링하기 위해 커맨드들을 FPGA에게 전송할 수 있다. 일단 신경 네트워크 가속기가 완료되고 결과들이 검색되면, FPGA 실행 스테이지의 스레드는 업데이트된 패킷을 포스트-프로세싱 스테이지에게 전달한다.

[0063] 일 실시예에서, 포스트-프로세싱 스테이지는 패킷 내의 데이터를 FPGA에 의해 사용되는 데이터 포맷으로부터 신경 네트워크 애플리케이션에 의해 사용되는 데이터 포맷으로 변환한다. 예를 들어, 포스트-프로세싱 스테이지의 스레드는 프리-프로세싱 스테이지에 대한 역변환(reverse conversion)을 수행한다. 예를 들어, FPGA로부터의 결과들은 고정 소수점 값들로부터 부동 소수점 값들로 변환될 수 있다. 더욱이, 스레드는 더 이상 데이터를 행렬들로서 저장하지 않을 수 있다.

[0064] 포스트-프로세싱 스테이지를 완료한 후에, 라이브러리는 관련 데이터를 호스트 메모리에 저장하고 태스크가 완료되었음을 신경 네트워크 애플리케이션에 통보한다. 일 실시예에서, 라이브러리는 패킷의 일 부분만을 메모리에 저장한다. 예를 들어, 패킷 내의 필드들은 파이프라인 스테이지들에 의해서만 사용될 수 있고, 따라서 패킷이 파이프라인(500)을 통과했을 때 폐기된다.

[0065] 도 5는 3개의 스테이지를 포함하는 파이프라인(500)을 예시하지만, 라이브러리는 임의의 개수의 스테이지들을 포함하는 파이프라인을 구축할 수 있다. 예를 들어, 일부 신경 네트워크 애플리케이션들의 경우, 라이브러리는 FPGA 실행 스테이지 및 포스트-프로세싱 스테이지만을 사용할 수 있다. 다른 애플리케이션들의 경우, 라이브러리는 패킷이 동일한 FPGA 내의 또는 상이한 FPGA들 내의 상이한 신경 네트워크 가속기들 사이에서 전달되는 3개 초과의 스테이지를 갖는 파이프라인을 포함할 수 있다.

[0066] 방법(400)으로 돌아가서, 블록(420)에서, 라이브러리는 제1 데이터 패킷을 프로세싱하는 동안 스레드들을 사용하여 다른 패킷들을 프로세싱한다. 즉, 도 5에 도시된 파이프라인(500)은 다수의 스레드들을 병렬로 프로세싱할 수 있다. 따라서, 프리-프로세싱 스테이지가 제1 패킷을 프로세싱하는 동안, FPGA 실행 스테이지 및 포스트-프로세싱 스테이지가 동시에 다른 패킷들을 프로세싱한다(신경 네트워크 어플리케이션이 다수의 태스크들을 제출했다고 가정함). 그 결과, 파이프라인은 다운시간을 감소시킬 수 있고 신경 네트워크 가속기를 실행하는 커널의 이용률을 개선시킬 수 있다. 예를 들어, 일단 신경 네트워크 가속기가 현재 패킷들을 프로세싱하는 것을 완료하면, 프리-프로세싱 스테이지에 의해 프로세싱된 패킷은, 신경 네트워크 가속기가 즉각 그 패킷을 프로세싱하기 시작할 수 있도록, FPGA에게 전송될 수 있다.

[0067] 블록(425)에서, 라이브러리는 신경 네트워크 가속기를 사용하여 제1 패킷을 프로세싱한 결과들을 신경 네트워크 애플리케이션에게 반환한다. 즉, 일단 제1 패킷이 파이프라인(예를 들어, 프리-프로세싱 스테이지, FPGA 실행 스테이지, 및 포스트-프로세싱 스테이지)을 통과했으면, 라이브러리는 결과들을 신경 네트워크 애플리케이션을 위한 할당된 메모리에 저장하고 태스크가 완료되었음을 애플리케이션에 지시할 수 있다. 스테이지들이 파이프라이닝되어 있기 때문에, 라이브러리는 다수의 태스크들(또는 패킷들)을 병렬로 프로세싱할 수 있다. 따라서, 각각의 시간 기간의 끝에서, 라이브러리는 태스크를 완료한 결과들을 신경 네트워크 애플리케이션에게 전송할 수 있다. 스테이지들이 파이프라이닝되지 않은 경우, 라이브러리는 신경 네트워크 애플리케이션에 의해 제출된 새로운 태스크를 시작하기 전에 태스크가 완료되기를 기다려야 한다. 그 경우에, 라이브러리가 프리-프로세싱 동작 및 포스트-프로세싱 동작을 수행할 때 신경 네트워크 가속기는 유휴 상태이다.

[0068] 도 6은 예에 따른, 신경 네트워크 애플리케이션에 의해 제출된 태스크들에 대한 파이프라인의 실행을 조정하기 위한 방법(600)의 플로차트이다. 블록(605)에서, 라이브러리는 신경 네트워크 가속기의 실행에 관한 메트릭을 획득한다. 일 실시예에서, 라이브러리는 라이브러리가 FPGA 내의 하드웨어 요소들을 모니터링할 수 있게 해주는 FPGA 드라이버와 통신한다. 예를 들어, 드라이버는 신경 네트워크 가속기들을 실행하는 커널들 또는 계산 노드들이 실행되기 시작하는 시기 및 작업이 완료되기까지 얼마나 걸리는지와 같은 메트릭들을 라이브러리에게 보고할 수 있다. 더욱이, 드라이버는 메모리에 대한 판독들 및 기입들과 그 판독들 및 기입들이 얼마나 걸리는지(판독 또는 기입되는 데이터의 양에 따라 달라질 수 있음)에 관해 보고할 수 있다.

[0069] 일 실시예에서, 라이브러리는 이러한 데이터를 드라이버에 요청할 수 있다. 대안적으로, 드라이버는 FGPA에 대한 하드웨어 인터페이스를 자동으로 모니터링하고 이러한 메트릭들을 라이브러리에게 보고할 수 있다.

[0070] 블록(610)에서, 라이브러리는 메트릭들을 실시간 그래픽으로 디스플레이한다. 일 실시예에서, 라이브러리는 FPGA 내의 신경 네트워크 가속기들을 실행하는 하나 이상의 커널의 이용률을 나타내는 차트를 출력한다. 이 차트는 신경 네트워크의 조작자가 커널들의 이용률을 시각화하고 라이브러리에 의해 구축된 파이프라인을 개선시킬 방식들을 식별할 수 있게 해준다.

[0071] 도 7은 예에 따른, 신경 네트워크 애플리케이션에 의해 제출된 태스크들을 파이프라이닝하는 것에 대응하는 타이밍 차트(700)이다. 차트(700)는 커널 A를 사용하여 패킷들을 프로세싱하기 위한 판독들(705), DDR 기입들(710), 및 DDR 기입들(715) 및 커널 B를 사용하여 패킷들을 프로세싱하기 위한 판독들(720), DDR 기입들(725), 및 DDR 기입들(730)을 포함한다. 이 실시예에서, FPGA는 2개의 신경 네트워크 가속기를, 병렬로, 실행할 수 있는 2개의 커널을 포함한다. 달리 말하면, 파이프라인의 FPGA 실행 스테이지 동안, 라이브러리는 커널 A와 커널 B에 의해 병렬로 수행될 태스크들을 제출할 수 있다.

[0072] 더욱이, 이 실시예에서, 라이브러리는 태스크들의 배치(batch)를 커널들에 제출한다. 여기서, 라이브러리는 8개의 태스크 또는 패킷을 배치로서 커널들 각각에 제출한다. 차례로, 커널들은 8개의 태스크를 순차적으로 프로세싱한다. 예를 들어, DDR 기입(710)은 라이브러리가 N개의 태스크들에 대한 데이터를 커널 A 및 대응하는 신경 네트워크 가속기를 위해 할당된 메모리에 기입하는 것을 표현하는 N개의 도트들을 포함할 수 있다. 유사하게, DDR 기입들(725)은 N개의 태스크들의 상이한 배치가 커널 B를 위해 할당된 메모리에 기입되는 것을 표현하는 동일한 N개의 도트들을 또한 포함한다. DDR 기입들(715)은 라이브러리에 의해 생성되는 패킷으로부터의 정보가 FPGA에게 전송되는 시간들을 나타낸다.

[0073] 커널 A에 대한 원들 및 커널 B에 대한 X들은 이러한 커널들이 태스크를 프로세싱하기 시작하는 시기를 나타내는 반면, 라인들은 이러한 태스크들에 걸리는 시간 길이를 나타낸다. 도시된 바와 같이, 커널들은 거의 지속적으로 태스크를 실행하고 있다. 즉, 커널들은 높은 이용률(rate of utilization)을 갖는다. 이것은 커널들이 태스크를 완료하는 시기와 새로운 태스크를 시작하는 시기 사이에 공간(또는 갭들)이, 있더라도, 거의 없다는 사실에 의해 시각적으로 보여지고 있다.

[0074] 판독들(705 및 720)은 라이브러리가 커널들이 태스크들을 프로세싱한 것으로부터의 결과들을 검색하는 시기를 예시한다. 판독들(705)은 커널 A에 의해 제공되는 판독 결과들을 나타내는 N개의 정사각형들을 포함하고, 판독들(720)은 커널 B에 의해 제공되는 판독 결과들을 나타내는 N개의 플러스 부호들을 포함한다. 이러한 방식으로, 차트(700)는 커널 A 및 커널 B의 이용률은 물론 신경 네트워크의 조작자가 커널들 상에서 실행되는 신경 네트워크 가속기들의 이용률 또는 효율을 결정하는 데 도움을 줄 수 있는 메모리에 대한 판독들 및 기입들을 시각화한다.

[0075] 일 실시예에서, 라이브러리는 차트(700)를 실시간으로 업데이트한다. 즉, 커널들이 실행되고 판독들 및 기입들이 수행될 때 라이브러리는 (왼쪽에서 시작하여 오른쪽으로) 차트를 생성할 수 있다. 따라서, 조작자는 차트(700)가 생성되는 것을 볼 수 있다. 일 실시예에서, 조작자는 차트를 줌인하거나 과거의 결과들을 보기 위해 차트를 시간상 뒤로 이동시킬 수 있다.

[0076] 비록 타이밍 차트가 도시되어 있지만, FPGA 내의 커널들 및 DDR 메모리에 대한 메트릭들이 다른 방식들로 디스플레이될 수 있다. 타이밍 차트 대신에, 라이브러리는 유휴 시간과 실행 시간의 비를 나타내는 이용률 퍼센티지를 출력할 수 있다. 다른 예에서, 라이브러리는 커널들의 평균 이용률을 예시하는 막대 그래프를 출력할 수 있다.

[0077] 방법(600)으로 돌아가서, 블록(615)에서, 라이브러리는 파이프라인 스테이지들의 실행을 변경하는 최적화 파라미터들을 신경 네트워크의 조작자로부터 수신한다. 차트(700)를 볼 때, 조작자는 프리-프로세싱 스테이지가 전

형적으로 FPGA 실행 스테이지 및 포스트-프로세싱 스테이지보다 실행하는 데 더 오래 걸린다는 것을 식별했을 수 있다. 그 결과, 새로운 패킷(또는 패킷들의 배치)이 신경 네트워크 가속기들에 의한 프로세싱을 위해 제출될 수 있도록 프리-프로세싱 스테이지를 기다리는 동안 커널들은 유휴 상태일 수 있다. 이에 응답하여, 조작자는 파이프라인 스테이지들 중 하나 이상의 파이프라인 스테이지의 실행 시간을 변경하는 최적화 파라미터를 제공할 수 있다. 이 예에서, 최적화 파라미터는 프리-프로세싱 스테이지의 지속시간을 감소시킬 수 있다.

[0078] 일 실시예에서, 최적화 파라미터는 파이프라인 스테이지들의 스레드들을 실행하도록 할당된 하드웨어 자원들의 양이다. 예를 들어, 프리-프로세싱 스테이지가 파이프라인 내의 다른 스테이지들보다 실행하는 데 더 많은 시간을 지속적으로 요구하는 경우, 조작자는 호스트 내의 부가의 프로세싱 코어들을 프리-프로세싱 스테이지를 실행하도록 할당할 수 있다. 그렇게 하는 것은 프리-프로세싱 스테이지가, 평균적으로, 지속시간이 다른 스테이지들에 더 가까운 지속시간을 갖도록 그의 지속시간을 감소시킬 수 있다. 그 결과, 파이프라인의 전체적인 실행 시간이 감소할 수 있고 FPGA 내의 커널들의 이용률이 증가할 수 있다.

[0079] 다른 예에서, 최적화 파라미터는 파이프라인 스테이지들을 실행하는 스레드들에 더 많은 가상 메모리를 할당하는 것 또는 스레드들이 더 빠른 메모리 요소들 또는 통신 버스들을 사용할 수 있게 해주는 것을 포함할 수 있다. 예를 들어, 라이브러리와 드라이버가 더 빠른 호스트-대-FPGA 통신 스킴(예를 들어, PCIe)을 사용할 수 있게 해주는 것에 의해 FPGA 실행 스테이지의 지속시간이 증가될 수 있다.

[0080] 블록(620)에서, 라이브러리는 수신된 최적화 파라미터에 따라 신경 네트워크 가속기의 이용률을 증가시키기 위해 파이프라인의 스레드들을 실행하는 데 사용되는 하드웨어 자원들을 조정한다. 위에서 언급된 바와 같이, 라이브러리는 파이프라인 스테이지들 중 하나의 파이프라인 스테이지의 스레드를 실행하도록 할당된 프로세싱 코어들의 개수를 증가시킬 수 있거나, 스레드들이 더 빠른 메모리 요소들을 사용하도록 허용할 수 있거나, 또는 호스트와 FPGA 사이의 더 빠른 통신 경로를 가능하게 해줄 수 있다.

[0081] 일 실시예에서, 하드웨어 자원들을 조정하는 것은 파이프라인 내의 스테이지들의 지속시간들을 보다 가깝게 정렬시킨다. 달리 말하면, 라이브러리는 스테이지들이 더 동일하도록 - 이는 FPGA가 더 효율적으로 이용된다는 것을 의미함 - 스테이지들의 지속시간들을 조정할 수 있다. 그렇게 하는 것은 스테이지들 중 일부의 지속시간을 증가시킬 수 있다. 예를 들어, 포스트-프로세싱 스테이지가 FPGA 실행 스테이지보다 짧은 지속시간을 갖지만 프리-프로세싱 스테이지가 FPGA 실행 스테이지보다 긴 지속시간을 갖는 경우, 조작자는 포스트-프로세싱 스테이지를 실행하도록 이전에 할당된 프로세싱 코어들을 프리-프로세싱 스테이지에 할당할 수 있다. 그 결과, 포스트-프로세싱 스테이지의 지속시간은 증가하고 프리-프로세싱 스테이지의 지속시간은 감소한다. 그러나 이러한 지속시간들이 FPGA 실행 스테이지의 지속시간과 동일하면, 파이프라인의 전체적인 실행 시간 및 신경 네트워크 가속기들을 실행하는 커널들의 이용률이 증가할 수 있다.

[0082] 일 실시예에서, 라이브러리는 태스크들을 외부 신경 네트워크 가속기에 제출하는 것 또는 호스트에 제출하는 것 사이를 전환하기 위한 디버깅 함수를 포함한다. 예를 들어, 디버깅할 때, 태스크들을 FPGA에 제출하는 대신에, 라이브러리는 호스트 상의 프로세서들을 사용하여 태스크를 실행할 수 있다. 이것은 더 많은 시간이 걸릴 수 있지만, 그렇게 하는 것은 신경 네트워크 또는 신경 네트워크 가속기의 설계에 의해 문제가 야기되는지를 결정할 수 있다. 달리 말하면, 태스크를 호스트에 제출함으로써, 조작자는 FPGA 상에서 태스크를 실행하는 것에 의해 임의의 에러들이 야기되는지를 결정할 수 있다. 이러한 방식으로, 호스트 프로세서들은 FGPA 특징들을 디버깅하기 위한 베이스라인(baseline)으로서 역할할 수 있다. 일 실시예에서, 라이브러리는 동일한 태스크를 FPGA 상의 신경 네트워크 가속기와 호스트 상의 프로세서들 둘 다에 제출하고 결과들을 비교한다.

[0083] 도 8은 예에 따른 신경 네트워크들을 구현하기 위한 시스템(800)을 묘사하는 블록 다이어그램이다. 시스템(800)은 컴퓨터 시스템(802) 및 하나 이상의 컴퓨터 시스템(808)을 포함한다. 컴퓨터 시스템(802)은 하나 이상의 설계 툴(804)을 제공하는 소프트웨어를 실행하도록 구성된 종래의 컴퓨팅 컴포넌트들을 포함한다. 각각의 컴퓨터 시스템(808)은 (위에서 설명된 예들 중 임의의 예에서와 같은) 하나 이상의 신경 네트워크(810)를 실행한다. 신경 네트워크(들)(810)는 (위에서 설명된 예들 중 임의의 예에서와 같은) 애플리케이션들(812), (위에서 설명된 예들 중 임의의 예에서와 같은) 가속 라이브러리들(814), 및 (위에서 설명된 예들 중 임의의 예에서와 같은) 하나 이상의 하드웨어 가속기(816)를 사용하여 구현된다.

[0084] 예에서, 하드웨어 가속기(들)(816)는, FPGA들과 같은, 프로그래밍가능 IC들을 포함한다. 가속 라이브러리들(814)은 하드웨어 가속기(들)(816)와 인터페이싱하기 위해 API들을 제공한다. 가속 라이브러리들(814)은, 신경 네트워크 계층들 및 다른 유형들의 신경 네트워크 구조들의 미리 정의되고 최적화된 구현들을 포함하여, 신경 네트워크 함수들을 제공하는 라이브러리들을 또한 포함할 수 있다. 따라서, 신경 네트워크(들)(810)는 하드웨

어 가속기(들)(816)에 구현되는 하드웨어 부분들은 물론, 가속 라이브러리들(814)에 구현되는 소프트웨어 부분들 둘 다를 포함할 수 있다. 애플리케이션들(812)은 신경 네트워크(들)(816)를 구현하도록 하드웨어 가속기(들)(816)를 프로그래밍 및 제어하기 위해 가속 라이브러리들(814)의 API들을 호출한다.

[0085] 설계자는 신경 네트워크(들)(810)를 정의하기 위해 설계 툴(들)(804)과 상호작용한다. 설계 툴(들)(804)은 하드웨어 가속기(들)(816)를 프로그래밍하기 위한 파일들(예를 들어, FPGA들을 위한 구성 비트스트림들), 가속 라이브러리들(814)을 제공하는 파일들, 및 애플리케이션들(812)을 제공하는 파일들을 생성할 수 있다. 설계자는 레지스터 전송 언어(register transfer language; RTL)를 사용하여 또는, C, C++, OpenCL 등과 같은, 프로그래밍 언어, 또는 RTL과 프로그래밍가능 언어(들)의 조합을 사용하여 신경 네트워크(들)(810)의 하드웨어 부분들을 정의할 수 있다. 사용자는, C, C++, OpenCL 등과 같은, 프로그래밍 언어를 사용하여 신경 네트워크(들)(810)의 소프트웨어 부분들을 정의할 수 있다. 설계 툴(들)(804)은 소프트웨어 정의(software-defined) 신경 네트워크들을 컴파일하여 하드웨어 가속기(들)(816)를 프로그래밍하기 위한 파일들 및 가속 라이브러리들(814)을 위한 라이브러리 파일들을 생성한다. 설계자는 신경 네트워크(들)(810)의 하드웨어 부분들 및 소프트웨어 부분들을 개발하는 것을 돕기 위해 클래스 라이브러리들, 템플릿 라이브러리들 등을 제공하는 라이브러리들(106)을 사용할 수 있다.

[0086] 사용자는 프로그래밍 언어(예를 들어, C, C++, Python 등)를 사용하여 애플리케이션들(812)을 정의할 수 있다. 사용자는, Caffe, TensorFlow, MXNet 등과 같은, 신경 네트워크 프레임워크들 및 라이브러리들을 사용할 수 있다.

[0087] 도 9는 예에 따른 컴퓨팅 시스템(808)을 묘사하는 블록 다이어그램이다. 컴퓨팅 시스템(808)은 하드웨어(904) 및 하드웨어(904) 상에서 실행되는 소프트웨어(906)를 포함한다. 하드웨어(904)는 프로세싱 시스템(910), 시스템 메모리(916), 저장 디바이스들("스토리지(918)"), 및 하드웨어 가속기(816)를 포함한다. 소프트웨어(906)는 운영 체제(OS)(944), 가속 라이브러리들(814), 및 애플리케이션들(812)을 포함한다.

[0088] 프로세싱 시스템(910)은 마이크로프로세서(912), 지원 회로들(914), 및 주변기기 버스(915)를 포함한다. 마이크로프로세서(912)는, x86 기반 프로세서, ARM® 기반 프로세서 등과 같은, 임의의 유형의 범용 중앙 프로세싱 유닛(CPU)일 수 있다. 마이크로프로세서(912)는 하나 이상의 코어 및 연관된 회로부(예를 들어, 캐시 메모리들, 메모리 관리 유닛들(memory management units; MMU들), 인터럽트 제어기들 등)를 포함할 수 있다. 마이크로프로세서(912)는 본 명세서에서 설명되는 하나 이상의 동작을 수행하고 시스템 메모리(916) 및/또는 스토리지(918)에 저장될 수 있는 프로그램 코드를 실행하도록 구성된다. 지원 회로들(914)은 마이크로프로세서(912), 시스템 메모리(916), 스토리지(918), 하드웨어 가속기(816), 또는 임의의 다른 주변 디바이스 사이의 데이터 흐름을 관리하기 위해 마이크로프로세서(912)와 협력하는 다양한 디바이스들을 포함한다. 예를 들어, 지원 회로들(914)은 칩세트(예를 들어, 노스 브리지, 사우스 브리지, 플랫폼 호스트 제어기 등), 전압 레귤레이터들, 펌웨어(예를 들어, BIOS) 등을 포함할 수 있다. 지원 회로들(914)은, 하드웨어 가속기(816)와 같은, 다양한 주변기기들에 접속되는 주변기기 버스(915)와 마이크로프로세서(912) 사이의 데이터 흐름을 관리한다. 일부 예들에서, 마이크로프로세서(912)는 칩세트(예를 들어, 노스 브리지, 사우스 브리지 등)의 기능성의 전부 또는 상당 부분을 흡수하는 시스템-인-패키지(System-in-Package; SiP), 시스템-온-칩(System-on-Chip; SoC) 등일 수 있다. 주변기기 버스는, PCIe(Peripheral Component Interconnect Express)와 같은, 확장 버스 표준을 구현할 수 있다. 이 예에서, 프로세싱 시스템(910)은 하드웨어 가속기(816)와 별개로 도시되어 있다. 아래에서 추가로 논의되는 다른 예들에서, 프로세싱 시스템(910)과 하드웨어 가속기(816)는 시스템-온-칩(SoC)을 사용하여 동일한 IC 상에 구현될 수 있다.

[0089] 시스템 메모리(916)는, 실행가능 명령어들 및 데이터와 같은, 정보가 저장되고 검색될 수 있게 해주는 디바이스이다. 시스템 메모리(916)는, 예를 들어, 더블 데이터 레이트(DDR) 동적 RAM(DRAM)과 같은, 하나 이상의 랜덤 액세스 메모리(RAM) 모듈을 포함할 수 있다. 저장 디바이스(918)는 로컬 저장 디바이스들(예를 들어, 하나 이상의 하드 디스크, 플래시 메모리 모듈, 솔리드 스테이트 디스크, 및 광학 디스크) 및/또는 컴퓨팅 시스템(808)이 하나 이상의 네트워크 데이터 저장 시스템과 통신할 수 있게 해주는 스토리지 인터페이스를 포함한다. 하드웨어(904)는, 그래픽 카드들, 범용 직렬 버스(USB) 인터페이스들 등과 같은, 컴퓨팅 시스템의 다양한 다른 종래의 디바이스들 및 주변기기들을 포함할 수 있다.

[0090] 하드웨어 가속기(816)는 프로그래밍가능 IC(928), 비휘발성 메모리(924), 및 RAM(926)을 포함한다. 프로그래밍가능 IC(928)는 FPGA 등 또는 FPGA 등을 갖는 SoC일 수 있다. NVM(924)은, 플래시 메모리 등과 같은, 임의의 유형의 비휘발성 메모리를 포함할 수 있다. RAM(926)은 DDR DRAM 등을 포함할 수 있다. 프로그래밍가능

IC(928)는 NVM(924) 및 RAM(926)에 커플링된다. 프로그래밍가능 IC(928)는 또한 프로세싱 시스템(910)의 주변기기 버스(915)에 커플링된다.

[0091] OS(914)는, Linux®, Microsoft Windows®, Mac OS® 등과 같은, 본 기술분야에서 알려진 임의의 상용 운영 체제일 수 있다. 가속 라이브러리들(814)은 하드웨어 가속기(816)의 커맨드 및 제어를 위한 API들을 제공하는 드라이버들 및 라이브러리들을 포함한다. 애플리케이션들(812)은 신경 네트워크(들)를 구현하기 위해 가속 라이브러리들(814)의 API들을 호출하는 마이크로프로세서(912) 상에서 실행되는 소프트웨어를 포함한다.

[0092] 동작 중에, 프로그래밍가능 IC(928)는 가속 회로(930)로 구성된다. 일 예에서, 가속 회로(930)는 도 2에서의 신경 네트워크 가속기(165)이지만, 본 명세서에서의 실시예들은 이에 제한되지 않으며 다른 유형들의 신경 네트워크 가속기들 또는 다른 유형들의 하드웨어 가속기들일 수 있다. 가속 회로(930)는 일반적으로 베이스 플랫폼(base platform)(930A) 및 커널(930B)을 포함한다. 예를 들어, 가속 회로(930)는 정적 영역(934) 및 프로그래밍가능 영역(936)을 사용하여 구현될 수 있다. 정적 영역(934)은 주변기기 버스(915), NVM(924), 및 RAM(926)에 대한 인터페이스를 제공하기 위한 지원 회로들(940)을 포함한다. 프로그래밍가능 영역(936)은 하나 이상의 커널 회로("커널(들)(938)")를 포함할 수 있다. 베이스 플랫폼(930A)은 정적 영역(934)을 사용하여 구현되고, 커널(930B)은 프로그래밍가능 영역(936)을 사용하여 구현된다. 다른 예에서, 베이스 플랫폼(930A)은 또한 프로그래밍가능 영역(936)의 일 부분을 사용하여 구현될 수 있다. 따라서, 일부 예들에서, 프로그래밍가능 영역(936)은 일부 인터페이스 회로들을 또한 포함한다. 일부 예들에서, 가속 회로(930)는 하나 초과의 프로그래밍가능 영역(936)을 포함할 수 있으며, 이들 각각은 커널(들)(938)로 개별적으로 구성될 수 있다.

[0093] 정적 영역(934)은 그의 회로부가 프로그래밍가능 영역(936)의 재구성들에 걸쳐 일정하게 유지된다는 점에서 "정적"이다. 예에서, 지원 회로들(940)은 PCIe 엔드포인트 회로들, 직접 메모리 액세스(DMA) 제어기, 인터커넥트들(interconnects), 메모리 제어기, 메모리 인터페이스 회로(예를 들어, DDR 인터페이스), (부분 재구성을 지원하기 위한) 디커플러(decoupler) 회로들, 플래시 프로그래머, 디버그 회로들 등을 포함한다. 일부 예들에서, 프로그래밍가능 영역(936)은 지원 회로들(940) 중 어느 것도 포함하지 않는다. 다른 예들에서, 일부 지원 회로들은 프로그래밍가능 영역(936)에서 구현된다. 그러한 경우에, 프로그래밍가능 영역(936)은 "확장 프로그래밍가능 영역"이라고 지칭될 수 있다. 어느 경우든지, 하나의 예에서, PCIe 회로들 및 DMA 회로들과 같은, 일부 지원 회로들(940)은 항상 정적 영역(934)에 존재한다.

[0094] 도 10은 예에 따른 가속 회로(930)를 묘사하는 블록 다이어그램이다. 가속 회로(930)는 지원 회로들(940) 및 커널(938)을 포함한다. 이 예에서, 지원 회로들(940)은 PCIe 엔드포인트 회로("PCIe 엔드포인트(1002)"), PCIe DMA 제어기(1004), 인터커넥트 회로들("인터커넥트(1006)"), 메모리 제어기들(1010), 및 메모리 인터페이스들(1012)을 포함한다. 지원 회로들(940)은, 명확성을 위해 생략된, 다른 회로들(예를 들어, 디커플러 회로들, 디버그 회로들 등)을 포함할 수 있다. PCIe 엔드포인트(1002)는 주변기기 버스(915)에 대한 물리적 인터페이스를 제공한다. PCIe DMA 제어기(1004)는 RAM(926) 및 커널(938)에 대한 DMA 동작들을 용이하게 한다. 인터커넥트(1006)는 PCIe DMA 제어기(1004)를 메모리 제어기들(1010) 및 커널(938)에 커플링시킨다. 메모리 제어기들(1010)은 메모리 인터페이스들(1012)에 커플링된다. 메모리 인터페이스들(1012)은 RAM(926)에 커플링된다.

[0095] 동작 중에, 가속 라이브러리들(946)은 PCIe DMA 제어기(1004)를 통해 직접적으로 RAM(926)에 액세스할 수 있다. 가속 라이브러리들(946)은 또한 PCIe DMA 제어기(1004)를 통해 커널(938)에 액세스할 수 있다. 커널(938)은 메모리 제어기들(1010)을 통해 RAM(926)에 액세스할 수 있다. 시스템 메모리(916)와 RAM(926) 사이의 DMA 동작들을 사용하여 소프트웨어(906)와 커널(938) 사이에서 데이터가 교환될 수 있다.

[0096] 이 예에서, 커널(938)은 인터커넥트(1006)와 통신하기 위해 인터페이스들(1030, 1031, 및 1032)을 사용한다. 특히, 이러한 인터페이스들은 제1 판독 인터페이스(1030), 제2 판독 인터페이스(1031), 및 판독/기입 인터페이스(1032)를 포함할 수 있다. 예를 들어, 판독 인터페이스(1030)는 커널(938)을 제어하기 위한 제어 인터페이스로서 사용될 수 있다. 판독 인터페이스(1031)는 메모리 인터페이스들(1012) 중 제1 인터페이스를 통해 RAM(926)으로부터 판독하는 데 사용될 수 있다. 판독/기입 인터페이스(1032)는 메모리 인터페이스들(1012) 중 제2 인터페이스를 통해 RAM(926)으로부터 판독하고 기입하는 데 사용될 수 있다.

[0097] 커널(938)은 인터커넥트 인터페이스(1004), 제어 로직(1042), 및 프로세싱 회로들(1041)을 포함한다. 프로세싱 회로들(1041)은 IM2COL 회로("IM2COL(1044)"), 판독 제어 회로("판독 제어(1046)"), 멀티플렉서(1056), 선입선출(first-in-first-out) 회로들("FIFO들(1058)"), DSP 어레이(1062), 스케일러 회로("스케일러(1064)"), 맥스 풀링(max pool) 회로("맥스 풀링(1066)"), 멀티플렉서(1068), FIFO들(1054), 기입 제어 회로("기입 제어

(1052)"), 캐시(1048), 판독 제어 회로("판독 제어(1050)"), 및 FIFO들(1060)을 포함한다. 인터커넥트 인터페이스(1040)는 인터페이스들(1030, 1031, 및 1032), 제어 로직(1042), 및 프로세싱 회로들(1041)에 커플링된다. 인터커넥트 인터페이스(1040)는 제어 로직(1042)과 인터페이스(1030) 사이는 물론, 프로세싱 회로들(1041)과 인터페이스들(1031 및 1032) 사이의 통신을 용이하게 하기 위해 스위치들, 클록 변환기들 등을 포함할 수 있다.

[0098] 이 예에서, 인터커넥트 인터페이스(1040)는 IM2COL 회로(1044), 판독 제어 회로(1046), 캐시(1048), 및 기입 제어 회로(1052)의 입력들에 커플링된다. IM2COL 회로(1044) 및 판독 제어 회로(1046)의 출력들은 멀티플렉서(1056)의 입력들에 커플링된다. 멀티플렉서(1056)의 출력은 FIFO들(1056)의 입력에 커플링된다. FIFO들(1056)의 출력은 계산 어레이(1062)의 제1 입력에 커플링된다. 캐시(1048)의 출력은 판독 제어 회로(1050)의 입력에 커플링된다. 판독 제어 회로(1050)의 출력은 FIFO들(1060)의 입력에 커플링된다. FIFO들(1060)의 출력은 계산 어레이(1062)의 제2 입력에 커플링된다. 계산 어레이(1062)의 출력은 스케일러(1064)의 입력에 커플링된다. 스케일러(1064)의 출력은 맥스 풀링 회로(1066)의 입력 및 멀티플렉서(1068)의 입력에 커플링된다. 맥스 풀링 회로(1066)의 출력은 멀티플렉서(1068)의 다른 입력에 커플링된다. 멀티플렉서(1068)의 출력은 FIFO들(1054)의 입력에 커플링된다. FIFO들(1054)의 출력은 기입 제어 회로(1052)에 커플링된다.

[0099] 동작 중에, 계산 어레이(1062)는 신경 네트워크를 구현하기 위해 행렬 곱셈 연산들을 수행한다. 계산 어레이(1062)의 입력들은 FIFO들(1058)로부터 입력 활성화 행렬들을 수신하고 FIFO들(1060)로부터 가중치 행렬들을 수신한다. 입력 활성화 행렬들은 판독 제어 회로(1046)를 사용하여 RAM(926)으로부터 직접적으로 판독될 수 있다. 대안적으로, 입력 활성화들은 RAM(926)으로부터 판독되고 계산 어레이(1062)로의 입력을 위해 IM2COL 회로(1044)에 의해 프로세싱될 수 있다. IM2COL 회로(1044)의 실시예들이 아래에서 설명된다. 가중치 행렬들은 판독 제어 회로(1050)에 의해 RAM(926)으로부터 판독되고 캐시(1048)에 캐싱될 수 있다. 스케일러(1064)는 계산 어레이(1062)의 출력을 스케일링할 수 있다. 맥스 풀링 회로(1066)는 계산 어레이(1062)의 스케일링된 출력에 대해 맥스 풀링 함수를 구현할 수 있다. 일 예에서, 맥스 풀링 회로(966)는 구성가능 로직 블록들(configurable logic blocks; CLB들) 또는 다른 구성가능 로직을 사용하여 구현된다. 맥스 풀링 회로(1066) 또는 스케일러(1064)의 출력 중 어느 하나가 FIFO들(1054)에 저장될 수 있다. 기입 제어 회로(1052)는 FIFO들 내의 데이터를 RAM(926)에 기입한다. 제어 로직(1042)은, IM2COL 회로(1044), 판독 제어 회로(1046), 멀티플렉서들(1056 및 1068), 판독 제어 회로(1050), 스케일러(1064), 맥스 풀링 회로(1066), 및 기입 제어 회로(1052)와 같은, 프로세싱 회로들(1041) 내의 다양한 회로들을 제어한다.

[0100] 도 11은 예에 따른 프로그래밍가능 IC(928)를 묘사하는 블록 다이어그램이다. 프로그래밍가능 IC(228)는 프로그래밍가능 로직(3), 구성 로직(25), 및 구성 메모리(26)를 포함한다. 프로그래밍가능 IC(928)는, NVM(924), RAM(926), 및 다른 회로들(29)과 같은, 외부 회로들에 커플링될 수 있다. 프로그래밍가능 로직(3)은 로직 셀들(30), 지원 회로들(31), 및 프로그래밍가능 인터커넥트(interconnect)(32)를 포함한다. 로직 셀들(30)은 복수의 입력들의 일반적인 로직 함수들을 구현하도록 구성될 수 있는 회로들을 포함한다. 지원 회로들(31)은, 트랜시버들, 입/출력 블록들, 디지털 신호 프로세서들, 메모리들 등과 같은, 전용 회로들을 포함한다. 로직 셀들과 지원 회로들(31)은 프로그래밍가능 인터커넥트(32)를 사용하여 상호접속될 수 있다. 로직 셀들(30)을 프로그래밍하기 위한, 지원 회로들(31)의 파라미터들을 설정하기 위한, 그리고 프로그래밍가능 인터커넥트(32)를 프로그래밍하기 위한 정보는 구성 로직(25)에 의해 구성 메모리(26)에 저장된다. 구성 로직(25)은 비휘발성 메모리(924) 또는 임의의 다른 소스(예컨대, DRAM(28) 또는 다른 회로들(29))로부터 구성 데이터를 획득할 수 있다. 일부 예들에서, 프로그래밍가능 IC(928)는 프로세싱 시스템(2)을 포함한다. 프로세싱 시스템(2)은 마이크로프로세서(들), 메모리, 지원 회로들, IO 회로들 등을 포함할 수 있다. 예를 들어, 프로세싱 시스템(2)은 프로세싱 시스템(910)과 유사한 회로들을 포함할 수 있다. 일부 예들에서, 프로세싱 시스템(2)은 프로세싱 시스템(910) 대신에 사용될 수 있다. 그러한 경우에, 전체 컴퓨팅 시스템(808)은 프로그래밍가능 IC(928)를 사용하여 구현될 수 있으며, 여기서 소프트웨어(906)는 프로세싱 시스템(2) 상에서 실행된다.

[0101] 도 12는 트랜시버들(37), CLB들(33), BRAM들(34), 입/출력 블록들("IOB들")(36), 구성 및 클로킹 로직("구성/클록들")(42), DSP들(35), 특수 입/출력 블록들("I/O")(41)(예를 들어, 구성 포트들 및 클록 포트들), 및 디지털 클록 관리자들, 아날로그-디지털 변환기들, 시스템 모니터링 로직 등과 같은 다른 프로그래밍가능 로직(39)을 포함한 많은 수의 상이한 프로그래밍가능 타일들을 포함하는 프로그래밍가능 IC(928)의 FPGA 구현을 예시한다. FPGA는 PCIe 인터페이스들(40), 아날로그-디지털 변환기들(ADC)(38) 등을 또한 포함할 수 있다.

[0102] 일부 FPGA들에서, 각각의 프로그래밍가능 타일은, 도 12의 상부에 포함된 예들에 의해 도시된 바와 같이, 동일한 타일 내의 프로그래밍가능 로직 요소의 입력 및 출력 단자들(48)에 대한 접속들을 갖는 적어도 하나의 프로그래밍가능 인터커넥트 요소("INT")(43)를 포함할 수 있다. 각각의 프로그래밍가능 인터커넥트 요소(43)는 동

일한 타일 또는 다른 타일(들)에서 인접한 프로그래밍가능 인터커넥트 요소(들)의 인터커넥트 세그먼트들(49)에 대한 접속들을 또한 포함할 수 있다. 각각의 프로그래밍가능 인터커넥트 요소(43)는 로직 블록들(도시되지 않음) 사이의 일반적인 라우팅 자원들의 인터커넥트 세그먼트들(50)에 대한 접속들을 또한 포함할 수 있다. 일반적인 라우팅 자원들은 인터커넥트 세그먼트들(예컨대, 인터커넥트 세그먼트들(50))의 트랙들을 포함하는 로직 블록들(도시되지 않음)과 인터커넥트 세그먼트들을 접속시키기 위한 스위치 블록들(도시되지 않음) 사이의 라우팅 채널들을 포함할 수 있다. 일반적인 라우팅 자원들의 인터커넥트 세그먼트들(예컨대, 인터커넥트 세그먼트들(50))은 하나 이상의 로직 블록에 걸쳐 있을 수 있다. 일반적인 라우팅 자원들과 합쳐진 프로그래밍가능 인터커넥트 요소들(43)은 예시된 FPGA를 위한 프로그래밍가능 인터커넥트 구조("프로그래밍가능 인터커넥트")를 구현한다.

[0103] 예시적인 구현에서, CLB(33)는 사용자 로직 및 단일 프로그래밍가능 인터커넥트 요소("INT")(43)를 구현하도록 프로그래밍될 수 있는 구성가능 로직 요소("CLE")(44)를 포함할 수 있다. BRAM(34)은 하나 이상의 프로그래밍가능 인터커넥트 요소에 부가하여 BRAM 로직 요소("BRL")(45)를 포함할 수 있다. 전형적으로, 타일에 포함되는 인터커넥트 요소들의 개수는 타일의 높이에 의존한다. 묘사된 예에서, BRAM 타일은 5개의 CLB와 동일한 높이를 갖지만, 다른 개수들(예컨대, 4개)이 또한 사용될 수 있다. DSP 타일(35)은 적절한 개수의 프로그래밍가능 인터커넥트 요소들에 부가하여 DSP 로직 요소("DSPL")(46)를 포함할 수 있다. IOB(36)는, 예를 들어, 프로그래밍가능 인터커넥트 요소(43)의 하나의 인스턴스에 부가하여 입/출력 로직 요소("IOL")(47)의 2개의 인스턴스를 포함할 수 있다. 본 기술분야의 통상의 기술자에게 명백할 것인 바와 같이, 예를 들어, I/O 로직 요소(47)에 접속된 실제 I/O 패드들은 전형적으로 입/출력 로직 요소(47)의 영역으로 한정되지 않는다.

[0104] 묘사된 에에서, (도 12에 도시된) 다이의 중심 부근의 수평 영역은 구성, 클록, 및 다른 제어 로직에 사용된다. 이 수평 영역 또는 열로부터 연장되는 수직 열들(51)은 FPGA의 폭에 걸쳐 클록들 및 구성 신호들을 분배하는 데 사용된다.

[0105] 도 12에 예시된 아키텍처를 이용하는 일부 FPGA들은 FPGA의 대부분을 구성하는 규칙적인 열 구조를 방해하는 부가의 로직 블록들을 포함한다. 부가의 로직 블록들은 프로그래밍가능 블록들 및/또는 전용 로직일 수 있다.

[0106] 도 12가 단지 예시적인 FPGA 아키텍처를 예시하는 것으로 의도되어 있음에 유의한다. 예를 들어, 행에 있는 로직 블록들의 개수들, 행들의 상대 폭, 행들의 개수 및 순서, 행들에 포함된 로직 블록들의 유형들, 로직 블록들의 상대 크기들, 및 도 12의 상부에 포함된 인터커넥트/로직 구현들은 순전히 예시적인 것이다. 예를 들어, 실제 FPGA에서는, 사용자 로직의 효율적 구현을 용이하게 하기 위해, CLB들이 나타날 때마다 하나 초과의 인접한 CLB 행이 전형적으로 포함되지만, 인접한 CLB 행들의 개수는 FPGA의 전체적인 크기에 따라 달라진다.

[0107] 이상에서, 본 개시내용에서 제시된 실시예들이 언급되었다. 그렇지만, 본 개시내용의 범위가 설명된 특정 실시예들로 제한되지 않는다. 그 대신에, 상이한 실시예들에 관련되어 있는지 여부와 관계없이, 본 명세서에서 설명된 특징들 및 요소들의 임의의 조합이 고려된 실시예들을 구현 및 실시하기 위해 고려된다. 게다가, 비록 본 명세서에서 개시된 실시예들이 다른 가능한 해결책들에 비해 또는 종래 기술에 비해 장점들을 달성할 수 있지만, 특정의 장점이 주어진 실시예에 의해 달성되는지 여부가 본 개시내용의 범위를 제한하지 않는다. 따라서, 본 명세서에서 설명된 양태들, 특징들, 실시예들 및 장점들은 예시적인 것에 불과하고, 청구항(들)에 명시적으로 언급된 경우를 제외하고는, 첨부된 청구항들의 요소들 또는 제한들인 것으로 간주되지 않는다. 마찬가지로, "본 발명"에 대한 언급이 본 명세서에서 개시된 임의의 발명 주제의 일반화로서 해석되어서는 안되며, 청구항(들)에서 명시적으로 언급된 경우를 제외하고는, 첨부된 청구항들의 요소 또는 제한인 것으로 간주되어서는 안된다.

[0108] 본 명세서에서 설명된 양태들은 전적으로 하드웨어인 실시예(entirely hardware embodiment), 전적으로 소프트웨어인 실시예(entirely software embodiment)(펌웨어, 상주 소프트웨어, 마이크로코드(micro-code) 등을 포함함) 또는 소프트웨어 양태와 하드웨어 양태를 조합한 실시예 - 이들 모두는 일반적으로 본 명세서에서 "모듈" 또는 "시스템"이라고 지칭될 수 있음 - 의 형태를 취할 수 있다.

[0109] 본 발명은 시스템, 방법, 및/또는 컴퓨터 프로그램 제품일 수 있다. 컴퓨터 프로그램 제품은 프로세서로 하여금 본 발명의 양태들을 수행하게 하기 위한 컴퓨터 판독가능 프로그램 명령어들을 갖는 컴퓨터 판독가능 저장 매체(또는 매체들)를 포함할 수 있다.

[0110] 컴퓨터 판독가능 저장 매체는 명령어 실행 디바이스에 의해 사용하기 위한 명령어들을 유지 및 저장할 수 있는 유형적 디바이스일 수 있다. 컴퓨터 판독가능 저장 매체는, 예를 들어, 전자 저장 디바이스, 자기 저장 디바이

스, 광학 저장 디바이스, 전자기 저장 디바이스, 반도체 저장 디바이스, 또는 전술한 것의 임의의 적합한 조합일 수 있지만, 이들로 제한되지 않는다. 컴퓨터 판독가능 저장 매체의 더 구체적인 예들의 비-총망라적(non-exhaustive) 리스트는 다음과 같은 것: 휴대용 컴퓨터 디스켓, 하드 디스크, 랜덤 액세스 메모리(RAM), 판독 전용 메모리(ROM), 소거가능 프로그래밍가능 판독 전용 메모리(EPROM 또는 플래시 메모리), 정적 랜덤 액세스 메모리(SRAM), 휴대용 콤팩트 디스크 판독 전용 메모리(CD-ROM), 디지털 다기능 디스크(DVD), 메모리 스틱, 플로피 디스크, 명령어들이 기록되어 있는 펀치 카드들 또는 그루브 내의 융기된 구조들과 같은 기계적으로 인코딩된 디바이스, 및 전술한 것의 임의의 적합한 조합을 포함한다. 컴퓨터 판독가능 저장 매체는, 본 명세서에서 사용되는 바와 같이, 전파들(radio waves) 또는 다른 자유 전파(freely propagating) 전자기파들, 도파관 또는 다른 전송 매체들을 통해 전파되는 전자기파들(예를 들어, 광섬유 케이블을 통과하는 광 펄스들), 또는 전선(wire)을 통해 전송되는 전기 신호들과 같은, 일시적 신호들 그 자체인 것으로서 해석되어서는 안된다.

[0111] 본 명세서에서 설명된 컴퓨터 판독가능 프로그램 명령어들은 컴퓨터 판독가능 저장 매체로부터 각자의 컴퓨팅/프로세싱 디바이스들로 또는 네트워크, 예를 들어, 인터넷, 로컬 영역 네트워크, 광역 네트워크, 및/또는 무선 네트워크를 통해 외부 컴퓨터 또는 외부 저장 디바이스로 다운로드될 수 있다. 네트워크는 구리 전송 케이블들, 광학 전송 섬유들, 무선 전송, 라우터들, 방화벽들, 스위치들, 게이트웨이 컴퓨터들 및/또는 에지 서버들을 포함할 수 있다. 각각의 컴퓨팅/프로세싱 디바이스 내의 네트워크 어댑터 카드 또는 네트워크 인터페이스는 네트워크로부터 컴퓨터 판독가능 프로그램 명령어들을 수신하고 각자의 컴퓨팅/프로세싱 디바이스 내의 컴퓨터 판독가능 저장 매체에 저장하기 위해 컴퓨터 판독가능 프로그램 명령어들을 포워딩한다.

[0112] 본 발명의 동작들을 수행하기 위한 컴퓨터 판독가능 프로그램 명령어들은 어셈블러 명령어들, ISA(instruction-set-architecture) 명령어들, 머신 명령어들, 머신 의존적 명령어들, 마이크로코드, 펌웨어 명령어들, 상태 설정 데이터, 또는 Smalltalk, C++ 등과 같은 객체 지향 프로그래밍 언어, 및 "C" 프로그래밍 언어 또는 유사한 프로그래밍 언어들과 같은 종래의 절차적 프로그래밍 언어들을 포함한, 하나 이상의 프로그래밍 언어의 임의의 조합으로 작성된 소스 코드 또는 오브젝트 코드 중 어느 하나일 수 있다. 컴퓨터 판독가능 프로그램 명령어들은 전체적으로 사용자의 컴퓨터 상에서, 부분적으로 사용자의 컴퓨터 상에서, 독립형 소프트웨어 패키지로서, 부분적으로 사용자의 컴퓨터 상에서 그리고 부분적으로 원격 컴퓨터 상에서 또는 전체적으로 원격 컴퓨터 또는 서버 상에서 실행될 수 있다. 후자의 시나리오에서, 원격 컴퓨터는 로컬 영역 네트워크(LAN) 또는 광역 네트워크(WAN)를 포함한, 임의의 유형의 네트워크를 통해 사용자의 컴퓨터에 접속될 수 있거나, 또는 (예를 들어, 인터넷 서비스 제공업체를 사용하여 인터넷을 통해) 외부 컴퓨터에 대한 접속이 이루어질 수 있다. 일부 실시예들에서, 예를 들어, 프로그래밍가능 로직 회로부, 필드 프로그래밍가능 게이트 어레이들(FPGA), 또는 프로그래밍가능 로직 어레이들(PLA)을 포함한 전자 회로부는, 본 발명의 양태들을 수행하기 위해, 컴퓨터 판독가능 프로그램 명령어들의 상태 정보를 이용하여 전자 회로부를 개인화함으로써 컴퓨터 판독가능 프로그램 명령어들을 실행할 수 있다.

[0113] 본 발명의 양태들이 본 발명의 실시예들에 따른 방법들, 장치들(시스템들), 및 컴퓨터 프로그램 제품들의 플로차트 예시들 및/또는 블록 다이어그램들을 참조하여 본 명세서에서 설명되었다. 플로차트 예시들 및/또는 블록 다이어그램들의 각각의 블록, 및 플로차트 예시들 및/또는 블록 다이어그램들에서의 블록들의 조합들이 컴퓨터 판독가능 프로그램 명령어들에 의해 구현될 수 있음이 이해될 것이다.

[0114] 이러한 컴퓨터 판독가능 프로그램 명령어들은, 컴퓨터 또는 다른 프로그래밍가능 데이터 프로세싱 장치의 프로세서를 통해 실행되는 명령어들이 플로차트 및/또는 블록 다이어그램 블록 또는 블록들에 명시된 기능들/동작들을 구현하기 위한 수단들을 생성하도록, 머신을 생성하기 위해 범용 컴퓨터, 특수 목적 컴퓨터, 또는 다른 프로그래밍가능 데이터 프로세싱 장치의 프로세서에 제공될 수 있다. 명령어들을 저장하고 있는 컴퓨터 판독가능 저장 매체가 플로차트 및/또는 블록 다이어그램 블록 또는 블록들에 명시된 기능/동작의 양태들을 구현하는 명령어들을 포함하는 제조 물품을 포함하도록, 특정의 방식으로 기능하라고 컴퓨터, 프로그래밍가능 데이터 프로세싱 장치, 및/또는 다른 디바이스들에 지시할 수 있는 이러한 컴퓨터 판독가능 프로그램 명령어들이 또한 컴퓨터 판독가능 저장 매체에 저장될 수 있다.

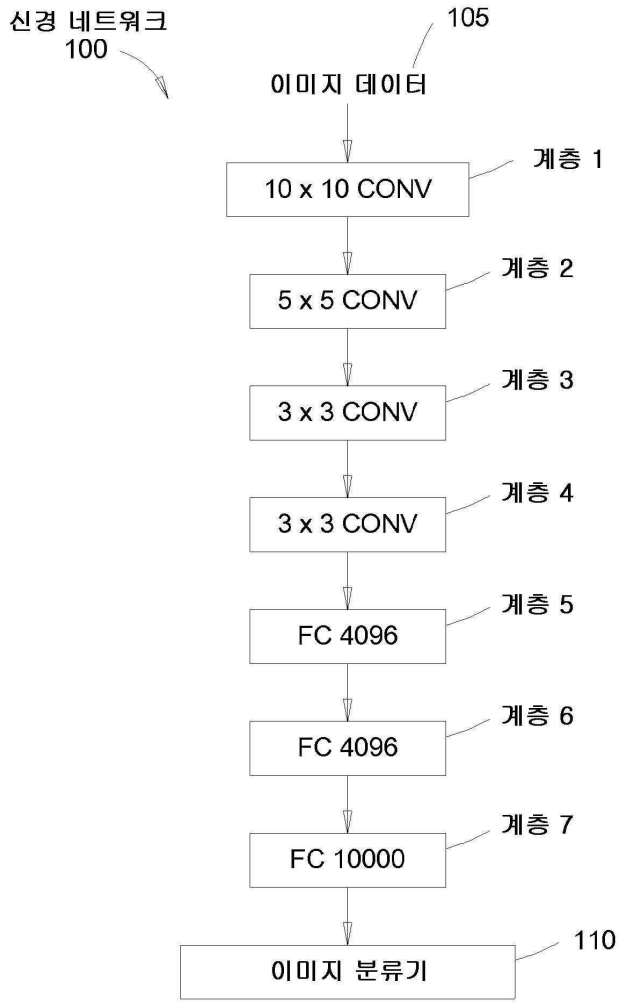
[0115] 컴퓨터, 다른 프로그래밍가능 장치, 또는 다른 디바이스 상에서 실행되는 명령어들이 플로차트 및/또는 블록 다이어그램 블록 또는 블록들에 명시된 기능들/동작들을 구현하도록, 컴퓨터 판독가능 프로그램 명령어들은 또한 컴퓨터 구현 프로세스를 생성하기 위한 일련의 동작 단계들이 컴퓨터, 다른 프로그래밍가능 장치 또는 다른 디바이스들 상에서 수행되게 하기 위해, 컴퓨터, 다른 프로그래밍가능 데이터 프로세싱 장치, 또는 다른 디바이스 상에 로딩될 수 있다.

[0116] 도면들에서의 플로차트 및 블록 다이어그램들은 본 발명의 다양한 실시예들에 따른 시스템들, 방법들, 및 컴퓨터 프로그램 제품들의 가능한 구현들의 아키텍처, 기능성, 및 동작을 예시하고 있다. 이 점에서, 플로차트 또는 블록 다이어그램들에서의 각각의 블록은 명시된 논리적 기능(들)을 구현하기 위한 하나 이상의 실행가능 명령어를 포함하는 모듈, 세그먼트, 또는 명령어들의 부분을 표현할 수 있다. 일부 대안적인 구현들에서, 블록에 표시된 기능들이 도면들에 표시된 순서와 달리 발생할 수 있다. 예를 들어, 관여된 기능성에 따라, 연속하여 도시된 2개의 블록이, 실제로는, 실질적으로 동시에 실행될 수 있거나, 또는 블록들이 때때로 역순으로 실행될 수 있다. 블록 다이어그램들 및/또는 플로차트 예시의 각각의 블록, 및 블록 다이어그램들 및/또는 플로차트 예시에서의 블록들의 조합들이 명시된 기능들 또는 동작들을 수행하는 특수 목적 하드웨어 기반 시스템들에 의해 구현될 수 있거나, 또는 특수 목적 하드웨어와 컴퓨터 명령어들의 조합들을 수행할 수 있음에 또한 유의해야 할 것이다.

[0117] 전술한 내용이 특정 예들에 관한 것이지만, 다른 예들 및 추가 예들이 그의 기본적인 범위를 벗어나지 않으면서 고안될 수 있으며, 그의 범위는 이하의 청구항들에 의해 결정된다.

도면

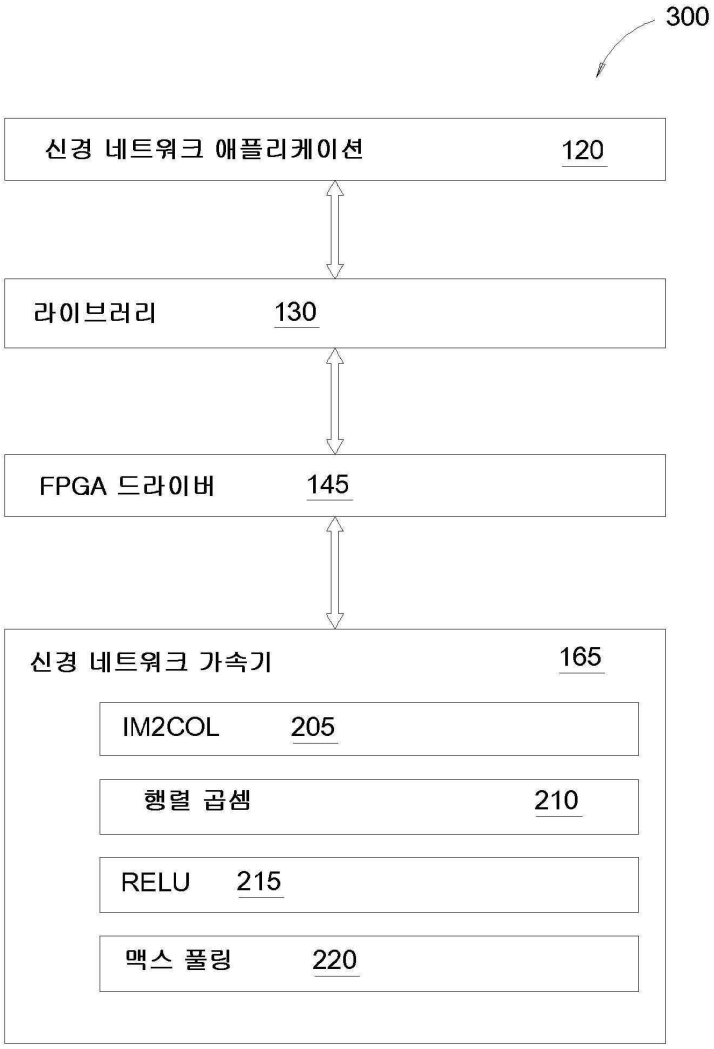
도면1



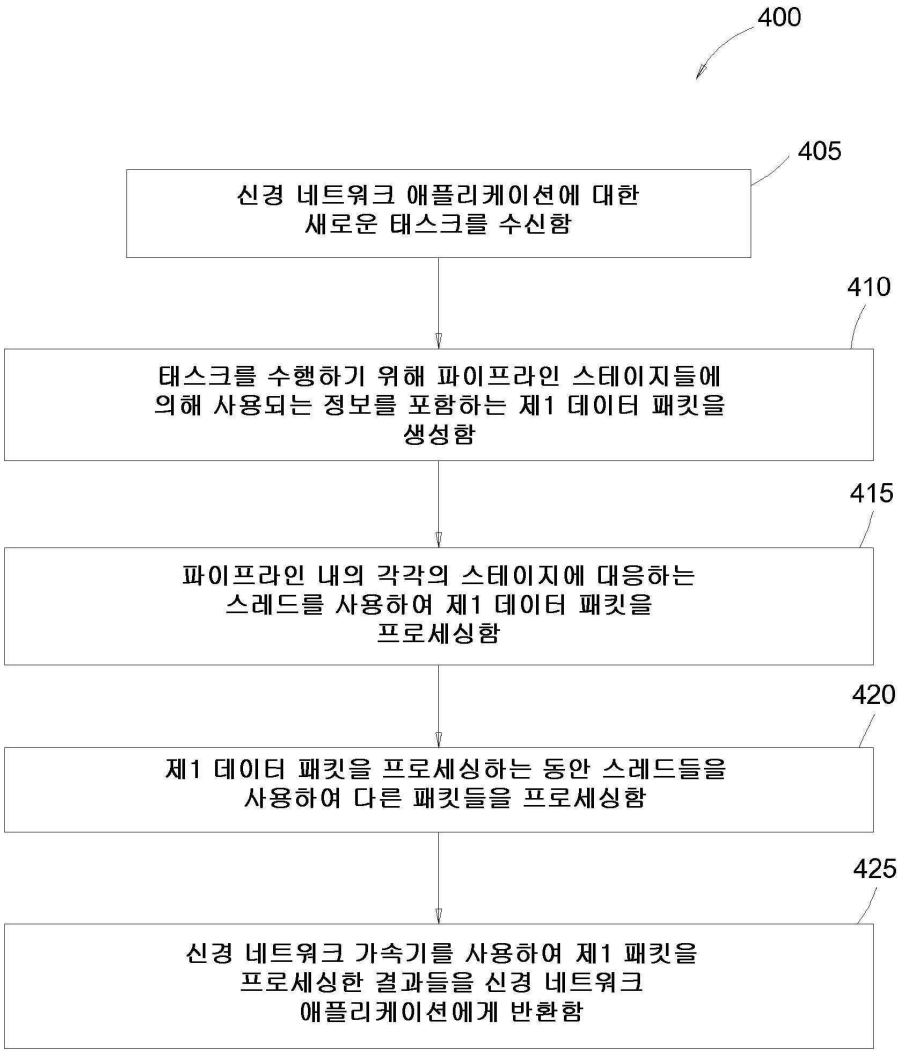
도면2


100
호스트 105
프로세서 110
메모리 115
신경 네트워크 애플리케이션 120
할당된 블록들 125
라이브러리 130
파이프라인 스테이지들 135
메모리 맵 140
FPGA 드라이버 145
FPGA 150
프로그래밍가능 로직 155
커널들 160
신경 네트워크 가속기들 165
메모리 170
할당된 블록들 175

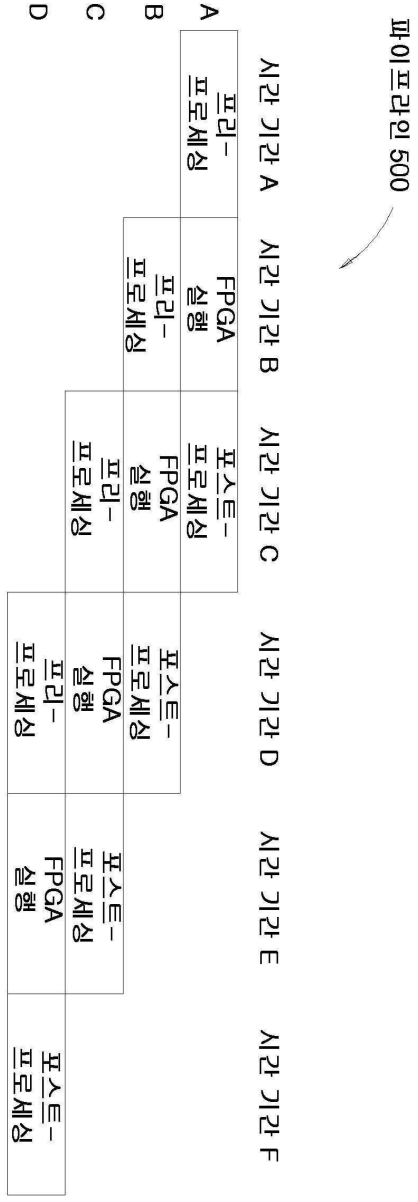
도면3


300
신경 네트워크 애플리케이션 120
라이브러리 130
FPGA 드라이버 145
신경 네트워크 가속기 165
IM2COL 205
행렬 곱셈 210
RELU 215
맥스 풀링 220

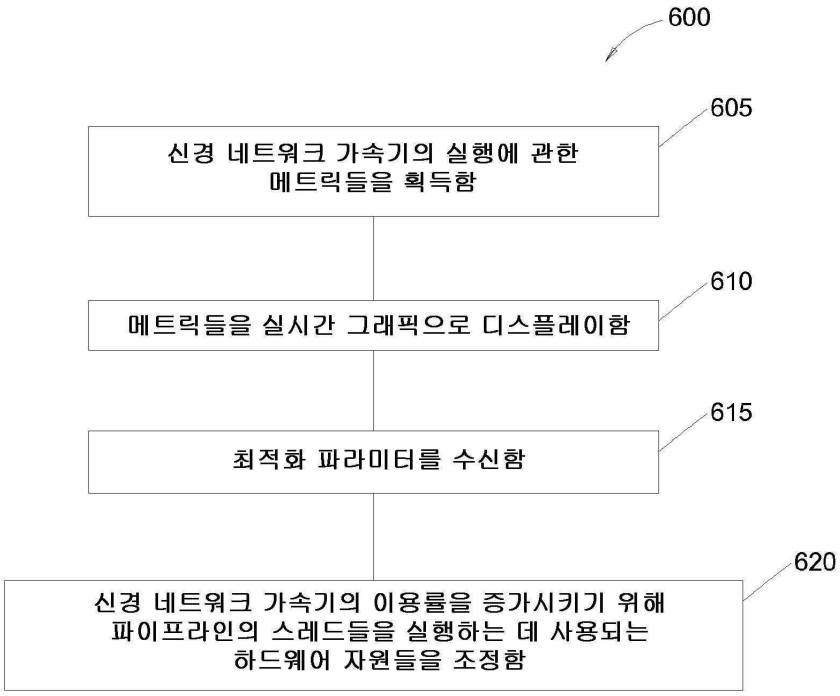
도면4


400
405
신경 네트워크 애플리케이션에 대한 새로운 태스크를 수신함
410
태스크를 수행하기 위해 파이프라인 스테이지들에 의해 사용되는 정보를 포함하는 제1 데이터 패킷을 생성함
415
파이프라인 내의 각각의 스테이지에 대응하는 스레드를 사용하여 제1 데이터 패킷을 프로세싱함
420
제1 데이터 패킷을 프로세싱하는 동안 스레드들을 사용하여 다른 패킷들을 프로세싱함
425
신경 네트워크 가속기를 사용하여 제1 패킷을 프로세싱한 결과들을 신경 네트워크 애플리케이션에게 반환함

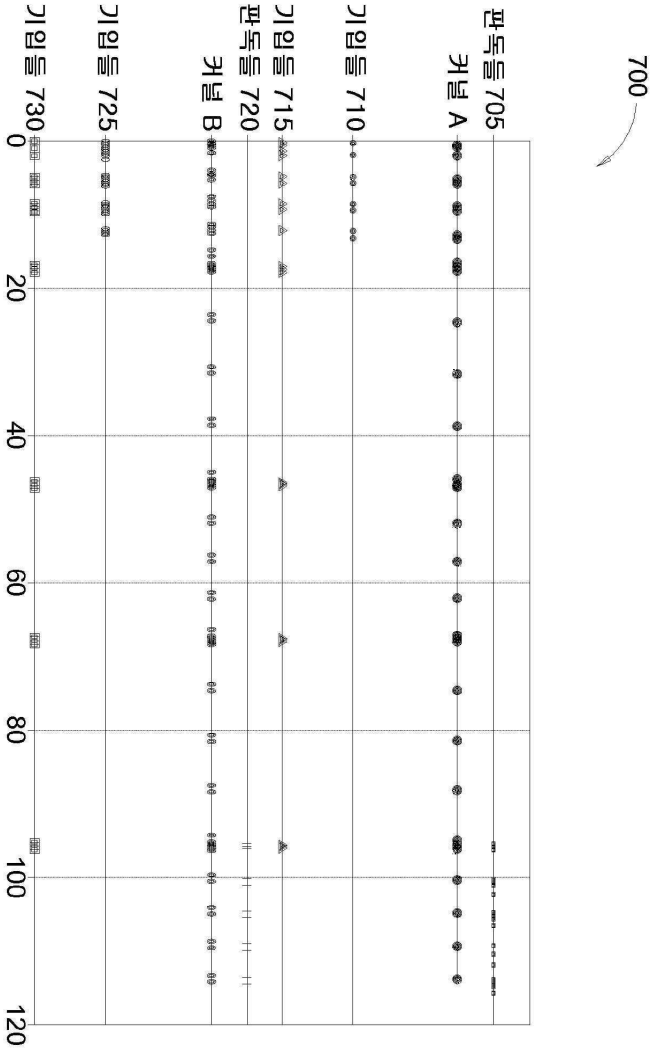
도면5


파이프라인 500
시간 기간 A
시간 기간 B
시간 기간 C
시간 기간 D
시간 기간 E
시간 기간 F
패킷 A
패킷 B
패킷 C
패킷 D
프리-프로세싱
FPGA 실행
포스트-프로세싱

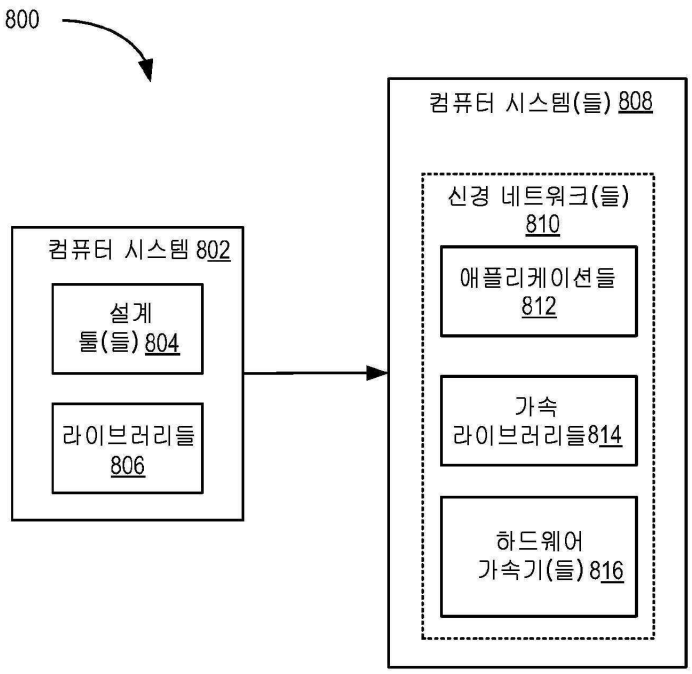
도면6


600
605
신경 네트워크 가속기의 실행에 관한 메트릭들을 획득함
610
메트릭들을 실시간 그래픽으로 디스플레이함
615
최적화 파라미터를 수신함
620
신경 네트워크 가속기의 이용률을 증가시키기 위해 파이프라인의 스레드들을 실행하는 데 사용되는 하드웨어 자원들을 조정함

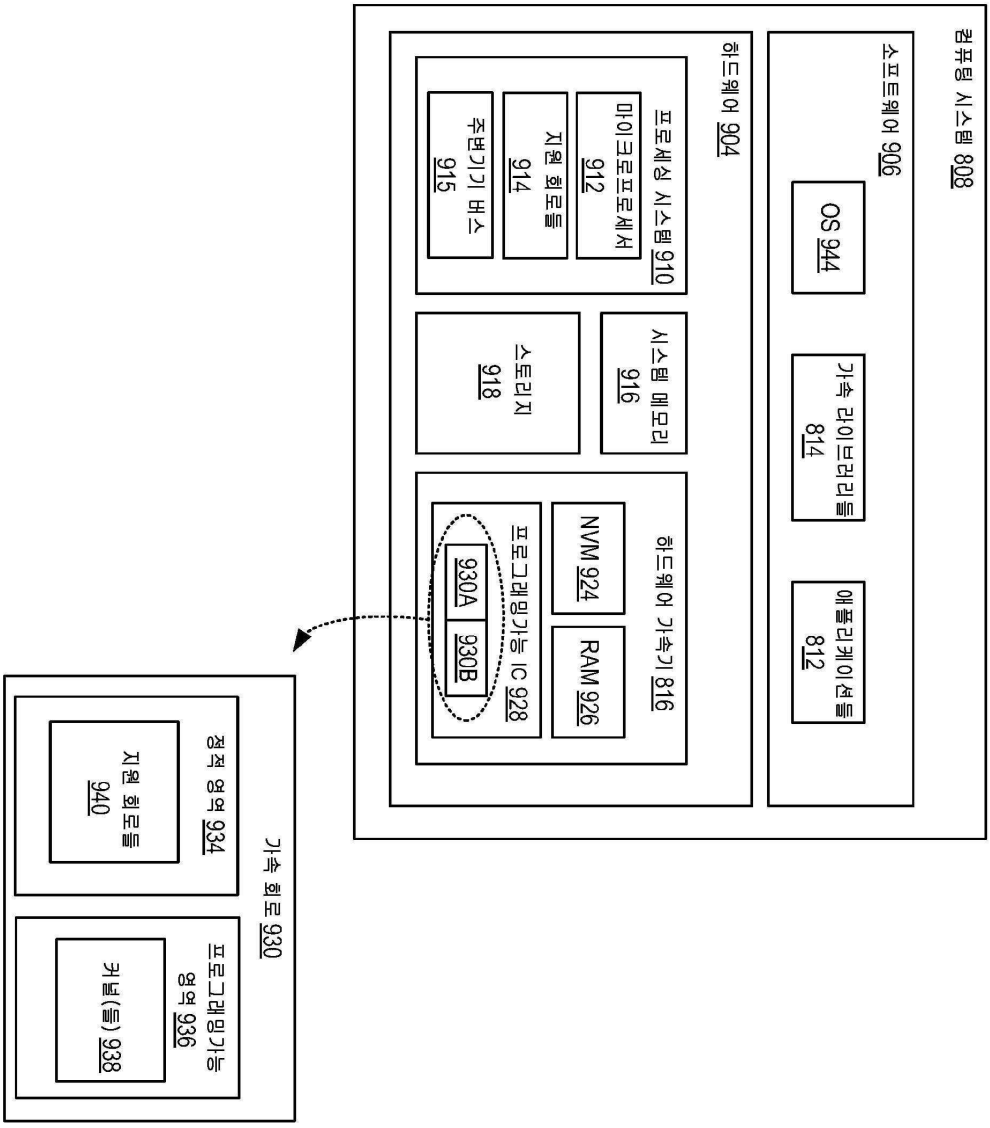
도면7


700
판독들 705
커널 A
DDR 기입들 710
DDR 기입들 715
판독들 720
커널 B
DDR 기입들 725
DDR 기입들 730
0
20
40
60
80
100
120

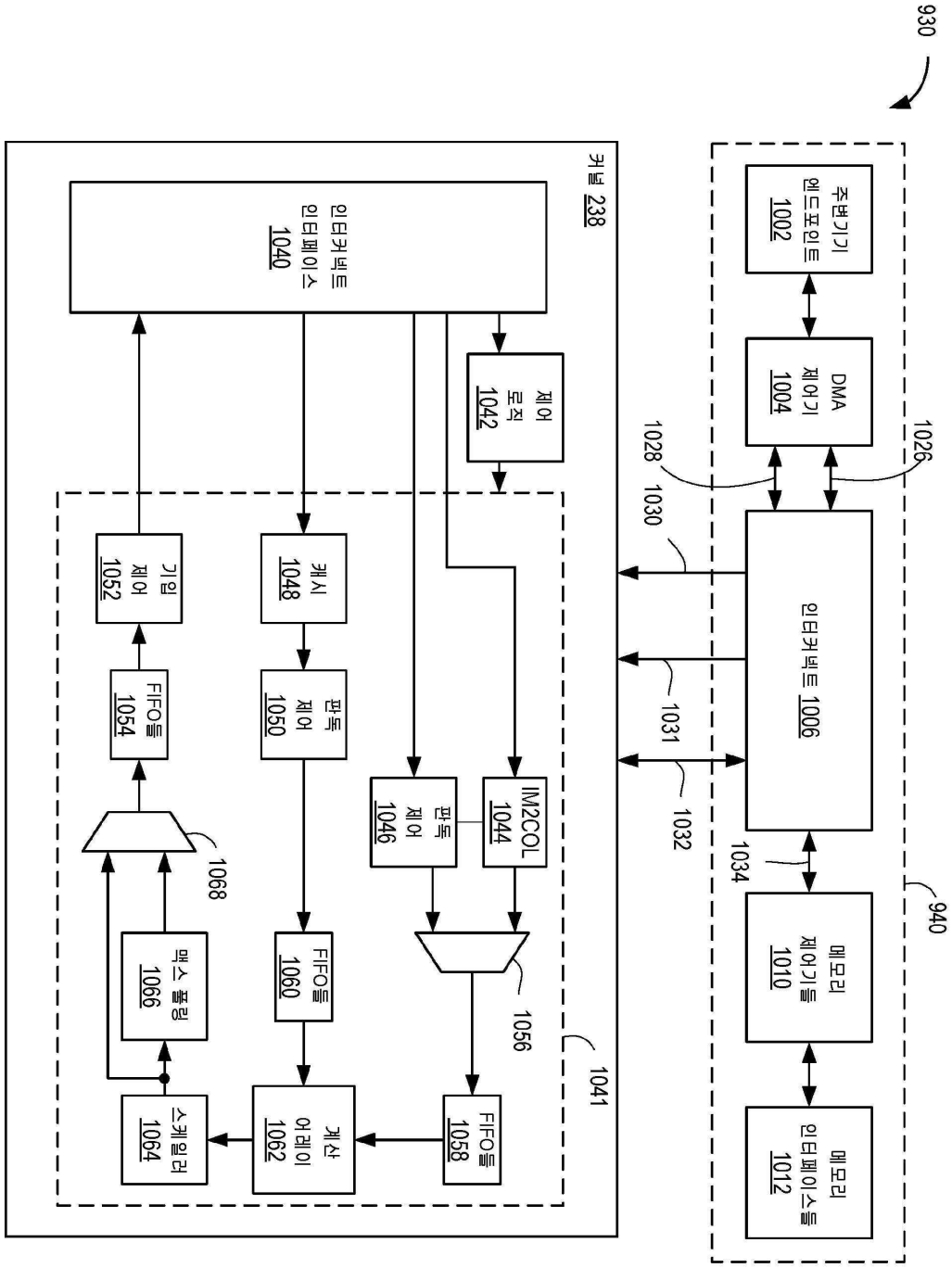
도면8


800
컴퓨터 시스템(들) 808
신경 네트워크(들) 810
애플리케이션들 812
가속 라이브러리들814
하드웨어 가속기(들) 816
컴퓨터 시스템 802
설계 툴(들) 804
라이브러리들 806

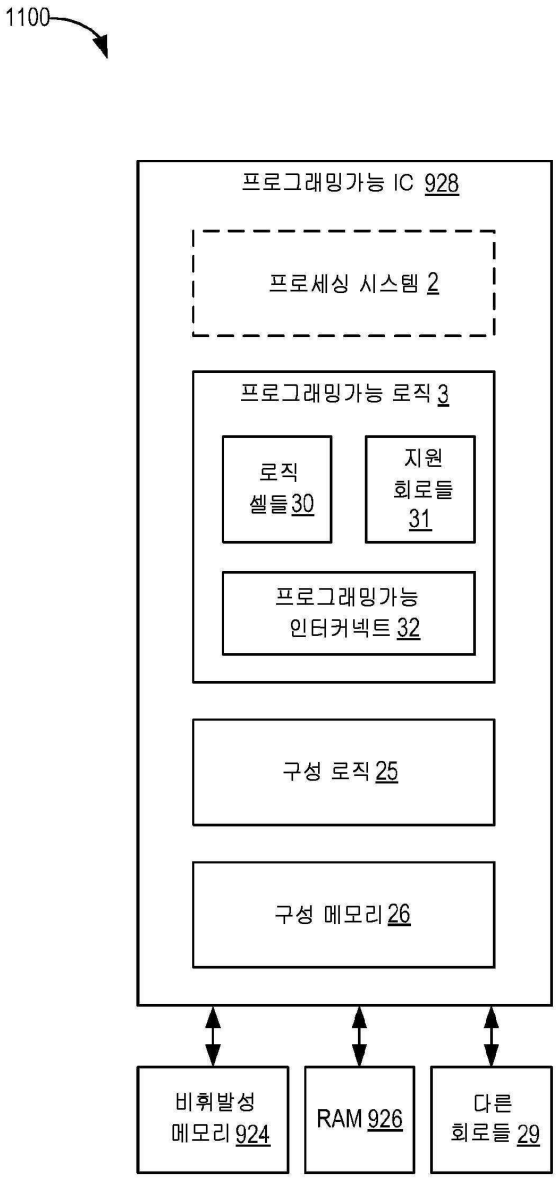
도면9


컴퓨팅 시스템 808
소프트웨어 906
OS 944
가속 라이브러리들 814
애플리케이션들 812
하드웨어 904
프로세싱 시스템 910
마이크로프로세서 912
지원 회로들 914
주변기기 버스 915
시스템 메모리 916
스토리지 918
하드웨어 가속기 816
NVM 924
RAM 926
프로그래밍가능 IC 928
930A
930B
가속 회로 930
정적 영역 934
지원 회로들 940
프로그래밍가능 영역 936
커널(들) 938

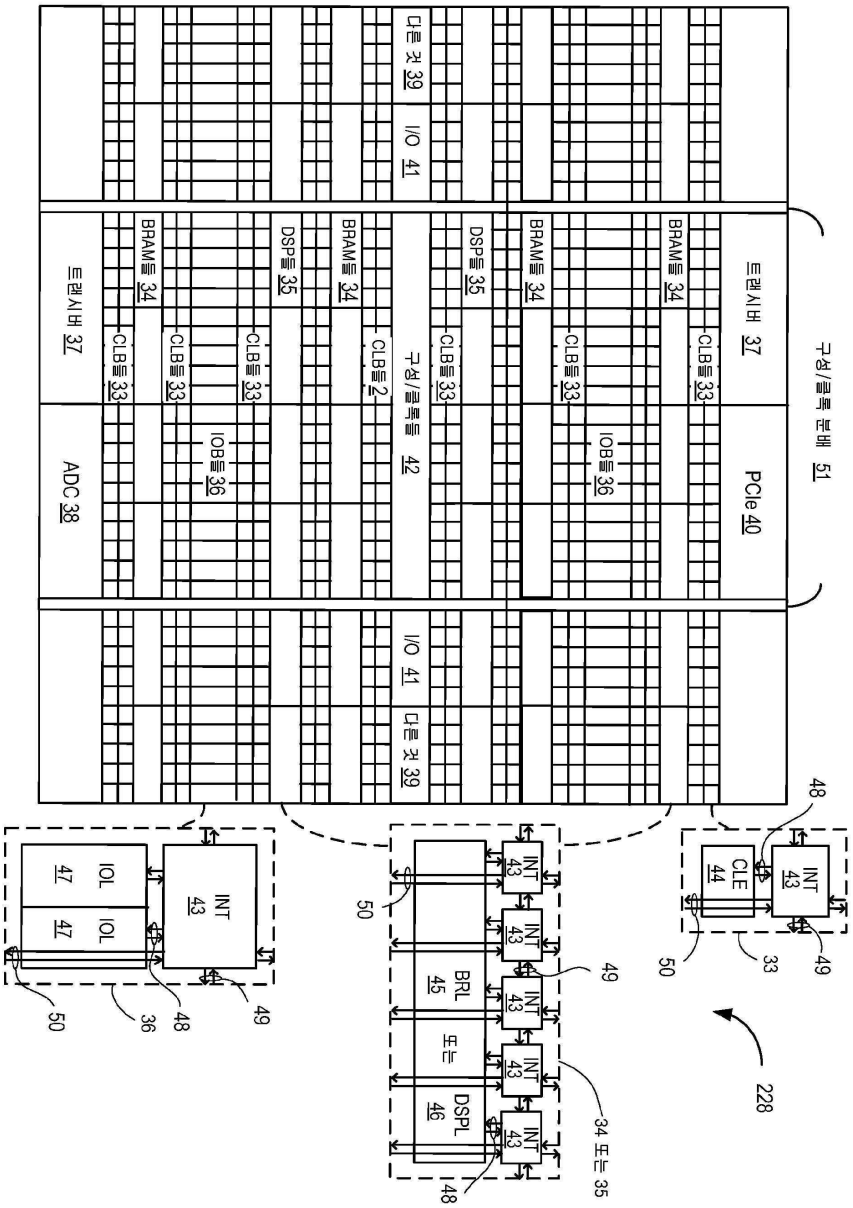
도면10


930
주변기기 엔드포인트 1002
DMA 제어기 1004
1028
1026
1030
인터커넥트 1006
1031
1032
1034
940
메모리 제어기들 1010
메모리 인터페이스들 1012
커널 238
인터커넥트 인터페이스 1040
제어 로직 1042
기입 제어 1052
캐시 1048
IM2COL 1044
FIFO들 1054
판독 제어 1050
판독 제어 1046
1068
맥스 풀링 1066
FIFO들 1060
1056
스케일러 1064
계산 어레이 1062
FIFO들 1058
1041

도면11


1100
프로그래밍가능 IC 928
프로세싱 시스템 2
프로그래밍가능 로직 3
로직 셀들 30
지원 회로들 31
프로그래밍가능 인터커넥트 32
구성 로직 25
구성 메모리 26
비휘발성 메모리 924
RAM 926
다른 회로들 29

도면12


다른 것 39
I/O 41
트랜시버 37
BRAM들 34
CLB들 33
DSP들 35
구성/클록들 42
ADC 38
IOB들 36
PCIe 40
구성/클록 분배 51
IOL 47
INT 43
CLE 44
BRL 45
또는
DSPL 46
228
34 또는 35
48
49
50
33
36