(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0092007
(43) 공개일자 2016년08월03일

(51) 국제특허분류(Int. Cl.)
G06F 3/06 (2006.01) *H04L 29/06* (2006.01)
(52) CPC특허분류
G06F 3/0659 (2013.01)
G06F 3/061 (2013.01)
(21) 출원번호 10-2016-7017509
(22) 출원일자(국제) 2014년12월01일
심사청구일자 2016년06월29일
(85) 번역문제출일자 2016년06월29일
(86) 국제출원번호 PCT/US2014/067927
(87) 국제공개번호 WO 2015/084728
국제공개일자 2015년06월11일
(30) 우선권주장
14/094,273 2013년12월02일 미국(US)

(71) 출원인
마이크론 테크놀로지, 인크.
미국, 아이다호, 보이세, 사우스 페더럴 웨이 8000
(72) 발명자
큐어위츠, 케네스 엠.
미국 95682 캘리포니아주 캐머런 파크 애쉬랜드 코트 408
에일럴트, 션
미국 95663 캘리포니아주 펜린 잉글리시 콜로니 웨이 1721
(뒷면에 계속)
(74) 대리인
양영준, 백만기

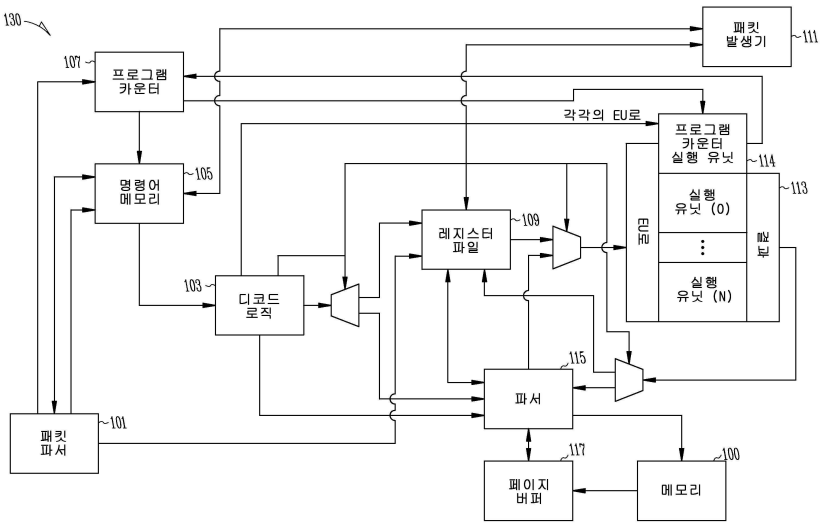
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 **자율 메모리를 위한 방법 및 시스템**

(57) 요 약

방법, 장치 및 시스템이 개시되었다. 방법의 일 실시형태는 자율 메모리 디바이스가 명령어 세트를 수신하는 단계, 메모리 디바이스가 명령어 세트를 실행하는 단계, 명령어 세트에 응답하여 메모리 디바이스로부터 회복된 어느 데이터와 명령어 세트를 패킷으로 조합하는 단계, 및 메모리 디바이스로부터 패킷을 송신하는 단계를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 3/0679 (2013.01)
G06F 3/0688 (2013.01)
H04L 69/22 (2013.01)
G06F 2003/0697 (2013.01)
(72) 발명자
아켈, 아민 디.
미국 95742 캘리포니아주 랜초 코르도바 보더랜즈 드라이브 4214
왕, 홍류
미국 95630 캘리포니아주 폴섬 메레디스 웨이 1225

명 세 서

청구범위

청구항 1

방법으로서,

자율 메모리 디바이스에서 명령어 세트를 수신하는 단계;

상기 메모리 디바이스에서 상기 명령어 세트를 실행하는 단계;

상기 명령어 세트에 응답하여 상기 메모리 디바이스로부터 회복된 어느 데이터와 상기 명령어 세트를 패킷으로 조합하는 단계; 및

상기 메모리 디바이스로부터 상기 패킷을 송신하는 단계를 포함하는 방법.

청구항 2

제1항에 있어서, 상기 메모리 디바이스에서 상기 명령어 세트를 수신하는 단계 및 상기 메모리 디바이스로부터 상기 패킷을 송신하는 단계는 각각 상기 메모리 디바이스에 결합된 네트워크로부터 상기 명령어 세트를 수신하는 단계 및 상기 네트워크에 상기 패킷을 송신하는 단계를 포함하는 방법.

청구항 3

제1항에 있어서, 상기 명령어 세트를 수신하는 단계는 상기 명령어 세트를 포함하는 패킷을 수신하는 단계를 포함하고 그리고 상기 방법은

수신된 상기 명령어 세트와 연관된 초기 프로그램 카운터 값을 프로그램 카운터에 로딩하는 단계;

상기 명령어 세트를 명령어 메모리에 로딩하는 단계; 및

상기 명령어 세트와 연관된 초기 조건 세트를 레지스터 파일에 로딩하는 단계를 포함하는 수신된 상기 패킷을 파싱하는 단계를 더 포함하는 방법.

청구항 4

제3항에 있어서, 상기 명령어 세트를 실행하는 단계는

상기 명령어 세트의 제1 명령어를 실행한 후에 새로운 프로그램 카운터 값을 계산하는 단계; 및

상기 새로운 프로그램 카운터 값을 상기 프로그램 카운터에 저장하는 단계를 포함하는 방법.

청구항 5

제1항에 있어서, 상기 명령어 세트를 실행하는 단계는 제1 실행 유닛에서 제1 명령어를 그리고 제2 실행 유닛에서 제2 명령어를 실행하는 단계를 포함하되 상기 제1 및 제2 명령어의 실행은 실질적으로 병렬인 방법.

청구항 6

제1항에 있어서, 상기 메모리 디바이스는 복수의 노드 중 제1 노드이고 그리고 상기 메모리 디바이스로부터 상기 패킷을 송신하는 단계는 상기 복수의 노드 중 제2 노드에 상기 패킷을 송신하는 단계를 포함하는 방법.

청구항 7

제6항에 있어서,

상기 복수의 노드 중 제3 노드로부터 초기 조건을 수신하는 단계; 및

상기 초기 조건을 파일 레지스터에 저장하는 단계를 더 포함하는 방법.

청구항 8

제1항에 있어서, 상기 명령어 세트는 펜스 플래그를 포함하고 그리고 상기 명령어 세트를 저장하는 단계는

명령어 메모리에서 상기 펜스 플래그 이전의 하나 이상의 명령어 및 상기 명령어 메모리에서 상기 펜스 플래그 다음의 하나 이상의 명령어를 저장하는 단계를 포함하는 방법.

청구항 9

제8항에 있어서,

제1 실행 유닛에서 상기 펜스 플래그 이전의 상기 하나 이상의 명령어를 실행하는 단계; 및

제2 실행 유닛에서 상기 펜스 플래그 이후의 상기 하나 이상의 명령어를 실행하는 단계를 더 포함하는 방법.

청구항 10

제9항에 있어서, 상기 펜스 플래그 이전의 상기 하나 이상의 명령어를 실행하는 단계는 상기 펜스 플래그 이후의 상기 하나 이상의 명령어를 실행하는 단계와 실질적으로 동시에 수행되는 방법.

청구항 11

제1항에 있어서, 상기 명령어 세트를 실행하는 단계는

프로그램 카운터 실행 유닛에 복수의 피연산자를 제공하는 단계;

상기 프로그램 카운터 실행 유닛에 연산자를 제공하는 단계; 및

상기 복수의 피연산자에 대한 상기 연산자의 실행으로부터의 결과에 응답하여 업데이트된 프로그램 카운터 값을 발생시키는 단계를 포함하는 방법.

청구항 12

장치로서,

명령어 및 시작 위치를 포함하는 패킷을 수신하도록 구성된 패킷 파서(packet parser);

상기 패킷 파서에 결합되어 상기 명령어를 수신하도록 구성된 명령어 메모리;

상기 명령어 메모리 및 상기 패킷 파서에 결합된 프로그램 카운터로서, 상기 패킷 파서로부터 상기 시작 위치를 초기에 수신하고 그리고 상기 시작 위치에서 상기 명령어 메모리로부터 명령어를 검색하도록 구성된 상기 프로그램 카운터;

상기 명령어를 실행하도록 상기 명령어 메모리에 결합된 복수의 실행 유닛;

상기 복수의 실행 유닛에 결합된 파서로서, 로컬 메모리로부터 데이터의 판독을 제어하도록 구성된 상기 파서;

상기 파서 및 상기 명령어 메모리에 결합되어 상기 파서 및 상기 패킷 파서로부터의 상기 데이터를 저장하도록 구성된 레지스터 파일; 및

상기 명령어 메모리 및 상기 레지스터 파일에 결합된 패킷 발생기로서, 상기 명령어 세트 및 상기 데이터를 포함하는, 송신을 위한 패킷을 발생시키도록 구성된 상기 패킷 발생기를 포함하는 장치.

청구항 13

제12항에 있어서, 상기 복수의 실행 유닛의 각각은

복수의 산술 로직 유닛(ALU); 및

상기 복수의 산술 로직 유닛 중 적어도 2개의 출력 사이에 결합된 멀티플렉싱 기능을 포함하는 장치.

청구항 14

제13항에 있어서, 상기 복수의 ALU은 상기 명령어로부터의 각각의 명령어와 연관된 ALU을 포함하는 장치.

청구항 15

제13항에 있어서, 상기 복수의 실행 유닛의 각각은 if-then-else 문을 구현하는 장치.

발명의 설명

기 술 분 야

[0001] **우선권 출원**

[0002] 본 출원은 2013년 12월 2일자로 출원된 미국 출원 제14/094,273호에 대한 우선권의 이익을 주장하며, 그의 전문은 참고로 본 명세서에 편입된다.

배 경 기 술

[0003] 전형적으로 메모리 디바이스는 컴퓨터 또는 다른 전자 디바이스에서 내부, 반도체, 집적 회로로서 제공된다. 램(RAM), 롬(ROM), 동적 램(DRAM), 동기식 동적 램(SDRAM), 및 비-휘발성(예를 들어, 플래시) 메모리를 포함하는 여러 다른 유형의 메모리가 많이 있다.

[0004] 소정 수의 비-휘발성 메모리 디바이스는 컴퓨터 시스템에서 기계적으로-동작되는 하드 디스크 드라이브를 에뮬레이팅할 수 있는 고체 상태 드라이브(SSD)를 제작하도록 조합될 수 있다. 고체 상태 드라이브는 움직이는 부품이 없기 때문에 기계적 하드 드라이브보다 더 큰 신뢰도로 더 고속의 액세스를 제공할 수 있다.

[0005] 컴퓨터 시스템의 증가하는 성능에 적어도 부분적으로 기인하여, 메모리 및 고체 상태 드라이브 제조자는 컴퓨터 시스템 성능 증가와 보조를 맞추려고 하기 위해 그들 메모리의 성능을 증가시키도록 일정한 압박을 받고 있을 수 있다. 컴퓨터 시스템 상의 어느 연산 부담이라도 완화하도록 메모리로의 기록 및 판독을 더 효율적이게 할 일반적 필요성이 있다.

도면의 간단한 설명

[0006] 도 1은 자율 메모리 프로세싱 장치의 일 실시형태의 예시적 기능적 블록 선도;

도 2는 도 1의 실시형태에 따른 패킷 파서(packet parser)의 일 실시형태의 예시적 블록 선도;

도 3은 도 1의 실시형태에 따른 프로그램 카운터의 일 실시형태의 예시적 블록 선도;

도 4는 도 1의 실시형태에 따른 명령어 메모리의 일 실시형태의 예시적 블록 선도;

도 5는 도 1의 실시형태에 따른 디코드 로직의 일 실시형태의 예시적 블록 선도;

도 6은 도 1의 실시형태에 따른 레지스터 파일의 일 실시형태의 예시적 블록 선도;

도 7a 및 도 7b는 도 1의 실시형태에 따른 실행 유닛의 일 실시형태의 예시적 블록 선도;

도 8은 도 1의 실시형태에 따른 파서의 일 실시형태의 예시적 블록 선도;

도 9는 도 1의 실시형태에 따른 패킷 발생기의 일 실시형태의 예시적 블록 선도;

도 10은 도 1의 실시형태에 따른 명령어 실행을 위한 포맷의 일 실시형태의 예시적 선도;

도 11은 메모리 시스템의 일 실시형태의 예시적 블록 선도; 및

도 12는 자율 메모리 디바이스에서의 자율 메모리 프로세싱 장치의 연산의 일 실시형태의 예시적 순서도.

발명을 실시하기 위한 구체적인 내용

[0007] 이하의 상세한 설명에서는, 그 일부분을 형성하고 있는 그리고 특정 실시형태를, 예시의 방식으로, 보여주고 있는 첨부 도면을 참조한다. 도면에서는, 유사한 숫자가 수개의 도면의 곳곳에서 실질적으로 유사한 컴포넌트를 기술한다. 본 발명의 범위로부터 벗어남이 없이 구조적, 논리적 및 전기적 변경이 이루어질 수도 있고 다른 실시예가 이용될 수도 있다. 그러므로, 이하의 상세한 설명은 한정적 의미로 받아들여지려는 것이 아니다.

[0008] 본 발명은 어느 일 유형의 메모리로도 한정되지 않는다. 자율 메모리 프로세싱 장치는 반도체 메모리, 광학 메모리 또는 자기 메모리를 포함하는 어느 유형의 메모리 디바이스, 메모리 디바이스 그룹 또는 메모리 기술과도

연관될 수 있다. 예를 들어, 메모리는 비-휘발성(예를 들어, NAND 플래시, NOR 플래시, 상 변화 메모리(PCM)) 또는 휘발성(예를 들어, DRAM, SRAM)을 포함할 수 있다.

[0009] 여기에서 사용되는 바와 같이, 노드는 수신된 패킷을 파싱하기 위한 패킷 파서, 노드로부터 네트워크로 송신될 패킷을 발생시키기 위한 패킷 발생기, 및 노드를 어느 네트워크와도 인터페이싱시킬 수 있는 네트워크 포트를 포함할 수 있다. 노드는 부가적으로는 노드의 동작을 제어하기 위한 프로세싱 엘리먼트는 물론 데이터를 저장하기 위한 메모리도 포함할 수 있다. 다른 실시형태에 있어서, 노드는 부가적 기능을 위한 부가적 하드웨어 및/또는 소프트웨어/펌웨어를 포함할 수 있다. 자율 프로세싱 장치를 갖는 자율 메모리 디바이스는 노드라고 생각될 수 있다.

[0010] 도 1은 자율 메모리 프로세싱 장치의 일 실시형태의 기능적 블록 선도를 예시하고 있다. 그러한 장치는 메모리(100)와 연관될 수 있고 중앙 프로세싱 유닛(CPU)-기반 컴퓨팅 시스템에서 메모리 대역폭 병목을 완화하도록 사용될 수 있다. 자율 메모리 프로세싱 장치는 자율 메모리 디바이스에 위치할 수 있다.

[0011] 자율 메모리 프로세싱 장치는 패킷 파서(101), 프로그램 카운터(107), 명령어 메모리(105), 디코드 로직(103), 레지스터 파일(109), 파서(115), 패킷 발생기(111), 하나 이상의 실행 유닛(EU)(113), 및 페이지 버퍼(117)를 포함할 수 있다. 다른 실시형태가 다른 엘리먼트 및 다른 아키텍처를 사용할 수 있으므로 도 1의 엘리먼트 및 아키텍처는 단지 예시의 목적을 위한 것일 뿐이다.

[0012] 도 2는 패킷 파서(101)의 블록 선도를 예시하고 있다. 패킷 파서(101)는 네트워크(예를 들어, 메모리(100) 외부 네트워크)에 결합되어 그로부터의 데이터 패킷을 받아들일 수 있다. 패킷 파서(101)는 또한 패킷 파서(101)가 네트워크로부터 패킷으로 수신된 프로그램 카운트(예를 들어, 명령어 메모리 위치)를 프로그램 카운터(107)에 로딩할 수 있도록 프로그램 카운터(107)의 입력에 결합될 수 있다. 패킷 파서(101)는 또한 프로그램 카운터(107)가 그 현재 프로그램 카운트(예를 들어, 명령어 메모리 위치)를 패킷 파서(101)에 로딩할 수 있도록 프로그램 카운터(107)의 출력에 결합될 수 있다. 패킷 파서(101)는 명령어 메모리(105) 및 레지스터 파일(109) 내에 네트워크로부터 패킷으로 수신된 데이터(예를 들어, 명령어)의 로딩을 가능하게 하도록 명령어 메모리(105) 및 레지스터 파일(109)의 입력에 더 결합될 수 있다.

[0013] 도 3은 프로그램 카운터(107)의 블록 선도를 예시하고 있다. 예시의 목적으로, 프로그램 카운터(107)는 32 비트 카운터로서 도시되어 있다. 그렇지만, 다른 실시형태는 다른 프로그램 카운터 사이즈를 사용할 수도 있다.

[0014] 프로그램 카운터(107)는 하나 이상의 실행 유닛(113)의 일부분일 수 있는 프로그램 카운터 실행 유닛(PCEU)(114) 및 패킷 파서(101)로부터의 입력을 가질 수 있다. 프로그램 카운터(107)는 명령어 메모리(105)에 결합된 출력을 가질 수 있다.

[0015] 프로그램 카운터(107)는 프로그램(예를 들어, 실행가능한 명령어)을 포함하고 있을 수 있는 명령어 메모리(105)에서의 특정 명령어 위치에 액세스하도록 프로그램 카운트 값(예를 들어, 명령어 메모리 위치)을 포함하고 있을 수 있다. 프로그램 카운트 값은, 패킷 파서(101)에 의해 결정되고 그로부터 수신되는 바와 같이, 들어오는 패킷에서의 특정 데이터 필드로부터 설정될 수 있거나, 또는 프로그램 카운터 실행 유닛(114)으로부터의 계산된 값일 수 있다. 프로그램 카운터(107)는 그 후 명령어 메모리(105)에 프로그램 카운트의 값(예를 들어, 32-비트 레지스터)을 출력할 수 있다.

[0016] 도 4는 명령어 메모리(105)의 블록 선도를 예시하고 있다. 명령어 메모리(105)는 프로그램(예를 들어, 실행가능한 명령어)을 저장하기 위한 소정 수의 레지스터를 포함할 수 있다. 패킷 파서(101)는 명령어 메모리(105)의 기록 포트에 결합될 수 있다. 명령어 메모리(105)는, 패킷 파서(101)에 의해 결정되는 바와 같은, 들어오는 패킷 내 수신된 명령어가 패킷으로부터 명령어 메모리(105)에 로딩될 수 있게 되도록 패킷 파서(101)에 의해 기록될 수 있다.

[0017] 명령어 메모리(105)는 명령어 메모리(105) 내의 특정 위치에 액세스하기 위한 주소를 각각 받아들일 수 있는 2개의 주소 포트를 포함할 수 있다. 하나의 주소는 프로그램 카운터(107)로부터 유래할 수 있다. 다른 주소는 패킷 발생기(111)로부터 유래할 수 있다.

[0018] 하나의 연산 동안, 명령어 메모리(105)는 프로그램 카운터(107)의 주소에 의해 나타낸 위치로부터의 명령어(예를 들어, 데이터 포트)를 출력할 수 있다. 이러한 명령어는 수행할 연산에 관하여 실행 유닛(113)에 명령하기 위해 실행 유닛(113)에 의해 디코딩 및 실행될 수 있다. 이러한 명령어는 실행 유닛(113)에 피연산자는 물론 레지스터 파일(109)로의 인덱스도 주어 프로세싱을 위해 실행 유닛(113)에 무슨 데이터를 출력할지에 관하여 레지

스터 파일(109)에 명령할 수 있다.

[0019] 도 5는 디코드 로직(103)의 블록 선도를 예시하고 있다. 디코드 로직(103)은 실행 유닛 디코드 로직(501), 파서 디코드 로직(502), 및 디멀티플렉싱 기능(503)(예를 들어, 디멀티플렉서)을 포함할 수 있다.

[0020] 디멀티플렉싱 기능(503)으로의 입력은 명령어 메모리(105)의 출력으로부터의 명령어 스트림에 결합될 수 있다. 명령어 스트림에서의 하나 이상의 제어 비트는 명령어 스트림에서의 특정 명령어의 수신지(예를 들어, EU 디코드 로직(501), 파서 디코드 로직(502))를 선택하도록 사용될 수 있다.

[0021] 명령어가 EU 디코드 로직(501)으로 발신되면, EU 디코드 로직(501)은 실행 유닛(113) 중 하나에 명령어를 발신하기 위해 명령어를 프로세싱할 수 있다. 명령어는 무슨 유형의 연산을 수행할지에 관하여는 물론 실행 유닛(113) 중 하나에 명령어의 실행 동안 사용될 피연산자를 주기 위해서도 실행 유닛(113) 중 하나에 명령할 수 있다. 피연산자는 레지스터 파일(109)의 레지스터 내에 인덱싱되고 실행 유닛(113) 중 하나가 그 데이터를 프로세싱할 수 있도록 무슨 데이터를 출력할지에 관하여 그 레지스터에 명령할 수 있다.

[0022] 디멀티플렉싱 기능(503)은 또한 파서(115)에 결합되는 파서 디코드 로직(502)에 명령어를 발신할 수 있다. 명령어는 프로세싱을 위해 페이지 버퍼(117)의 특정 세그먼트로부터 실행 유닛(113) 중 하나로 데이터를 판독하기 위해 페이지 버퍼(117)의 어느 세그먼트에 액세스할지 파서에 순차적으로 명령하는 파서 디코드 로직(502)을 제어할 수 있다.

[0023] 도 6은 레지스터 파일(109)의 블록 선도를 예시하고 있다. 레지스터 파일(109)은 패킷 파서(101), 패킷 발생기(111), 실행 유닛(113) 중 하나 이상, 및 메모리 판독 표시로부터의 입력을 포함할 수 있다. 메모리 판독 표시는 메모리 연산이 완료되었을 때를 나타내는 파서(115)에 의해 발생되는 신호일 수 있다. 레지스터 파일(109)은 패킷 발생기(111), 실행 유닛(113), 및 파서(115)로의 출력을 포함할 수 있다.

[0024] 레지스터 파일(109)은 실행 유닛(113)에 의한 프로세싱이 일어나고 있는 동안 변수를 저장할 메모리(예를 들어, 복수의 레지스터)를 포함할 수 있다. 이들 변수는 하나 이상의 명령어에 응답하여 메모리로부터 검색된 데이터를 포함할 수 있다. 레지스터 파일(109)은 레지스터 내의 초기 조건을 설정하기 위해 패킷 파서(101)에 의해 기록될 수 있고 패킷 발생기(111)에 의해 판독될 수 있다. 실행 유닛(113)의 각각은 멀티플렉싱 기능을 통해 레지스터 파일(109)로부터 인수를 수신할 수 있다. 패킷 발생기(111)로의 출력은 레지스터 파일(109)의 레지스터에 저장된 데이터를 네트워크로의 송신을 위한 패킷으로 번들링하도록 사용될 수 있다.

[0025] 도 7a는 일반적으로 실행 유닛(113)(예를 들어, 실행 유닛(0-N))의 일 실시형태의 블록 선도를 예시하고 있는 한편 도 7b는 구체적으로 프로그램 카운터 실행 유닛(114)의 일 실시형태의 블록 선도를 예시하고 있다. PCEU(114)는 실행 유닛(113)의 그룹의 일부분이라고 생각될 수 있지만 다른 실행 유닛(113)과는 다른 아키텍처를 가질 수 있다.

[0026] 특정 자율 메모리 프로세싱 장치에 포함될 수 있는 실행 유닛(113)의 특정 수에 대한 요건은 없다. 하나의 장치는 단일 실행 유닛(113)을 가질 수 있는 한편 다른 장치는 다수(예를 들어, 수백)의 실행 유닛을 가질 수 있다.

[0027] 도 7a는 실행 유닛(113)이 4개의 산술 로직 유닛(ALU)(701 내지 704)을 포함할 수 있음을 예시하고 있다. ALU1(703) 및 ALU2(704)의 출력은 멀티플렉싱 기능(706)에 입력될 수 있다. 어느 ALU(703, 704) 출력이 선택되는지는 멀티플렉싱 기능(706)을 위한 선택 신호로서 출력이 사용될 수 있는 Comp ALU(702)의 출력에 의해 결정될 수 있다. 제4 ALU, ALU Out(701)은 실행 유닛(113)에 의해 수행된 연산의 결과를 어디에 저장할지 레지스터 파일(109)에 나타낼 수 있는 레지스터 파일(109)로의 레지스터 주소(Rd)로서의 출력을 가질 수 있다.

[0028] 하위 3개의 ALU(702-704) 및 멀티플렉싱 기능(706)은 if-then-else 연산을 수행할 수 있다. 멀티플렉싱 기능(706)은 "소정 조건이면"을 제공할 수 있고 여기서 그 조건은 Comp ALU(702)에 의해 결정된다. 그리하여, 조건이 참이면, 그때 하나의 ALU(예를 들어, ALU1(703))의 출력이 Comp ALU(702)의 출력에 의해 선택되고, 그렇지 않으면 다른 ALU(예를 들어, ALU2(704))의 출력이 Comp ALU(702)의 출력에 의해 선택된다.

[0029] 예를 들어, ALU1(703)이 피연산자 입력(OPERAND1(R_1), OPERAND2(R_2)) 및 커맨드 입력(OPERATOR1)을 갖고 ALU2(704)이 피연산자 입력(OPERAND3(R_3), OPERAND4(R_4)) 및 커맨드 입력(OPERATOR2)을 갖는다고 가정되면, if-then-else 문은 다음처럼 보일 수 있다:

if (Condition)

then

Operand1 OPERATOR1 Operand2

else

Operand3 OPERATOR2 Operand4

[0030]

[0031] 여기서 "Operand1 OPERATOR1 Operand2"는 ALU1(703)에 의해 제공될 수 있고, "Operand3 OPERATOR2 Operand4"는 ALU2(704)에 의해 제공될 수 있고, "if (Condition)"은 Comp ALU(702) 및 멀티플렉싱 기능(706)에 의해 제공될 수 있다.

[0032] 도 10의 명령어의 포맷에 관하여 후속하여 설명되는 바와 같이, 피연산자 및 연산자는 명령어에 의해 제공될 수 있거나 또는 명령어는 어느 레지스터에 피연산자 값이 위치하고 있는지 나타낼 수 있다. 예를 들어, OPERAND1(R1)는 레지스터(R_1)에 위치하고 있을 수 있고, OPERAND(R2)는 레지스터(R_2)에 위치하고 있을 수 있고, OPERAND(R3)는 레지스터(R_3)에 위치하고 있을 수 있고, OPERAND(R4)는 레지스터(R_4)에 위치하고 있을 수 있다.

[0033] ALU1(703) 및 ALU2(704)는 동일한 연산 또는 다른 연산을 수행할 수 있다. 환언하면, OPERATOR1는 OPERATOR2와 동일할 수 있거나 또는 OPERATOR1는 OPERATOR2와 다를 수 있다.

[0034] ALU Out(701)은 피연산자 입력(R_5, R_6)(예를 들어, 레지스터(R_5, R_6)) 및 커맨드 입력(OPERATOR3)을 가질 수 있다. (R_5) 및 (R_6)로부터의 값에 대해 커맨드(OPERATOR3)를 수행하는 ALU Out(701)에 의해 발생되는 바와 같은 결과(R_d)는 어디에 실행 유닛(113)의 결과가 저장되는지 결정한다.

[0035] Comp ALU(702)은 피연산자 입력(R_7, R_8)(예를 들어, 레지스터(R_7, R_8)) 및 커맨드 입력(OPERATOR4)을 가질 수 있다. 앞서 논의된 바와 같이, (R_7) 및 (R_8)로부터의 값에 대해 커맨드(OPERATOR4)를 수행하는 결과는 멀티플렉싱 기능(106)의 선택을 결정한다.

[0036] 위의 ALU(701 내지 704)에서 커맨드(예를 들어, OPERATOR1, OPERATOR2, OPERATOR3, OPERATOR4)로서 사용될 수 있는 전형적 연산은 가산, 감산, 논리곱, 논리합, 논리부정, 부정 논리합, 같음, 작거나 같음, 작음, 같지 않음, 크거나 같음, 또는 큼을 포함할 수 있다. 다른 실시예는 다른 연산을 사용할 수 있으므로 이들 연산은 단지 예시의 목적을 위한 것이다.

[0037] 도 7b는 프로그램 카운터 실행 유닛(PCEU)(114)의 아키텍처를 예시하고 있다. 이러한 아키텍처는 실행 유닛0-n(113)과 유사하지만 ALU Out(701)이 없을 수 있다. PCEU(114)는 프로그램 카운터(107)에 대한 새로운 주소를 결정하는 것에 전용일 수 있으므로, PCEU(114) 연산의 결과를 저장할 위치가 프로그램 카운터(107)일 것이므로 ALU Out(701)은 포함되지 않는다.

[0038] PCEU(114)는 피연산자 입력(R9, R10) 및 커맨드 입력(OPERATOR5)을 갖는 Comp ALU(710)을 포함할 수 있다. ALU1(711)은 피연산자 입력(R11, R12) 및 커맨드 입력(OPERATOR6)을 포함할 수 있다. ALU2(712)은 피연산자 입력(R13, R14) 및 커맨드 입력(OPERATOR7)을 포함할 수 있다.

[0039] ALU1(711) 및 ALU2(712)의 출력은 멀티플렉싱 기능(714)으로의 입력일 수 있다. Comp ALU(710)의 출력은 멀티플렉싱 기능(714)을 위한 선택 신호를 제공할 수 있다. 그리하여, 앞서 설명된 실행 유닛(113)에서와 같이, PCEU(114)은 if-then-else 문을 제공할 수 있으며 여기서 멀티플렉싱 기능(714)은 "소정 조건이면"을 제공하고 그 조건은 Comp ALU(710)에 의해 결정된다. 그리하여, 조건이 참이면, 그때 하나의 ALU(예를 들어, ALU1(711))의 출력이 Comp ALU(710)의 출력에 의해 선택되고, 그렇지 않으면 다른 ALU(예를 들어, ALU2(712))의 출력이 Comp ALU(710)의 출력에 의해 선택된다. 그 결과는 프로그램 카운터(107)에 로딩될 수 있다.

[0040] 앞서 설명된 실행 유닛(113)에서와 같이, PCEU(114)에서 사용될 연산자 및 커맨드는 명령어 메모리로부터의 명령어로부터 로딩될 수 있거나 또는 명령어는 어느 레지스터가 그 값을 포함하고 있을 수 있는지 나타낼 수 있다.

[0041] 도 8은 파서(115)의 블록 선도를 예시하고 있다. 파서(115)는 기록될 주소는 물론 데이터도 포함하는 메모리 기

록 포트를 포함할 수 있다. 메모리 판독 주소 포트는 판독된 데이터가 메모리 판독 데이터 포트 내로 판독될 수 있도록 판독할 메모리로의 주소를 제공할 수 있다. 파서(115)는 또한 메모리 판독 연산이 완료되었을 때 메모리 판독 표시 신호를 출력할 수 있다. 파서(115)는 실행 유닛(113)으로의 출력, 레지스터 파일(109)로부터의 입력, 및 파서 디코드 로직(502)으로부터의 구성 입력을 더 포함할 수 있다.

[0042] 파서(115)는 직접 그것이 메모리(100)의 페이지 버퍼(117)에 기록 또는 그로부터 판독할 수 있도록 메모리(100)로의 직접 액세스를 가질 수 있다. 파서(115)는 페이지 버퍼(117)의 전체 길이로의 액세스를 갖고 그래서, 프로세싱을 더 관리가능하게 하기 위해, 그것은 페이지 버퍼(117)를 더 작은 세그먼트(예를 들어, 규칙적으로 획정된 세그먼트)로 세분할 수 있다. 예를 들어, 파서(115)는 페이지 버퍼의 처음 100 바이트에 대해 연산하고, 그 후 다음 100 바이트, 그리고 전체 페이지 버퍼(117)가 판독/기록될 때까지 이것을 계속할 수 있다. 이것을 성취하기 위해, 파서(115)에는 페이지 버퍼(117)의 어느 세그먼트로부터 판독할지 결정하는 패킷 파서(101)로부터의 주소가 주어질 수 있다.

[0043] 파서(115)는 페이지 버퍼(117)의 컨텐트를 어떻게 파싱할지 파서(115)에 명령할 수 있는 레지스터 파일(109)로부터의 구성 입력을 수신할 수 있다. 파서(115)는 레지스터 파일(109)에서 새로운 컨텐트가 이용가능하다고 실행 프로그램에 명령하는 메모리 판독 표시 신호를 발생시킬 수 있다.

[0044] 도 9는 패킷 발생기(111)의 일 실시형태의 블록 선도를 예시하고 있다. 패킷 발생기는 명령어 메모리(105) 및 레지스터 파일(109)로부터의 입력 그리고 명령어 메모리(105) 및 레지스터 파일(109)로의 출력을 포함할 수 있다. 패킷 발생기(111)는 부가적으로는 어느 발생된 패킷이라도 출력하기 위해 네트워크로의 출력을 갖는다.

[0045] 패킷 발생기(111)는 이들 엘리먼트(105, 109)로부터 데이터를 판독하기 위해 명령어 메모리(105)에 대한 주소 및 레지스터 파일(109)에 대한 주소를 발생시킬 수 있다. 패킷 발생기(111)는 그 후 레지스터 파일(109)로부터의 판독된 데이터(예를 들어, 명령어 메모리(105)로부터의 명령어 및 컨텍스트(예를 들어, 데이터, 메모리 판독으로부터의 결과, 수행된 연산으로부터의 결과))를 사용하고, 이러한 데이터를 번들링하고, 그리고 네트워크를 통하여 송신될 패킷을 발생시킬 수 있다.

[0046] 도 10은 도 1의 실시형태에 따른 명령어 실행의 포맷의 일 실시형태를 예시하고 있다. 각각의 명령어(1001 내지 1003)는 실행 유닛(113)에 의한 실행을 위해 명령어 메모리에 저장될 수 있다.

[0047] 명령어의 예시된 실시형태는 4개의 명령어(1000 내지 1003)를 포함한다. 각각의 명령어는 실행 유닛(113)의 서로 다른 ALU와 연관될 수 있다. 그리하여, 실행 유닛(113)이 다른 수량의 ALU을 포함하면, 실행 포맷은 다른 수량의 명령어(1000 내지 1003)를 포함할 수 있다. 이하의 논의에서는 도 10도 그리고 도 7a의 ALU도 참조한다.

[0048] 제1 명령어(1000)(예를 들어, 명령어(D))는 실행 유닛(113) 중 하나에 의한 연산의 결과의 수신지 레지스터(예를 들어, R_d)를 표현할 수 있다. 앞서 논의된 바와 같이, ALU Out(701)은 실행 유닛(113)의 결과를 저장할 수신지 레지스터(R_d)의 주소를 발생시킬 수 있다. 그리하여, ALU Out(701)은 레지스터(R_d)를 발생시키기 위한 제1 명령어(1000)와 연관될 수 있다.

[0049] 제2 명령어(1001)(예를 들어, 명령어(C))는 실행 유닛(113)에 의해 표현되는 if-then-else 문의 조건을 표현할 수 있다. 예시된 실시형태에 있어서, 조건은 비교값(V_C)에 의해 표현된다. 앞서 논의된 바와 같이, Comp ALU(702)은 멀티플렉싱 기능(706)을 위한 선택 신호로서 사용되는 조건을 발생시킬 수 있다. 그리하여, Comp ALU(702)은 V_C가 참인지의 비교를 위한 제2 명령어(1001)와 연관될 수 있다.

[0050] 제3 명령어(1002)(예를 들어, 명령어(T))는 실행 유닛(113)에 의해 표현되는 if-then-else 문의 "then" 결과를 표현할 수 있다. 예시된 실시형태에 있어서, "then" 결과는 V_t - 참인 경우의 값에 의해 표현된다. 앞서 논의된 바와 같이, ALU1(703)은 "then" 결과를 발생시킬 수 있다. 그리하여, ALU1(703)은 V_t인 "then" 결과를 위한 제3 명령어(1002)와 연관될 수 있다.

[0051] 제4 명령어(1003)(예를 들어, 명령어(F))는 실행 유닛(113)에 의해 표현되는 if-then-else 문의 "else" 결과를 표현할 수 있다. 예시된 실시형태에 있어서, "else" 결과는 V_f - 거짓인 경우의 값에 의해 표현된다. 앞서 논의된 바와 같이, ALU2(704)은 "else" 결과를 발생시킬 수 있다. 그리하여, ALU2(704)은 V_f인 "else" 결과를 위한 제4 명령어(1003)와 연관될 수 있다.

[0052] V_C인 조건, V_t인 "then" 결과, V_f인 "else" 결과, 및 R_d인 결과 레지스터를 사용하면, if-then-else 문은 다음에

의해 표현될 수 있다:

$$\text{if } (V_C == \text{TRUE})$$

$$\text{then}$$

$$\text{Reg}[R_d] := V_t$$

$$\text{else}$$

$$\text{Reg}[R_d] := V_f$$

[0053]

[0054] 도 11은 도 1의 자율 메모리 프로세싱 장치(130)를 편입할 수 있는 메모리 시스템의 일 실시형태의 블록 선도를 예시하고 있다. 메모리 시스템은 하나 이상의 메모리 디바이스(예를 들어, SSD)(1101, 1102)와 네트워크(1120)를 통하여 통신할 수 있는 컨트롤러(1100)(예를 들어, CPU)를 포함할 수 있다. 네트워크(1120)는 유선 버스 또는 무선 통신(예를 들어, 와이파이)일 수 있다.

[0055] 메모리 디바이스(1101)는 메모리 디바이스(1101)의 저장부를 구성하는 로컬 메모리(100)(예를 들어, RAM, DRAM, SRAM, NAND 플래시, NOR 플래시, 상 변화 메모리(PCM))는 물론 도 1의 자율 메모리 프로세싱 장치(130)도 포함할 수 있다. 자율 메모리 프로세싱 장치(130)는 메모리(100)에 비교적 가까이 위치하고 있을 수 있다(예를 들어, 동일 다이, 동일 다이 스택, 동일 메모리 모듈). 예를 들어, 자율 메모리 프로세싱 장치(130)는 메모리(100)의 뱅크 레벨에서 회로에 포함될 수 있다. 각각의 뱅크는 하나의 메모리 칩이 실질적으로 동시에 동작하는 자율 메모리 프로세싱 장치(130)의 다수의 인스턴스를 가질 수 있도록 서로 다른 자율 메모리 프로세싱 장치(130)를 가질 수 있다. 여기에서 사용되는 바와 같이, 로컬 메모리(100)는 네트워크를 통하여 가지 않고 자율 메모리 프로세싱 장치(130)에 접속되는 메모리일 수 있다.

[0056] 도 11의 시스템의 디바이스의 각각은 노드라고 생각될 수 있다. 각각의 노드는 네트워크(1120)를 통하여 다른 노드와 통신할 수 있다. 노드의 각각은 실질적으로 유사할 수 있거나 또는 노드 중 하나 이상은 서로 다른 아키텍처를 가질 수 있다. 예를 들어, 제1 메모리 디바이스(1101)는 프로그램 카운터 실행 유닛(114)에 부가하여 단일 실행 유닛(113)만을 가질 수 있는 한편 제2 메모리 디바이스(1102)는 프로그램 카운터 실행 유닛(114)에 부가하여 하나보다 많은 실행 유닛(113)을 가질 수 있다.

[0057] 그리하여, 후속하여 설명되는 바와 같이, 컨트롤러(1100)(예를 들어, 소스 노드)는 소스 노드의 현재 프로세싱 상태 및 명령어를 포함하고 있는 메시지(예를 들어, 패킷)를 메모리 디바이스(1101)(예를 들어, 표적 노드)에 발신할 수 있다. 다른 일 실시형태에 있어서, 제1 메모리 디바이스(1101)는 소스 노드일 수 있는 한편 제2 메모리 디바이스(1102)는 표적 노드일 수 있다.

[0058] 명령어는 메모리 디바이스(1101)로의 커맨드(예를 들어, 검색, 정렬, 비교)를 포함할 수 있다. 메모리 디바이스(1101)는 컨트롤러에 의한 개입 없이 커맨드에 의해 명령된 태스크를 수행할 수 있다. 자율 메모리 프로세싱 장치(130)는 메시지를 다른 노드(1100, 1102)에 발신 및 그로부터 수신하고, 프로세싱 명령어 및 상태를 다른 노드(1100, 1102)에 발신 및 그로부터 수신하고, 프로그램 상태를 복원 및 저장하고, 프로세싱 명령어를 실행하고, 로컬 메모리를 판독 및 기록하고, 그리고/또는 단일 노드에서 다수의 프로세싱 컨텍스트를 지원할 수 있다.

[0059] 자율 메모리 프로세싱 장치(130) 아키텍처는 (예를 들어, ALU을 포함하는) 실행 유닛(113)을 부가 및 제거하는 동적 무결절 융통성을 제공하여, 그리하여 필요에 따라 부가적 프로세싱 능력을 노드에 부여할 수 있다. 자율 메모리 프로세싱 장치(130)에서의 실행 유닛(113)의 동적 부가 및 제거는 이하의 예의 연산에서 예시될 수 있다.

[0060] 전형적 종래 기술 프로그램은 다음과 같이 발생될 수 있다:

$$\text{Instruction1 (ADD Register1, Register2, Register3)}$$

[0061] $$\text{Instruction2 (SUB Register2, Register3, Register4)}$$

[0062] 전형적 종래 기술 CPU 시스템에서와 같이, 이들 명령어에는 내포된 종속성이 있다. 예를 들어, Register2에서의

값은 Instruction1이 실행할 기회를 가지기 전에 겹쳐쓰여질 것이기 때문에 Instruction2는 Instruction1 전에 (또는 그와 동일한 사이클에서) 실행할 수 있지 않을 수 있다.

[0063] 자율 메모리 프로세싱 장치 아키텍처에 있어서, 더 복잡한 실행 유닛(EU) 아키텍처는 프로그램을 실행하는데 필요한 사이클의 수를 감축하기 위해 사용될 수 있다. 각각의 EU는 각각 별개의 태스크를 수행하는 소정 수의 다른 ALU(예를 들어, 4개의 ALU)을 포함하고 있을 수 있다. 그리하여, 자율 메모리 프로세싱 장치에 대해 기록되는 프로그램은 다음과 같이 발생될 수 있다(PCEU 더하기 하나의 EU를 갖는 아키텍처를 가정):

[PCEU Instruction1] [EU1 Instruction1]

[0064] [PCEU Instruction2] [EU1 Instruction2]

[0065] 각각의 [EU# Instruction#]는, 도 10에 예시된 바와 같이, 다음과 같이 나타날 수 있다:

[0066] [Destination Instruction] [Comparison Instruction] [If-true Instruction] [If-false Instruction]

[0067] 또한, 자율 메모리 프로세싱 장치 아키텍처의 일부분으로서, 프로세서는 그것들 내에 매립된 서로 다른 수의 EU를 가질 수 있다. 이것은, 예를 들면, 4개의 EU와 하나의 PCEU를 갖는 아키텍처를 가능하게 할 수 있다:

[PCEU Instruction1] [EU1 Instruction1] [EU2 Instruction1] [EU3 Instruction1] [EU4 Instruction1]

[0068] [PCEU Instruction2] [EU1 Instruction2] [EU2 Instruction2] [EU3 Instruction2] [EU4 Instruction2

[0069] 이들 EU의 명령어 중 어느 하나라도 이 사이클에서 수행할 부가적 작업이 있지 않을 수 있다는 사실에 기인하여 공백일 수 있다. 이것은 프로그램의 특정 스테이지에서의 병렬도 결여에 기인할 수 있다.

[0070] 자율 메모리 프로세싱 장치 아키텍처는 시스템에서 자율 메모리 프로세싱 장치 엔진의 이종 세트 간 상호작용을 가능하게 할 수 있다(예를 들어, 하나의 장치 "A"는 PCEU 더하기 하나의 EU를 가질 수 있는 한편, 다른 장치 "B"는, 동일한 상호접속된 시스템에서, PCEU 더하기 4개의 EU를 가질 수 있다). 이러한 시나리오에서 장치 A가 그 컨텍스트를 장치 "B"에 발신할 필요가 있다고 가정되면, 프로그램은 명령어의 순차적 스트림으로 패키징되어 장치 "B"에 배송될 수 있다. 장치 "B"는 그 후 다음과 같이 그 하드웨어 상에서 동일한 방식으로 그것들을 스케줄링할 수 있다:

[PCEU Instruction1] [EU1 Instruction1] [EMPTY] [EMPTY] [EMPTY]

[PCEU Instruction2] [EU1 Instruction2] [EMPTY] [EMPTY] [EMPTY]

[0071] …

[0072] 이것은 병렬도 손실에 이르러 시스템에서의 비효율을 초래할 수 있는데 모든 프로그램마다 결국에는 가장 좁은 자율 메모리 프로세싱 장치의 그것에 다가갈 것이기 때문이다.

[0073] 명령어는 명령어들 간 어느 종속성도 있지 않음을 보장하지 않고는 병렬 EU로 번들링되지 않을 수 있다. 이러한 종류의 비교는 전형적 종래 기술 시스템에서는 컴퓨터 계산상 비용이 많이 들 수 있으므로, 자율 메모리 프로세싱 장치는 명령어 "펜스" 플래그의 개념을 사용할 수 있다. "펜스" 플래그는 명령어 스트림이 그 스트림에서 이전 명령어에 대한 어느 종속성도 더 이상 갖지 않는 곳을 애플리케이션 라이터 또는 컴파일러가 마킹 가능하게 한다. 이러한 정보는 명령어 스트림이 돌려져 상당한 프로세싱 오버헤드 없이 프로세서의 이종 세트 상에서 스케줄링 가능하게 할 수 있다.

[0074] 예를 들어, 이하의 명령어 스트림: [PCEU Instruction] [EU Instruction1] [EU Instruction2] [EU Instruction3] [Fence Marker/Instruction] [EU Instruction4] [EU Instruction5] [EU Instruction6] [EU Instruction7] [Fence Flag/Instruction]은 자율 메모리 프로세싱 장치 "A" 상에서 이하의 방식: [PCEU] [1] [PCEU] [2] [F] [PCEU] [3] [PCEU] [4] [PCEU] [5] [PCEU] [6] [F] [PCEU] [7]으로 스케줄링될 수 있고(여기서

[F]는 "펜스" 마커를 나타냄), 자율 메모리 프로세싱 장치 "B"에서는: [PCEU] [1] [2] [3] [X] [F] [PCEU] [4] [5] [6] [7]로서 스케줄링될 수 있다.

[0075] "펜스" 명령어는 주어진 자율 메모리 프로세싱 장치(예를 들어, "A" 또는 "B")의 명령어 메모리에 그것이 로딩되어 있는 동안 패킷-인 로직에 의해 프로세싱될 수 있다. "펜스" 플래그의 존재는 명령어 메모리에 저장될 수 있지만, 스케줄링의 컨텍스트 외에는 무의미할 수 있다. 그렇지만, 그것은 패킷-아웃 로직이 원래 스트림을 복원할 수 있도록 명령어 메모리에 플래그로서 저장된다.

[0076] 자율 메모리 프로세싱 장치의 연산의 일례(예를 들어, 메모리 검색)로서, 패킷은 네트워크(예를 들어, 메모리 네트워크)로부터 패킷 파서(101)에 의해 수신될 수 있다. 패킷 파서(101)는 패킷을 세그먼트로 파싱할 수 있다. 일부 세그먼트는 패킷이 이전 노드를 떠났을 때 이전 노드가 있었던 상태를 표현하는 레지스터 컨텐트를 그것들이 포함하고 있을 수 있다는 점에서 컨텍스트일 수 있다.

[0077] 패킷은 실행될 프로그램에 대한 명령어 메모리(105)에서의 시작 위치를 포함하고 있을 수 있다. 이러한 시작점은 프로그램 카운터(107)에 로딩될 수 있다. 또한, 패킷은 명령어 메모리(105)에 로딩될 명령어 세트 및 레지스터 파일(109)에 로딩될 수 있는 초기 조건 세트를 포함하고 있을 수 있다. 초기 조건은 이전 노드로부터 명령어에 의해 발신되는 변수일 수 있다. 초기 조건은 또한 현재 실행 중인 프로그램에 의한 사용을 위한 상수일 수 있다.

[0078] 프로그램 카운터(107)에서의 값은 어느 명령어가 실행되도록 명령어 메모리(105)로부터 판독되는지 결정한다. 프로그램 카운터(107)에서의 다음 값은 이전 값으로부터의 증분이거나 또는 프로그램 카운터 실행 유닛(114)에 의해 결정되는 바와 같은 계산된 값일 수 있다.

[0079] 명령어는 파서(115)의 구성을 설정할 수 있다. 파서(115)는, 명령어의 실행을 통해, 페이지 버퍼(117)로부터 변수를 제거하도록 그리고 궁극적으로는 메모리 판독 연산을 수행하도록 구성될 수 있다.

[0080] 메모리 판독 연산이 일어날 때, 변수는 실시간으로 페이지 버퍼(117) 컨텐트로부터 제거되어 실행 유닛(113)에 입력으로서 제시될 수 있다. 다른 잠재적 입력은, 프로그램 명령어에 의해 결정되는 바와 같이, 레지스터 파일로부터 판독될 수 있고, 프로세싱을 위해 실행 유닛(113)에 제시될 수 있다. 앞서 설명된 바와 같이, "펜스"는 수개의 연이은 명령어를 병렬로 실행할 수 있는 능력을 제공할 수 있다. 병렬로 실행될 수 없는 명령어는 미루어져 후속 사이클 동안 실행될 수 있다.

[0081] 실행 유닛(113)은 그들 입력 인수를 복수의 세트의 입력 인수로서 프로세싱할 수 있으며, 각각의 세트는 병렬로 프로세싱된다. 그리하여, 다수의 실행 유닛(113)은 후에 레지스터 파일에 다시 전송되게 되거나, 파서(115)에 전송되어 궁극적으로는 하나 이상의 메모리 기록 연산에 대한 데이터로서 페이지 버퍼(117)에 기록될 수 있는 출력 변수를 발생시킬 수 있거나, 또는 출력 변수는 어떤 특정 액션을 발생시키도록 레지스터 파일에 들어갈 수 있다. 액션은 패킷 발생기(111)에 의해 패킷을 발생시키는 것이거나 또는 새로운 메모리 판독 또는 메모리 기록 연산을 개시하는 것일 수 있다.

[0082] 페이지 버퍼(117) 컨텐트(예를 들어, 검색 커맨드의 결과)는 요청하는 노드에 네트워크를 통하여 송신될 패킷에 포함되도록 패킷 발생기(111)에 제시될 수 있다. 패킷은 태스크(예를 들어, 검색)가 완료되었고 결과가 패킷에 포함되어 있다고 나타내는 요청하는 노드로의 메시지를 포함할 수 있다.

[0083] 연산의 더 넓은 예로서, 네트워크는 자율 메모리 디바이스의 패브릭을 포함할 수 있으며, 각각은 적어도 하나의 자율 메모리 프로세싱 장치를 포함한다. 데이터 그룹이 메모리 디바이스의 패브릭에 걸쳐 저장될 수 있다. 특정 데이터 리스트를 찾도록 데이터 그룹 전체를 검색하는 것이 소망될 때, 검색 프로그램은 특정 데이터 리스트를 찾도록 그 디바이스를 검색하도록 하나의 자율 메모리 디바이스에 푸싱될 수 있다. 프로그램이 그 특정 자율 메모리 디바이스 내에 저장된 데이터가 검색되었고 리스트로부터의 데이터 전부가 존재하지 않는다고 결정할 때, 프로그램은 하나 이상의 패킷으로 번들링되어 다른 자율 메모리 디바이스에 전송될 수 있고 거기서 그 디바이스의 자율 메모리 프로세싱 장치는 검색을 계속할 수 있다. 프로그램의 이러한 번들링은 자율 메모리 디바이스의 패브릭 전체가 검색되거나 데이터 리스트가 완료될 때까지 계속될 수 있다. 일부 실시형태에 있어서, 특정 자율 메모리 디바이스에서 발견된 데이터도 전송될 프로그램과 패킷(들)으로 번들링될 수 있다.

[0084] 그러한 일 실시형태가 도 12의 순서도에 예시되어 있다. 예시된 방법은 자율 메모리 디바이스(1101)에서의 자율 메모리 프로세싱 장치(130)에 의해 도 11의 시스템에서 실행될 수 있다.

[0085] 메모리 디바이스(1101)는 자율 메모리 프로세싱 장치(130)에 제공되는 패킷을 수신할 수 있다(1201). 장치(13

0)는 앞서 논의된 바와 같이 명령어, 프로그램 카운터, 및 데이터를 제거하도록 패킷을 파싱할 수 있다(1203). 그 후 명령어는 메모리(100)에 저장된 데이터에 대해 소망 태스크를 수행하도록 실행될 수 있다(1205). 그 후, 명령어, 및 명령어에 의해 발생된 어느 데이터라도 네트워크 상의 송신(1209)을 위한 패킷으로 번들링될 수 있다(1207).

[0086] 장치는 회로, 집적 회로 다이, 메모리 디바이스, 메모리 어레이 또는 시스템으로서 정의될 수 있다.

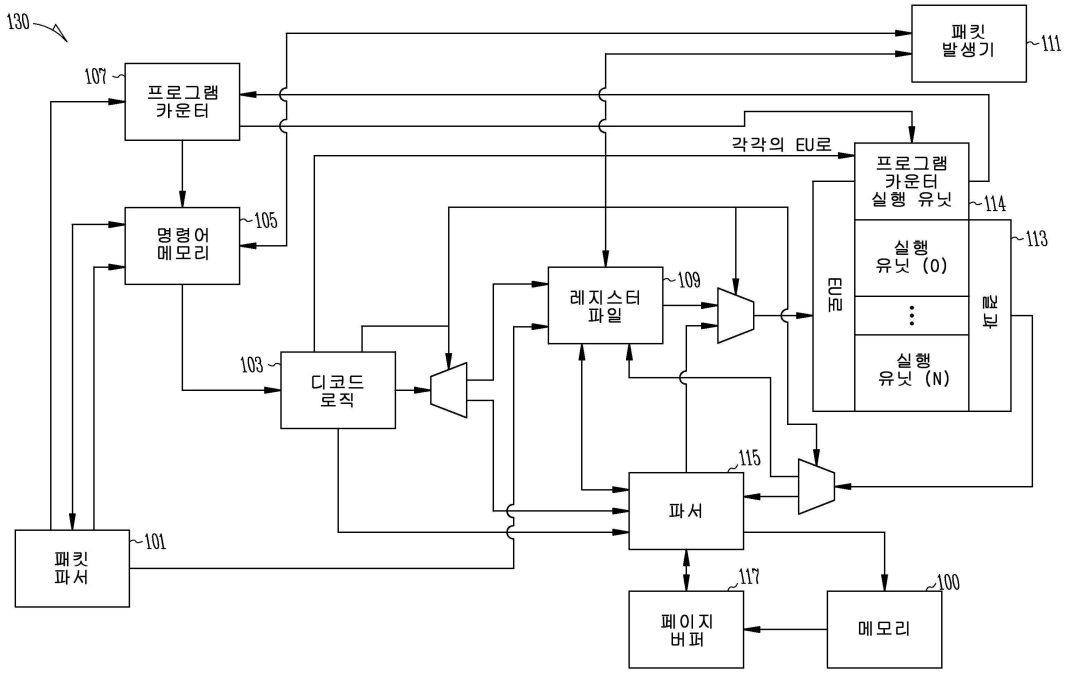
[0087] **결어**

[0088] 자율 메모리 디바이스 내에서의 자율 메모리 프로세싱 장치의 하나 이상의 실시형태는 전통적 CPU-기반 컴퓨팅 시스템의 메모리 대역폭 병목을 완화하도록 명령어의 프로세싱을 수행할 수 있다. 명령어 세트(예를 들어, 프로그램) 및/또는 데이터를 포함하고 있는 패킷은 노드에서의 메모리에서의 데이터가 CPU 또는 소스 노드로부터의 제어와 독립적인 명령어에 의해 연산될 수 있도록 그들 노드 간에 전송될 수 있다.

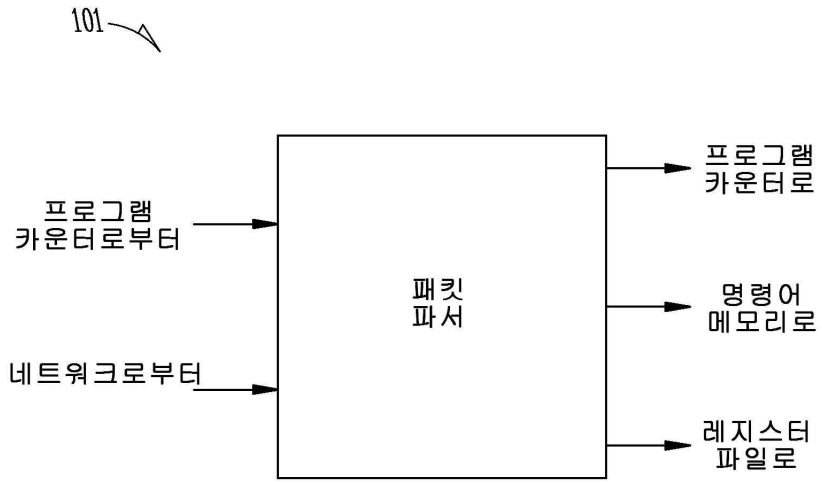
[0089] 여기에서는 특정 실시형태가 예시 및 설명되었지만, 나타낸 특정 실시형태에 대해 동일 목적을 달성하도록 계산되는 어느 배열이라도 대체될 수 있음을 당업자는 인식할 것이다. 당업자에게는 많은 적응이 명백할 것이다. 따라서, 본 출원은 어느 적응 또는 변형이라도 망라하도록 의도된다.

도면

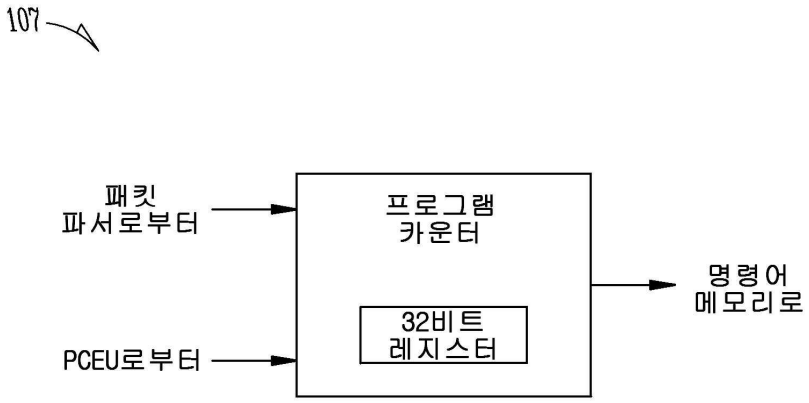
도면1



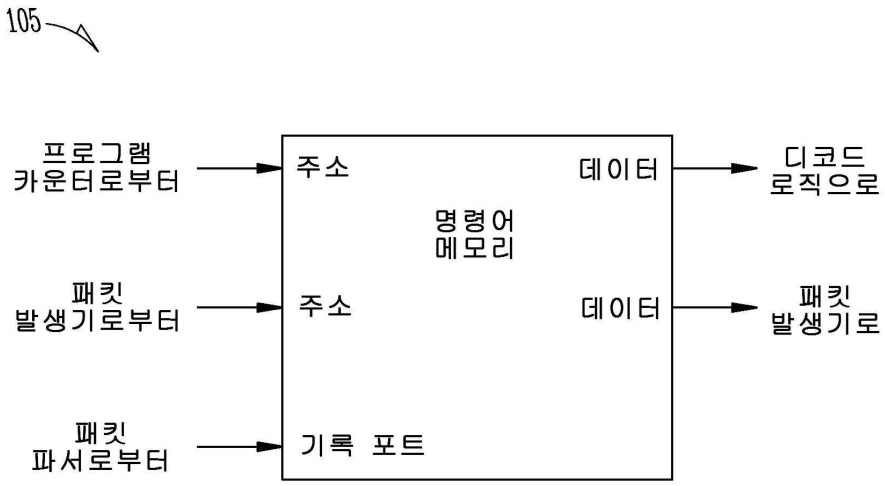
도면2


101
프로그램
카운터로부터
네트워크로부터
패킷
파서
프로그램
카운터로
명령어
메모리로
레지스터
파일로

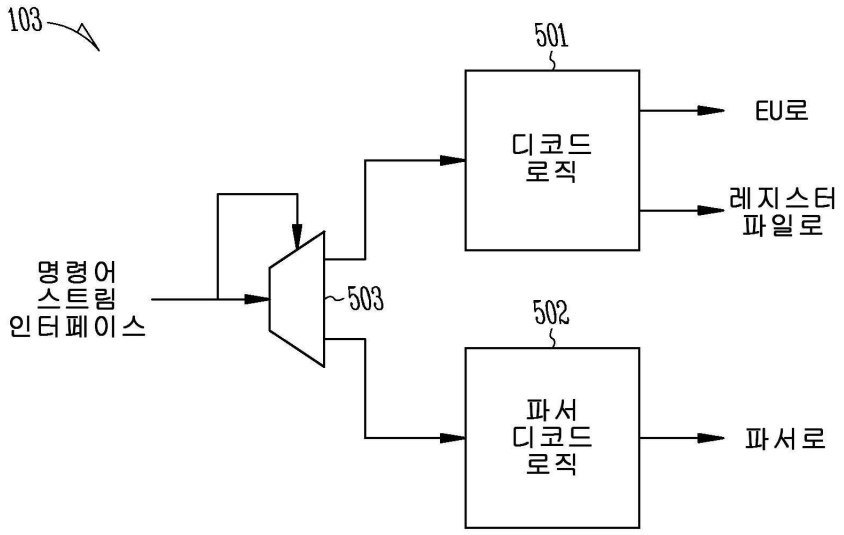
도면3


107
패킷
파서로부터
PCEU로부터
프로그램
카운터
32비트
레지스터
명령어
메모리로

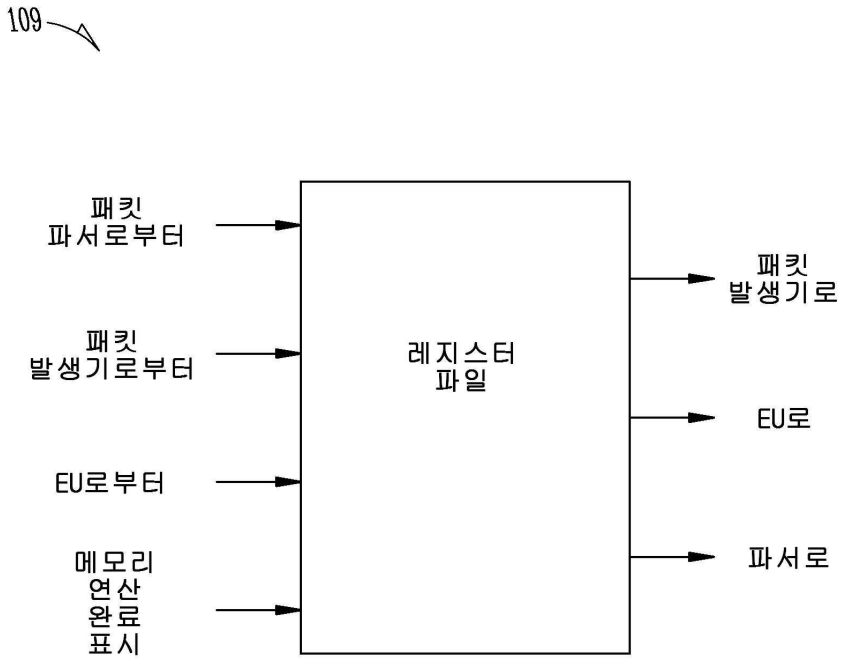
도면4


105
프로그램
카운터로부터
패킷
발생기로부터
패킷
파서로부터
주소
주소
기록 포트
명령어
메모리
데이터
데이터
디코드
로직으로
패킷
발생기로

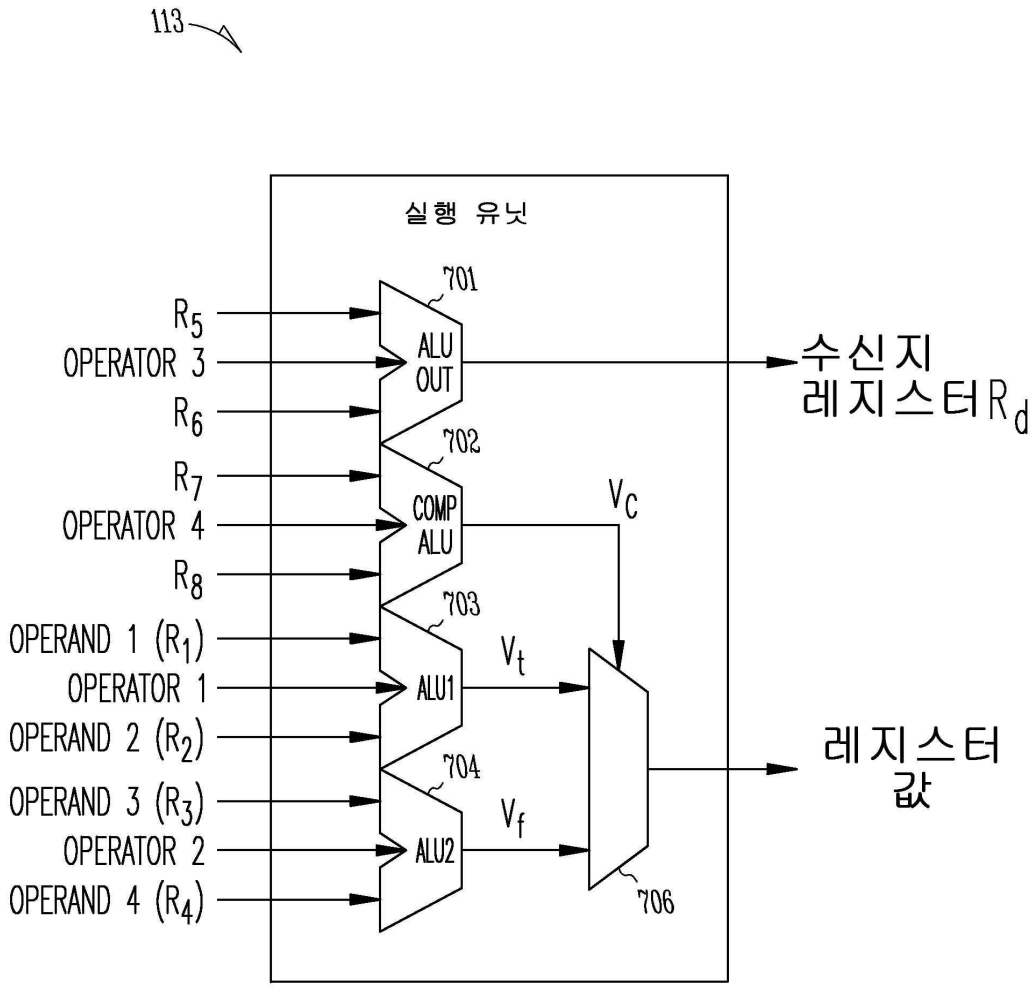
도면5


103
501
디코드
로직
EU로
레지스터
파일로
명령어
스트림
인터페이스
503
502
파서
디코드
로직
파서로

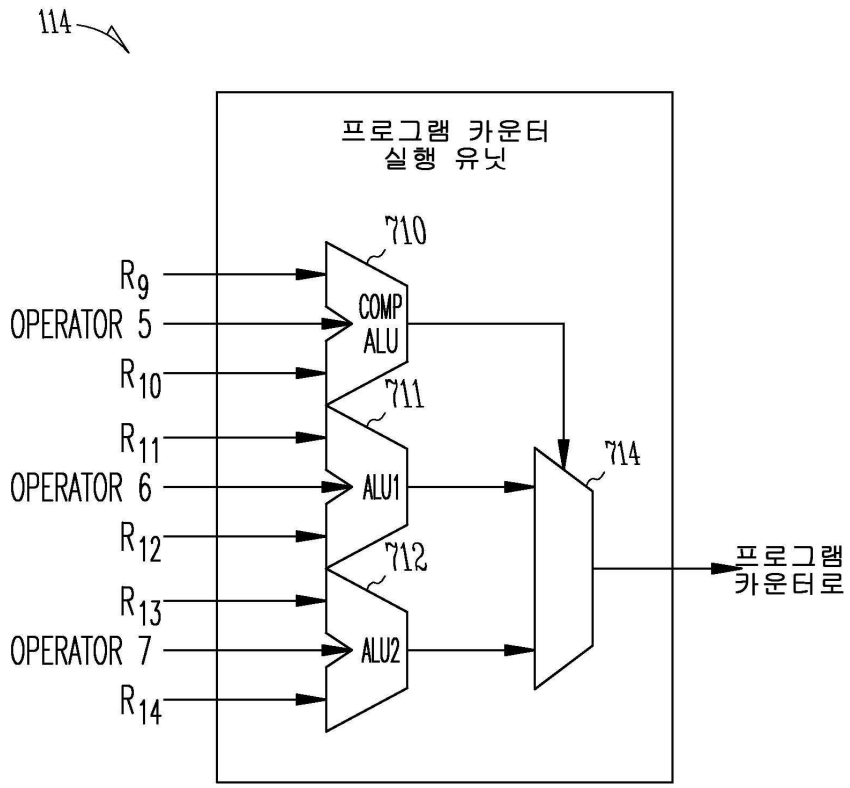
도면6


109
패킷
파서로부터
패킷
발생기로부터
EU로부터
메모리
연산
완료
표시
레지스터
파일
패킷
발생기로
EU로
파서로

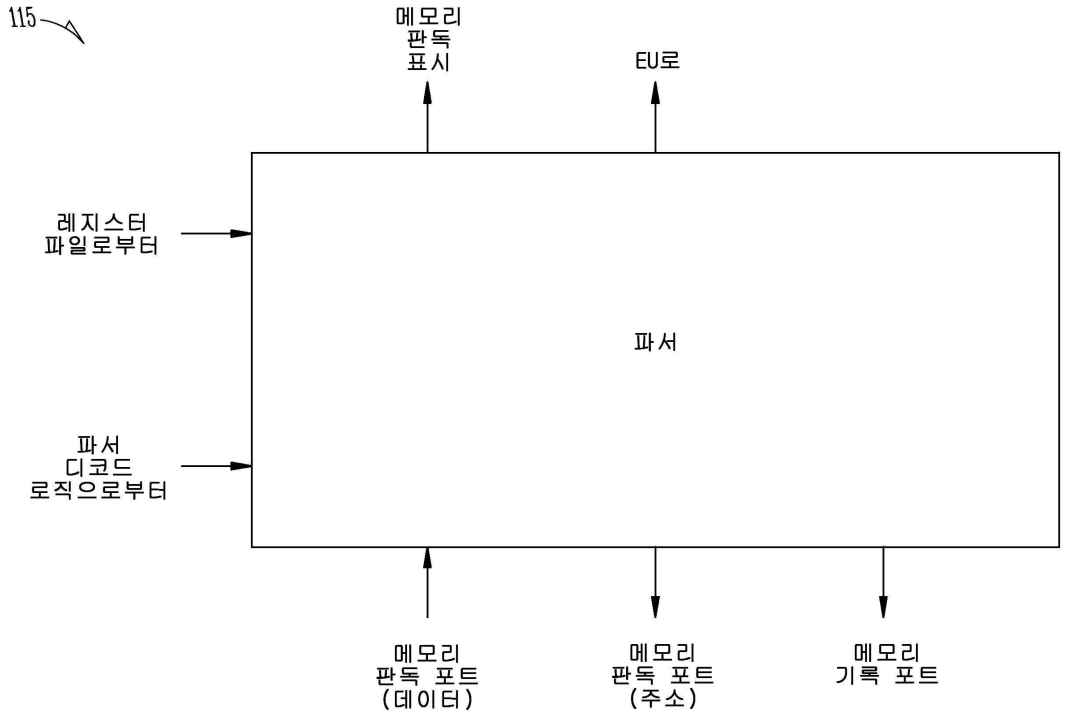
도면7a


113
실행 유닛
701
ALU OUT
702
COMP ALU
703
ALU1
704
ALU2
706
R5
OPERATOR 3
R6
R7
OPERATOR 4
R8
OPERAND 1 (R1)
OPERATOR 1
OPERAND 2 (R2)
OPERAND 3 (R3)
OPERATOR 2
OPERAND 4 (R4)
Vc
Vt
Vf
수신지 레지스터 Rd
레지스터 값

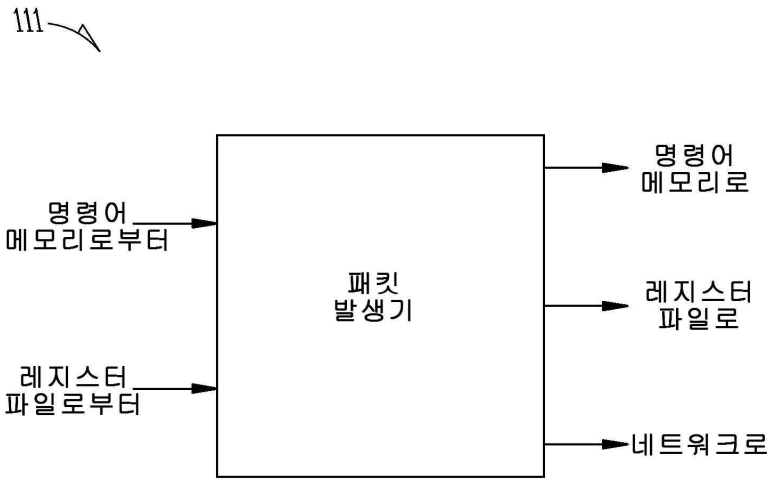
도면7b


114
프로그램 카운터
실행 유닛
710
COMP
ALU
711
ALU1
712
ALU2
714
R9
OPERATOR 5
R10
R11
OPERATOR 6
R12
R13
OPERATOR 7
R14
프로그램
카운터로

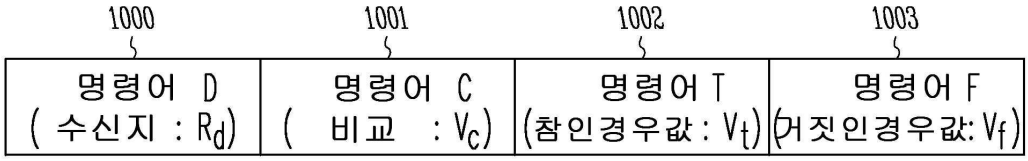
도면8


115
메모리
판독
표시
EU로
레지스터
파일로부터
파서
파서
디코드
로직으로부터
메모리
판독 포트
(데이터)
메모리
판독 포트
(주소)
메모리
기록 포트

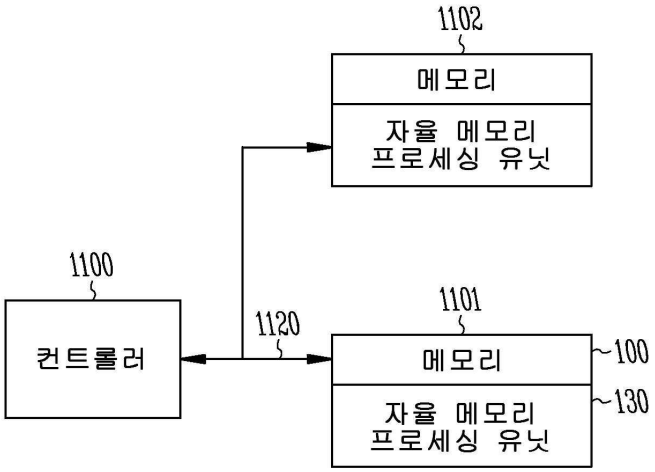
도면9



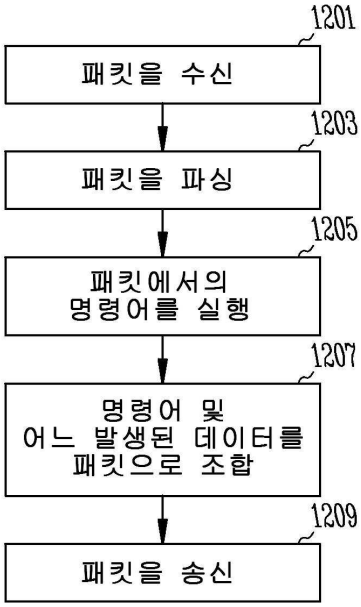
도면10



도면11



도면12


1201
패킷을 수신
1203
패킷을 파싱
1205
패킷에서의
명령어를 실행
1207
명령어 및
어느 발생된 데이터를
패킷으로 조합
1209
패킷을 송신