(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
H04L 12/28 (2006.01)

(45) 공고일자 2007년08월13일
(11) 등록번호 10-0748771
(24) 등록일자 2007년08월06일

(21) 출원번호 10-2002-7007422
(22) 출원일자 2002년06월10일
심사청구일자 2005년12월07일
번역문 제출일자 2002년06월10일
(86) 국제출원번호 PCT/CA2000/001442
국제출원일자 2000년12월08일
(65) 공개번호 10-2002-0081679
(43) 공개일자 2002년10월30일
(87) 국제공개번호 WO 2001/43370
국제공개일자 2001년06월14일

(81) 지정국 국내특허 : 아랍에미리트, 안티구와바부다, 알바니아, 아르메니아, 오스트리아, 오스트레일리아, 아제르바이잔, 보스니아 헤르체고비나, 바베이도스, 불가리아, 브라질, 벨라루스, 벨리제, 캐나다, 스위스, 중국, 코스타리카, 쿠바, 체코, 독일, 덴마크, 도미니카, 알제리, 에스토니아, 스페인, 핀란드, 영국, 그라나다, 그루지야, 가나, 감비아, 크로아티아, 헝가리, 인도네시아, 이스라엘, 인도, 아이슬랜드, 일본, 케냐, 키르키즈스탐, 북한, 대한민국, 카자흐스탄, 세인트루시아, 스리랑카, 리베이라, 레소토, 리투아니아, 룩셈부르크, 라트비아, 모로코, 몰도바, 마다가스카르, 마케도니아공화국, 몽고, 말라위, 멕시코, 뉴질랜드, 노르웨이, 폴란드, 포르투칼, 루마니아, 러시아, 수단, 스웨덴, 싱가포르, 슬로베니아, 슬로바키아, 시에라리온, 타지키스탄, 투르크맨, 터어키, 트리니아드토바고, 탄자니아, 우크라이나, 우간다, 미국, 우즈베키스탄, 베트남, 세르비아 앤 몬테네그로, 남아프리카, 짐바브웨, 모잠비크,

AP ARIPO특허 : 가나, 감비아, 케냐, 레소토, 말라위, 모잠비크, 수단, 시에라리온, 스와질랜드, 탄자니아, 우간다, 짐바브웨,

EA 유라시아특허 : 아르메니아, 아제르바이잔, 벨라루스, 키르키즈스탐, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크맨,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 사이프러스, 독일, 덴마크, 스페인, 핀란드, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 터어키,

OA OAPI특허 : 부르키나파소, 베닌, 중앙아프리카, 콩고, 코트디브와르, 카메룬, 가봉, 기니, 기니 비사우, 말리, 모리타니, 니제르, 세네갈, 차드, 토고,

(30) 우선권주장
60/170,232 1999년12월10일 미국(US)
60/212,966 2000년06월21일 미국(US)

(73) 특허권자
모사이드 테크놀로지스 코포레이션
미국, 94089-1116 캘리포니아, 써니베일, 올리언스 드라이브 1322

(72) 발명자
브라운,데이비드,에이.
캐나다,케이0에이1엘0,온타리오,카프,알리시아크레센트110

(74) 대리인
청운특허법인

(56) 선행기술조사문헌

공개번호 특1999-0078446

심사관 : 김대성

전체 청구항 수 : 총 48 항

(54) 최장의 정합 어드레스 탐색장치 및 방법

(57) 요약

본 발명은 검색 요청에 제공된 키에 대한 최장의 접두부 정합을 제공하기 위한 검색표에 관한 것이다. 검색 키는 검색표에 발행된 단일의 검색 요청에 제공된다. 상기 검색표는 키의 연속부분에 기초한 결과값에 대한 다단 검색을 실시한다.

대표도

도 2a

특허청구의 범위

청구항 1.

단일 검색 요청에서 제공된 키의 제1 부분에 의해 인덱스된 다수의 제1 메모리 로케이션을 포함하는 제1 레벨 매퍼; 및

상기 키의 제2 부분 및 상기 제1 메모리 로케이션의 인덱스된 하나에 저장된 제1 서브트리 선택에 따라 인덱스된 다수의 제2 메모리 로케이션을 포함하는 제2 레벨 매퍼를 포함하며, 검색표에 발행된 상기 단일 검색 요청은 상기 매퍼들중의 하나에 있는 로케이션에 저장된 상기 키에 대응하는 값을 나오게 하는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 2.

제1항에 있어서, 상기 제2 레벨 매퍼는

제1 서브트리 선택에 의해 인덱스된 서브트리 디스크립터를 저장하는 서브트리 메모리;

상기 제2 메모리 로케이션을 포함하는 서브트리 매퍼; 및

상기 선택된 서브트리 디스크립터 및 상기 키의 제2 부분에 따른 서브트리 매퍼에 대한 매퍼 인덱스를 제공하는 매퍼 어드레스 로직을 더 포함하는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 3.

제2항에 있어서, 상기 서브트리 디스크립터는

서브트리에 있는 각각의 가능한 노드에 대한 비트를 포함하는 비트맵; 및

상기 값을 저장하기 위해 각각 제2 메모리 로케이션의 블록의 베이스 어드레스를 저장하는 다수의 블록 포인터를 더 포함하는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 4.

제3항에 있어서, '0'으로 설정된 노드에 대응하는 비트맵에 있는 비트는 상기 노드가 선 노드와 값을 공유하는 것을 표시하는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 5.

제3항에 있어서, 상기 제2 메모리 로케이션에 대한 인덱스는 상기 노드에 대응하는 비트에 앞선 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 6.

제5항에 있어서, 상기 서브트리에 대한 매퍼에 저장된 값의 수는 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 7.

제3항에 있어서, 상기 블록은 16개의 제2 메모리 로케이션을 포함하는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 8.

제3항에 있어서, 상기 가능한 노드의 수는 256인 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 9.

제1항에 있어서, 상기 키의 제3 부분 및 상기 제2 메모리 로케이션의 하나에 저장된 제2 서브트리 선택에 따라 인덱스된 다수의 제3 메모리 로케이션을 포함하는 제3레벨 매퍼를 더 포함하는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 10.

제9항에 있어서, 상기 키의 제4 부분 및 상기 제3 메모리 로케이션의 하나에 저장된 제3 서브트리 선택에 따라 인덱스된 다수의 제4 메모리 로케이션을 포함하는 제4레벨 매퍼를 더 포함하는 것을 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 11.

제9항에 있어서, 상기 키는 32 비트이고, 상기 키의 제1 부분은 16 비트이며, 상기 키의 제2 부분은 8 비트이고, 상기 키의 제3 부분은 8 비트인 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 12.

제10항에 있어서, 상기 키는 40 비트이고, 상기 키의 제1 부분은 16 비트이며, 상기 키의 제2 부분은 8 비트이고, 상기 키의 제3 부분은 8 비트이며, 상기 키의 제4 부분은 8 비트인 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 13.

단일 검색 요청에서 제공된 키의 제1 부분으로 제1 레벨 매퍼에서 제1 로케이션을 선택하는 단계; 및

상기 인덱스된 제1 로케이션에 저장된 제1 서브트리 선택을 검지하여, 상기 키의 제2 부분 및 상기 제1 서브트리 선택에 따라 제2 레벨 매퍼에서 제2 로케이션을 선택하는 단계; 및

상기 키에 대한 단일 검색 요청에 응답으로 상기 매퍼들의 하나에 있는 로케이션에 저장된 상기 키에 대응하는 값을 보내는 단계를 포함하는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 14.

제13항에 있어서, 상기 제2 로케이션을 선택하는 단계는

서브트리 메모리에 있는 제1 서브트리 선택에 의해 인덱스된 서브트리 디스크립터를 선택하는 단계;

상기 선택된 서브트리 디스크립터 및 상기 키의 제2 부분에 따른 서브트리 매퍼에 대한 매퍼 인덱스를 결정하는 단계; 및

상기 매퍼 인덱스로 서브트리 매퍼에서 제2 메모리를 선택하는 단계를 더 포함하는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 15.

제14항에 있어서, 상기 결정하는 단계는

상기 서브트리 디스크립터에 저장되어 있으며 서브트리의 각각의 가능한 노드에 대한 비트를 포함하는 비트 맵에 근거하여 오프셋을 계산하는 단계; 및

상기 서브트리 디스크립터에 저장되어 있으며 각각 제2 메모리 로케이션의 블록의 베이스 어드레스를 저장하는 다수의 블록 포인터로부터 베이스 포인터를 선택하는 단계를 포함하는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 16.

제15항에 있어서, '0'으로 설정된 노드에 대응하는 비트맵에 있는 비트는 상기 노드가 선 노드와 값을 공유하는 것을 표시하는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 17.

제15항에 있어서, 상기 제2 메모리 로케이션에 대한 인덱스는 상기 노드에 대응하는 비트에 앞선 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 18.

제17항에 있어서, 상기 서브트리에 대한 매퍼에 저장된 값의 수는 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 19.

제15항에 있어서, 상기 블록은 16개의 제2 메모리 로케이션을 포함하는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 20.

제15항에 있어서, 상기 가능한 노드의 수는 256인 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 21.

제13항에 있어서,

상기 제2 로케이션에 저장된 제2 서브트리 선택을 검지하여, 상기 키의 제3 부분 및 제2 서브트리 선택에 따른 제3 레벨 매퍼에서 제3 로케이션을 선택하는 단계를 더 포함하는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 22.

제21항에 있어서,

상기 제3 로케이션에 저장된 제3 서브트리 선택을 검지하여, 상기 키의 제4 부분 및 제3 서브트리 선택에 따른 제4 레벨 매퍼에서 제4 로케이션을 선택하는 단계를 더 포함하는 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 23.

제21항에 있어서, 상기 키는 32 비트이고, 상기 키의 제1 부분은 16 비트이며, 상기 키의 제2 부분은 8 비트이고, 상기 키의 제3 부분은 8 비트인 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 24.

제22항에 있어서, 상기 키는 40 비트이고, 상기 키의 제1 부분은 16 비트이며, 상기 키의 제2 부분은 8 비트이고, 상기 키의 제3 부분은 8 비트이며, 상기 키의 제4 부분은 8 비트인 것을 특징으로 하는 키에 대한 최장의 접두부 정합 제공방법.

청구항 25.

단일 검색 요청에서 제공된 키의 제1 부분에 의해 인덱스된 다수의 제1 메모리 로케이션을 포함하는 제1 레벨 매퍼; 및

상기 키의 제2 부분 및 상기 제1 메모리 로케이션의 인덱스된 하나에 저장된 제1 서브트리 선택에 따라 인덱스된 다수의 제2 메모리 로케이션을 포함하는 제2 레벨 매퍼; 및

검색표에 발행된 상기 단일 검색 요청으로부터 상기 매퍼들의 하나에 있는 로케이션에 저장된 상기 키에 대응하는 값을 나오게 하는 수단을 포함하는 것을 특징으로 하는 최장의 접두부 정합 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 26.

제25항에 있어서, 상기 제2 레벨 매퍼는

제1 서브트리 선택에 의해 인덱스된 서브트리 디스크립터를 저장하는 서브트리 메모리;

상기 제2 메모리 로케이션을 포함하는 서브트리 매퍼; 및

상기 선택된 서브트리 디스크립터 및 상기 키의 제2 부분에 따른 서브트리 매퍼에 대한 매퍼 인덱스를 제공하는 수단을 더 포함하는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 27.

제26항에 있어서, 상기 서브트리 디스크립터는

서브트리에 대한 디폴트 값;

상기 서브트리에 있는 각각의 가능한 노드에 대한 비트를 포함하는 비트맵; 및

각각 제2 메모리 로케이션의 블록의 베이스 어드레스를 저장하는 다수의 블록 포인터를 더 포함하는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 28.

제27항에 있어서, '0'으로 설정된 노드에 대응하는 비트맵에 있는 비트는 상기 노드가 선 노드와 값을 공유하는 것을 표시하는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 29.

제27항에 있어서, 상기 제2 메모리 로케이션에 대한 인덱스는 상기 노드에 대응하는 비트에 앞선 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 30.

제29항에 있어서, 상기 서브트리에 대한 매퍼에 저장된 값의 수는 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 31.

제27항에 있어서, 상기 블록은 16개의 제2 메모리 로케이션을 포함하는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 32.

제27항에 있어서, 상기 가능한 노드의 수는 256인 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 33.

제25항에 있어서,

상기 키의 제3 부분 및 상기 제2 메모리 로케이션의 하나에 저장된 제2 서브트리 선택에 따라 인덱스된 다수의 제3 메모리 로케이션을 포함하는 제3레벨 매퍼를 더 포함하는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 34.

제33항에 있어서, 상기 키의 제4 부분 및 상기 제3 메모리 로케이션의 하나에 저장된 제3 서브트리 선택에 따라 인덱스된 다수의 제4 메모리 로케이션을 포함하는 제4레벨 매퍼를 더 포함하는 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 35.

제33항에 있어서, 상기 키는 32 비트이고, 상기 키의 제1 부분은 16 비트이며, 상기 키의 제2 부분은 8 비트이고, 상기 키의 제3 부분은 8 비트인 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 36.

제35항에 있어서, 상기 키는 40 비트이고, 상기 키의 제1 부분은 16 비트이며, 상기 키의 제2 부분은 8 비트이고, 상기 키의 제3 부분은 8 비트이며, 상기 키의 제4 부분은 8 비트인 것을 특징으로 하는 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 37.

단일 검색 요청에서 제공된 IP 어드레스의 제1 부분에 의해 인덱스된 다수의 제1 메모리 로케이션을 포함하는 제1 레벨 매퍼; 및

상기 IP 어드레스의 제2 부분 및 상기 제1 메모리 로케이션 중 인덱스된 하나에 저장된 제1 서브트리 선택에 따라 인덱스된 다수의 제2 메모리 로케이션을 포함하는 제2 레벨 매퍼를 포함하며, 검색표에 발행된 상기 단일 검색 요청은 상기 매퍼들의 하나에 있는 로케이션에 저장된 상기 IP 어드레스 대응하는 최장 프리픽스 매치를 나오게 하는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 38.

제37항에 있어서, 상기 제2 레벨 매퍼는

제1 서브트리 선택에 의해 인덱스된 서브트리 디스크립터를 저장하는 서브트리 메모리;

상기 제2 메모리 로케이션을 포함하는 서브트리 매퍼; 및

상기 선택된 서브트리 디스크립터 및 상기 키의 제2 부분에 따른 서브트리 매퍼에 대한 매퍼 인덱스를 제공하는 매퍼 어드레스 로직을 더 포함하는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 39.

제38항에 있어서, 상기 서브트리 디스크립터는

서브트리에 있는 각각의 가능한 노드에 대한 비트를 포함하는 비트맵; 및

상기 최장 접두부 정합을 저장하기 위해 각각 제2 메모리 로케이션의 블록의 베이스 어드레스를 저장하는 다수의 블록 포인터를 더 포함하는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 40.

제39항에 있어서, '0'으로 설정된 노드에 대응하는 비트맵에 있는 비트는 상기 노드가 선 노드와 값을 공유하는 것을 표시하는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 41.

제39항에 있어서, 상기 제2 메모리 로케이션에 대한 인덱스는 상기 노드에 대응하는 비트에 앞선 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 42.

제41항에 있어서, 상기 서브트리에 대한 매퍼에 저장된 값의 수는 비트맵에 저장된 '1'의 수에 따르는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 43.

제39항에 있어서, 상기 블록은 16개의 제2 메모리 로케이션을 포함하는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 44.

제39항에 있어서, 상기 가능한 노드의 수는 256인 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 45.

제37항에 있어서,

상기 IP 어드레스의 제3 부분 및 상기 제2 메모리 로케이션의 하나에 저장된 제2 서브트리 선택에 따라 인덱스된 다수의 제3 메모리 로케이션을 포함하는 제3레벨 매퍼를 더 포함하는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 46.

제45항에 있어서,

상기 IP 어드레스의 제4 부분 및 상기 제3 메모리 로케이션의 하나에 저장된 제3 서브트리 선택에 따라 인덱스된 다수의 제4 메모리 로케이션을 포함하는 제4레벨 매퍼를 더 포함하는 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 47.

제45항에 있어서, 상기 IP 어드레스는 32 비트이고, 상기 IP 어드레스의 제1 부분은 16 비트이며, 상기 IP 어드레스의 제2 부분은 8 비트이고, 상기 IP 어드레스의 제3 부분은 8 비트인 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

청구항 48.

제46항에 있어서, 상기 IP 어드레스는 40 비트이고, 상기 IP 어드레스의 제1 부분은 16 비트이며, 상기 IP 어드레스의 제2 부분은 8 비트이고, 상기 IP 어드레스의 제3 부분은 8 비트이며, 상기 IP 어드레스의 제4 부분은 8 비트인 것을 특징으로 하는 IP 어드레스에 대한 최장의 접두부 정합 제공 검색표를 저장한 컴퓨터로 읽을 수 있는 기록매체.

명세서

기술분야

인터넷은 라우터(routers)로 연결된 일련의 네트워크이다. 라우터는 각각의 예상 수신지 네트워크에 대해 수신된 데이타 패킷이 전송되어지는 차기 홉을 표시하는 경로표를 보유한다. 차기 홉은 다른 라우터 또는 최종 수신지일 수 있다.

라우터의 포트에서 수신된 인터넷 프로토콜("IP") 데이타 패킷은 IP 수신지 주소를 포함한다. 상기 IP 수신지 주소는 IP 데이타 패킷의 최종 수신지이다. 현재, 2개의 IP 버전, 즉 IP버전 4(IPv4)와 IP버전 6(IPv6)이 존재한다. IPv4는 IP 수신지 주소를 저장하기 위해 패킷된 데이타에 포함된 IP헤더에 32비트 필드를 제공한다. 라우터는 IP헤더에 저장된 IP 수신지 주소에 따라 수신지가 로칼 네트워크인 경우 최종 수신지 또는 차기 루프 라우터에 연결된 수신 데이타 패킷을 보낸다.

32비트 IPv4 수신지 주소는 40억개의 예상 루트를 제공한다. 인터넷 라우터는 통상적으로 40억개의 예상 루트중 50,000개를 저장한다. 그러나, 저장된 루트의 수는 IPv6의 확산과 인터넷의 성장에 따라 증가될 것이다.

본래, IP 어드레스 스페이스는 3가지 IP 어드레스 클라스, 즉 A클라스,B클라스,C클라스로 분류된다. 각각의 IP 어드레스 스페이스는 네트워크 어드레스와 호스트 어드레스로 분할된다. A클라스는 네트워크당 16백만개의 호스트와 126개의 네트워크를 고려한 것이다. B클라스는 네트워크당 64,000개의 호스트와 16382개의 네트워크를 고려한 것이며, C클라스는 네트워크당 256개의 호스트와 2백만개의 네트워크를 고려한 것이다. 그러나, IP 어드레스 스페이스를 서로다른 클라스로 분류하는 것은 가용 IP 어드레스의 수를 줄인다. C클래스만이 네트워크당 최대 256개의 호스트를 고려한 것으로 이는 대부분의 기관에서 너무 작다. 따라서, 대부분의 기관에는 B클라스 어드레스가 부여되며, 심지어 어드레스가 부여되지 않은 기관이 사용하지 않는 경우에도 다른 기관이 사용할 수 없는 64,000개의 호스트 어드레스를 차지한다. B클래스 IP 어드레스를 가진 기관의 호스트는 모두 16 최상위비트(MSB)로 이루어진 동일한 네트워크 어드레스를 저장한다, 예를 들어, 27.32.xx.xx.

배경기술

미사용 IP 호스트 어드레스를 자유롭게 사용하도록 하기 위한 클래스리스 인터도메인 라우팅(CIDR)이 소개되었다. 나머지 미사용 네트워크는 가변형 블록으로 기관에 배분되었다. 500개의 어드레스가 필요한 기관은 500개의 연속 어드레스를 얻었다. 예를 들어, 기관은 128.32.xx로 시작되는 500개의 가용 어드레스를 부여받을 수 있었다. 클래스리스 인터도메인 라우팅의 도입으로 라우터에 의해 저장되는 루트의 수는 증가되었다. 클래스리스 인터도메인 라우팅은 IP 수신지 주소에 대한 해당 차기 홉을 찾기 위해 정합 네트워크 어드레스를 탐색하는 대신 해당 루트를 찾기 위해 최장 접두부 대조가 필요하다. 상이한 차기 홉을 필요로 하는 또 다른 기관에 128.32.4.xx가 부여될 수 있기 때문에, 예를 들어, 128.xx.xx와 같은 B클래스 IP 어드레스의 16MSB 다음에서 검색이 더 이상 중단되지 않는다.

키에 대한 최장 접두부 정합을 검색하는 한가지 방법은 2진트리 검색을 사용하는 것이다. 2진트리 검색은 32단계까지 32비트 입력을 대조하며, 32비트 키와 일치하는 엔트리를 찾기 위해서는 32회의 검색이 필요하다. 정합을 검색하기 위한 다른 방법은 파트리시아 트리를 사용하는 것이다. 파트리시아 트리는 2진트리의 리프 아래에 엔트리가 존재하지 않는 경우 필요한 검색의 수를 줄인다.

IP 수신지 주소와 관련된 차기 홉을 효율적으로 검색하기 위한 또 다른 방법이 1998년 5월 11일자로 브로드닉 등에 의해 "신속한 라우팅 검색방법 및 시스템"이란 명칭으로 출원된 PCT 출원번호 제PCT/SE98/00854호에 개시되어 있다. 브로드닉의 방법은 이중 루트를 저장하지 않음으로써 저장된 차기 홉의 수를 줄인다. 차기 홉의 수를 줄임으로써, 메모리 조건이 저감되며, 이에 따라 루트 검색표가 고속 캐시 메모리에 저장될 수 있다.

브로드닉 등은 2진트리를 3단계로 분류한다. 2진트리를 3단계를 분류함으로써 검색수를 3개로 줄인다. 제 1 단계로 인덱스된 엔트리는 검색이 그 엔트리로부터 얻은 루트로써 제 1 단계에서 종료될 수 있는지의 여부 또는 검색이 IP 수신지 주소의 또 다른 부분을 사용하여 다음 단계로 계속되어야 하는지의 여부를 나타낸다.

도 1a는 2진트리의 제 1 단계를 나타내는 종래의 64K(65536)비트 맵을 도시한 도면이다. 64K 비트 맵(30)은 노드(44)당 1비트로서, 깊이(16)에서의 2진트리의 리브 또는 노드(44)를 나타낸다. 상기 비트 맵은 비트 마스크 길이(16)로 분리된다. 64K 비트 맵에는 2^{12}=4096개의 비트 마스크가 존재한다. 1비트 마스크가 도 1a에 도시되어 있다. 노드(44)에 해당하는 포인터 어레이에 저장된 서브트리 또는 루트 인덱스가 존재할 경우, 비트 맵(30)에서 비트는 "1"로 설정된다. 노드가 선행 노드(44)와 루트 엔트리를 공유하는 경우, 비트 맵(30)에서 비트는 "0"으로 설정된다.

도 1b는 캐시 메모리에서 구비된 종래의 검색표를 도시한 도면이다. 상기 검색표는 코드 워드 어레이(36), 베이스 인덱스 어레이(34) 및 맵 루트 업데이트블(40)을 포함한다. 또한, 32비트 IP 어드레스(38)가 도 1b에 도시되어 있다. 코드워드(46)는 비트 맵(30)의 각 비트 마스크에 대한 코드 워드 어레이(36)에 저장된다(도 1a 참조). 코드 워드(46)는 6비트 값(46a)과 10비트 오프셋(46b)을 포함한다. 베이스 인덱스(42)는 코드 워드 어레이(46)의 모든 4코드 워드(46)에 대해 베이스 인덱스 어레이(34)에 저장된다.

상기 코드 워드 어레이(36), 베이스 인덱스 어레이(34) 및 맵 루트 업데이트블(40)은 포인터 어레이(미도시)의 포인터를 선택하기 위해 사용된다. 상기 포인터는 다른 검색을 실시하기 위한 루트 인덱스 또는 인덱스를 저장한다.

코드 워드 어레이(36)에서 코드 워드(46)를 선택하고 베이스 인덱스 어레이(34)에서 베이스 인덱스(42)를 선택함으로써 포인터 어레이에서 포인터 군이 선택된다. 상기 코드 워드(46)는 IP 어드레스(38)의 첫번째 12비트(50)를 이용하여 선택된다. 상기 베이스 인덱스(42)는 IP 어드레스(38)의 첫번째 10비트(48)를 이용하여 선택된다. 상기 포인터 군의 선택 포인터는 맵 루트 업데이트블(33)을 이용하여 선택된다.

선택된 코드 워드(46)에서 10비트 값(46b)은 맵 루트 업데이트블(32)로의 인덱스이다. 상기 맵 루트 업데이트블(32)은 4비트 오프셋에 대한 비트 마스크내의 비트수를 맵핑한다. 상기 오프셋은 포인터 어레이의 선택된 포인터군에서 포인터를 특정한다. 10비트 값(46b)은 맵 루트 업데이트블(32)에서 열을 선택하며, IP 어드레스(52)의 비트(19:16)는 4비트 오프셋(54)을 선택한다.

따라서, 포인터 검색에는 하기된 캐시 메모리 억세스가 필요하다:

(1) 16비트 코드 워드(46) 읽기; (2) 16비트 베이스 어드레스(42) 읽기; (3) 맵 루트 업데이트블(32)로부터 4비트 오프셋(54) 읽기; (4) 포이터 인덱스에서 포인터 읽기, 여기서 상기 포인터 인덱스는 베이스 어드레스(42), 코드 워드 오프셋(46a) 및 4비트 오프셋(54)의 합이다.

상기 2진트리의 각 단계에 대하여 동일한 메모리 억세스가 필요하다. 따라서, 3단계 검색은 12 메모리 억세스가 필요하다.

발명의 상세한 설명

대리인 관리번호 제2037.2004-001호의 데이비드 에이. 브라운의 "최장의 정합 어드레스 탐색장치 및 방법"은 키에 해당하는 값을 다단계 검색하기 위한 검색표를 개시하고 있다. 검색표에 저장될 수 있는 루트 인덱스의 수는 검색표의 메모리 로케이션의 수에 의해 제한된다.

본 발명의 원리에 따르면, 제 1 검색 유닛과 제 2 검색 유닛을 포함하는 검색표가 제공된다. 상기 제 1 검색 유닛은 키를 수신하고, 제 1 검색 유닛에서의 다단계 검색을 통하여 제 1 결과를 제공한다. 제 2 검색 유닛은 상기 키를 수신하고, 제 2 검색 유닛에서의 다단계 검색을 통하여 제 2 결과를 제공한다. 상기 제 1 검색 유닛과 제 2 검색 유닛은 모두 키를 병렬로 수신하고, 단지 하나의 유닛만이 상기 제 1 결과 및 제 2 결과에 따라 검색 유닛중 단지 하나에 저장된 키에 대한 최종 결과를 제공한다.

상기 최종 결과는 검색표에 저장될 최종 결과의 총수에 따라 루트 업데이트블들에 배분된다. 상기 키는 IP 어드레스일 수 있으며, 최종 결과는 그 IP 어드레스에 대한 루트 인덱스이다.

상기 키에 의해 인덱스된 제 1 검색 유닛에서의 제 1 메모리 로케이션은 최종 결과를 저장하고, 상기 키에 의해 인덱스된 제 2 검색 유닛에서의 제 2 메모리 로케이션은 최종 결과가 제 1 검색 유닛에 저장되었다는 표식을 저장한다. 상기 제 1 검색 유닛과 제 2 검색 유닛은 각각 다수개의 매퍼 레벨을 포함하며, 각각의 매퍼 레벨은 결과를 저장하기 위한 메모리 로케이션을 포함한다.

일실시예에서, 제 1 검색 유닛에서 다수개의 매퍼 레벨중 하나의 로케이션에 저장된 키에 해당하는 최종 결과는 제 2 검색 유닛에 전달되며, 그 검색 유닛에 의해 제공된다.

선택적 실시예에서, 키에 해당하는 최종 결과는 그 최종 결과가 저장된 검색 유닛에 의해 제공된다.

실시예

이하, 본 발명의 바람직한 실시예를 설명한다.

밀집 모드

도 2a는 본 발명의 원리에 따른 최장의 정합 접두부 검색표(100)를 도시한 도면이다. 상기 검색표(100)는 키(104)에 대한 루트 인덱스(102)를 제공한다. 루트 인덱스(102)는 IP 수신지 주소의 차기 홉에 억세스하기 위해 사용된다. 도 2a에 도시된 실시예에서, 키(104)는 32비트 폭을 갖지만, 키(104)는 32비트에 한정되지 않는다. 검색표(100)는 3개의 매퍼(106a 내지 106c)를 포함한다. 각각의 매퍼(106a 내지 106c)는 별도의 어드레스를 가진 메모리를 포함한다. 상기 매퍼(106a 내지 106c)중 하나의 로케이션에는 루트 인덱스(102) 또는 키(104)에 해당하는 디폴트 루트 인덱스가 저장된다. 다수개의 매퍼에 대한 검색이 필요한 경우, 모든 매퍼(106a 내지 106c)가 키에 대해 억세스될 때까지, 각각의 매퍼에서 나온 매퍼 출력(110a 내지 110c)은 지연 메모리(150a 내지 150c)에 저장된다.

멀티플렉서(108)는 루트 인덱스(102)로서 당해 멀티플렉서(108)의 입력에 전달되는 매퍼 출력(110a 내지 110c)중 하나를 선택한다. 상기 매퍼 출력(110a 내지 110c)은 당해 매퍼 출력(110a 내지 110c)의 최상위비트(MSB)에 따라 선택된다. 상기 매퍼 출력(110a 내지 110c)이 루트 인덱스(102)를 포함하는 경우, 매퍼 출력(110a 내지 110c)의 MSB는 "1"뿐이다.

도 2b는 도 2a에 도시된 검색표(100)에서 매퍼(106a 내지 106c)에 저장된 엔트리의 2진트리 표현을 나타낸 도면이다. 도 2b를 도 2a와 함께 설명한다. 32비트 키(104)는 32단계 2진트리로 표현될 수 있다. 2진트리 구조는 32단계까지 각각의 비트를 검색하기 위해 32개의 검색을 필요로 한다. 검색의 수를 줄이기 위해, 2진트리의 32 단계는 3개의 매퍼 레벨로 분

할되며, 각각의 매퍼 레벨(112a 내지 112c)은 매퍼(106a 내지 106c)(도 2a 참조)에 해당한다. 매퍼 레벨 1(112a)은 2진 트리의 32 단계중 첫번째 16 단계를 포함한다. 그러나, 단순화를 위하여 16 단계중 5 단계만 도 2b에 도시되어 있다. 매퍼 레벨 2(112b)는 32 단계 2진트리중 다음의 8 단계를 포함하며, 도 2b에는 8 단계중 3 단계가 도시되어 있다. 매퍼 레벨 3은 32 단계 2진트리중 마지막 8 단계를 포함하며, 도 1b에는 8 단계중 3 단계가 도시되어 있다. 각각의 매퍼 레벨(112a 내지 112c)은 다수개의 노드를 포함한다. 매퍼 레벨 1(112a)에 16 단계(키(104)의 16MSB)가 존재하고, 매퍼 레벨 2(112b)에 8 단계가 존재하며, 매퍼 레벨 3에 8 단계가 존재하도록 32 단계를 분할하는 것이 현재 메모리 기술에서 최적인 것으로 나타났으나, 본 발명은 이러한 구성에 한정되지 않는다.

상기 키(104)의 첫번째 16 비트에 대한 16개의 비트별 검색을 실시하는 대신, 키(104)의 첫번째 16 비트와 관련된 루트 인덱스(102)가 매퍼(106a)(도 2a)에 저장된다. 상기 매퍼(106a)(도 2a)는 키(104)의 첫번째 16비트 MSB로 직접 인덱스된다. 선행 매퍼(106a)가 키(104)와 관련된 차기 홉 정보를 억세스하기 위해 사용된 루트 인덱스(102)를 저장하는지의 여부에 따라 차기 매퍼(106b)가 검색된다.

도 2b에 도시된 바와 같이, 매퍼 레벨 1(112a)에 도시된 노드 또는 리브는 각각 r0와 r1로 표시된 2개의 루트(114)(116)와, 각각 s0와 s1로 표시된 매퍼 레벨 2(112b)에 대한 2개의 포인터(130^4)(130^{23})를 포함한다. 각각의 루트(114)(116)에 대한 루트 인덱스(102)가 L1 매퍼(106a)에 저장된다. 또한, L2 매퍼(106b)에 대한 어드레스 포인터(120)가 서브트리 인덱스(130^4)를 위해 저장되며, L2 매퍼(106b)에 대한 어드레스 포인터(미도시)가 서브트리 인덱스(130^{23})를 위해 저장된다. 매퍼(106a)의 매퍼 엔트리(140^4)에서 서브트리 인덱스(130^4)를 위해 저장된 어드레스 포인터(120)는 키(104)와 관련된 루트 인덱스(102)를 찾기 위해 차기 레벨에 대한 검색이 필요하다는 것을 나타낸다.

상기 트리에서 임의의 노드 값은 루트(114)로부터 경로를 추적함으로써 결정될 수 있다. 2진트리에서 각각의 노드는 2개의 자식, 즉 우측 자식과 좌측 자식으로 표시된다. 우측 자식은 부모 노드가 "1"인 경우 선택된다. 좌측 자식은 부모 노드가 "0"인 경우 선택된다. 루트(114)로부터 노드(116)까지의 경로를 추적하면, r1은 '010'로 설정된 MSB를 가진 모든 키에 대한 L1 매퍼(160a)의 루트 인덱스(102)로서 저장된다. 루트(114)로부터 s0 노드(130^4)까지의 경로를 추적하면, s0는 '00011'로 설정된 MSB를 가진 모든 키에 대한 L1 매퍼(160a)에 저장된다.

상기 L1 매퍼(160a)는 직접 맵된 매퍼가며, 각각의 하부 레벨 노드 또는 리프에 대한 루트 인덱스(102)를 매퍼 레벨 1(112a)의 하부 레벨에 저장한다. 상기 매퍼 레벨 1(112a)의 하부 레벨은 32 레벨 2진트리의 16번째 레벨이다. 상기 16번째 레벨은 64K 노드를 갖는다. 그러나, 설명을 위하여, 매퍼 레벨 1(112a)의 하부 레벨은 32 레벨 2진트리의 레벨 5로서 도시되어 있다. 상기 L1 매퍼(160a)에 도시된 루트 인덱스(102)는 매퍼 레벨 1(112a)의 레벨 5(130^1 내지 130^{32})에 해당한다. 루트 노드(114)로부터 레벨 5 노드(130^1)(130^2)(130^3)까지의 경로를 추적하면, 루트 인덱스(102)는 r0이다. 따라서, r0는 L1 매퍼(160a)의 로케이션(140^1)(140^2)(140^3), 즉 인덱스(00000)(00001)(00010)에 저장된다. 노드(130^4)는 서브트리 인덱스(s0)를 저장하며, 따라서, s0은 L1 매퍼(160a)의 로케이션(140^4)의 어드레스(00011)에 저장된다. 이와 유사하게, 레벨 5 노드(130^5 내지 130^8)에 대한 루트 인덱스(102)는 r0이며, 따라서, r0은 L1 매퍼(160a)의 로케이션(140^5)(140^6)(140^7)(140^8)의 어드레스(00100)(00101)(00110)(00111)에 저장된다. 레벨 5 노드(130^9 내지 130^{12})에 대한 루트 인덱스(102)는 r1이며, 따라서, r1은 L1 매퍼(160a)의 로케이션(140^9)(140^{10})의 어드레스(01000)(010001)에 저장된다.

상기 L1 매퍼(160a)의 각 로케이션은 레벨 5 노드(300^1 내지 300^{32})에 부여된 루트 인덱스(102)를 차기 매퍼(106b 내지 106c)에 직접, 또는 레벨 5 노드(300^{1-32})의 부모, 또는 어드레스 포인터를 통해 저장한다. 매퍼 레벨 3(106c)은 32 단계 2진트리의 하부 레벨에서 2개의 호스트 노드, 즉 노드(138)의 ho와 노드(140)의 h1을 포함한다. 호스트 노드의 검색에는 키(104)의 모든 비트에 대한 검색이 요구된다. 도 2a에 도시된 바와 같이, h0의 루트 인덱스(102)가 L3 매퍼(106c)의 로케이션(146^4)에 저장된다. L1 매퍼(106a)와는 다르게, L2 매퍼(106b)와 L3 매퍼(106c)는 직접 맵되지 않는다.

매퍼(106b)(106c)에서, 루트 인덱스(102)는 각각의 예상 입력에 저장되지 않는다. 루트 인덱스(102)는 노드의 루트 인덱스(102)가 매퍼(106b 및 106c)에 기저장된 루트 인덱스(102)와 다른 경우에만 저장된다. 매퍼 레벨 2(112b)에 도시된 제 1 서브트리(A)의 레벨 3 노드를 참조하면, 노드(132^1)(132^2)의 루트 인덱스(102)는 r0이며, 따라서, r0에 대한 루트 인덱스가 L2 매퍼(106b)의 양 노드(132^1)(132^2)의 로케이션(142^1)에 저장된다. 노드(132^2)의 서브트리 인덱스(s0)는 로케이션(142^3)에 저장된다. 레벨 3 노드(132^4)(132^5)(132^6)관련된 루트 인덱스(102)는 r0이며, 이는 선행 노드(132^2)에 대해

저장된 s0와 다르기 때문에, r0는 L2 매퍼(106b)의 차기 로케이션(142^3)에 저장된다. 노드(132^7)는 선행 로드(132^6)와 동일한 루트를 공유하지 않기 때문에, 루트(r2)는 노드(132^7)에 대한 L2 매퍼(106b)의 로케이션(142^4)에 저장된다. 서브트리 인덱스(r3)가 차기 레벨 3 노드(132^7)를 위해 저장되며, 따라서, s3는 L2 매퍼(106b)의 로케이션(142^5)에 저장된다. 선행 노드로부터 루트 변화가 존재하는 경우에만 루트 인덱스를 저장함으로써, 루트 인덱스(102)를 저장하는데 필요한 메모리는 줄어든다. 도시된 바와 같이, 매퍼 레벨 2(112b)의 제 1 서브트리(A)에 8개의 레벨 3 노드(132^{1-8})의 루트 인덱스를 저장하기 위하여 L2 매퍼(106b)에는 단지 5개의 로케이션만 필요하다. 간접 매퍼(106b)(106c)는 도 5와 함께 후에 상세하게 설명한다.

도 3은 본 발명의 원리에 따른 40비트 키(210)에 대한 최장의 정합 접두부 검색표(200)를 도시한 도면이다. 일실시예에서, 40비트 키는 8비트 접두부와 32비트 IP 어드레스를 포함한다. 상기 8비트 접두부은 32비트 IP 어드레스와 관련된 가상 개인 네트워크(VPN) 식별자일 수 있다. 상기 검색표(200)는 4개의 매퍼(106a 내지 106d)를 포함한다. 매퍼(106a)는 도 2a를 참조하여 설명한 바와 같은 직접 맵된 매퍼가다. 매퍼(106b 내지 106d)는 간접 매퍼가다. 매퍼(106a)는 40비트 키(210)의 16MSB에 해당하는 L2 매퍼(106b)의 서브트리 인덱스 또는 루트 인덱스(102)를 저장한다. 따라서, 상기 L1 매퍼는 64K 에상 로케이션을 갖고, 64K 노드 각각에 대한 매퍼가 제 1 매퍼 레벨(112a)(도 2b 참조)에 존재한다. L1 매퍼(106a)의 해당 로케이션에 저장된 L1 매퍼 엔트리 데이타(220a)는 파이프라인(208)과 L2 간접 매퍼(106b)에 전달된다. 상기 L1 매퍼 엔트리 데이타(220a)가 키(210b)의 다음 8비트를 이용한 차기 레벨의 검색이 필요하다고 지적할 경우, 상기 키(210b)의 다음 8비트와 L1 매퍼 엔트리 데이타(220a)에 따라 L2 간접 매퍼(106b)에서 검색이 실시된다.

제 2 레벨 검색 결과는 L2 매퍼 엔트리 데이타(220b)에 제공되며, 이는 파이프라인(208)과 L3 간접 매퍼(106c)에 전달된다. 제 3 레벨 검색은 키(210c)의 다음 8비트와 L2 매퍼 엔트리 데이타(220b)에 따라 L3 간접 매퍼(106c)에서 실시된다.

검색 결과는 L3 매퍼 엔트리 데이타(220c)에 제공되며, 이는 파이프라인(208)과 L4 간접 매퍼(106d)에 전달된다. L3 매퍼 엔트리 데이타(220c)는 키(210d)의 다음 8비트와 L3 매퍼 엔트리 데이타(220c)에 따라 L4 간접 매퍼(106d)에서 제 4 검색을 실시할지 여부를 결정한다.

제 4 검색 결과는 L4 매퍼 엔트리 데이타(220d)에 제공된다. 키(210)에 대한 최장의 정합 접두부와 관련된 루트 인덱스(102)는 매퍼(106a 내지 106d)중 하나의 매퍼에서 단지 하나의 로케이션에만 저장된다. 따라서, 파이프라인(208)에 전달된 루트 인덱스(102)는 단지 하나의 매퍼 엔트리 데이타(220a 내지 220d)에만 포함된다. 상기 루트 인덱스(102)가 매퍼(106a 내지 106d)중 하나, 예를 들어, 매퍼(106b)에서 발견되면, 나머지 매퍼(106c 및 106d)에 대한 검색은 불필요하며, 매퍼(106c 및 106d)는 악세스되지 않는다. 파이프라인(208)은 매퍼 엔트리 데이타(220a 내지 220d)중 하나에 포함된 루트 인덱스(102)를 선택하기 위한 멀티플렉서(108)(도 2a 참조)를 포함한다. 예를 들어, 매퍼 엔트리 데이타(220a 내지 220d)의 MSB는 루트 인덱스가 포함되었는지의 여부에 대한 표식을 제공할 수 있다.

매퍼(106a 내지 106d)와 함께 파이프라인(208)을 이용함으로써, 상이한 키(210)를 가진 최장의 정합 접두부 루트 업데이트블(200)에 대한 다중 검색이 병렬로 실시될 수 있다. 파이프라인(208)은 필요하다면 40비트 키(210)에 해당하는 루트 인덱스를 찾기 위하여 다른 각각의 매퍼(106a 내지 106d)의 검색이 완료될 때까지 40비트 키(210)와 관련된 각각의 매퍼(106a 내지 106d)에 대한 매퍼 엔트리 데이타(220a 내지 220d)를 저장함으로써 40비트 검색표(200)의 다중 검색이 병렬로 실시될 수 있도록 한다. 직접 맵된 매퍼(106a)에 대한 단일의 메모리 악세스를 실시함으로써, 수신된 IP 어드레스에 해당하는 루트 인덱스에 대한 검색 요청이 검색표(200)에 제공된다. 다른 키에 해당하는 루트 인덱스에 대한 다음 검색이 직접 맵된 매퍼(106a)를 위한 차기 메모리 악세스 싸이클의 검색표(200)에 제공될 수 있다.

도 4는 도 3에 도시된 직접 맵된 매퍼(106a)에 저장될 수 있는 매퍼 엔트리의 형식을 나타낸 도면이다. 도 2b에 도시된 2진트리에서 임의의 노드에 대한 매퍼 엔트리는 노 엔트리(300), 루트 엔트리(302) 또는 서브트리 엔트리 디스크립터(304)를 저장할 수 있다. 각각의 매퍼 엔트리(300)(302)(304)는 서브트리 플렉(306)을 포함한다. 서브트리 플렉(306)의 상태는 매퍼 엔트리가 서브트리 엔트리 디스크립터(304)인지의 여부를 나타낸다. 서브트리 플렉(306)이 '1'로 설정된 경우, 매퍼 엔트리는 서브트리 엔트리 디스크립터(304)이며, 서브트리 인덱스(312)를 포함한다. 서브트리 인덱스(312)는 다음의 간접 맵된 매퍼(106b 내지 106d)에 저장된 서브트리 엔트리 디스크립터(304)의 어드레스이다. 상기 서브트리 엔트리는 도 4를 참조하여 후에 상세하게 설명한다. 서브트리 플렉이 '0'이면, 메모리 엔트리가 노 엔트리(300) 또는 루트 엔트리(302)인지의 여부를 결정하기 위해 노 엔트리 플렉(314)가 체크된다. 상기 노 엔트리 플렉(314)이 '1'이면, 상기 엔트리는 루트 엔트리(302)이며, 루트 인덱스 필드(310)의 키(104)와 관련된 루트 인덱스(102)(도 3 참조)를 저장한다. 상기 서브트리 플렉(306)은 루트 인덱스(102)(도 3 참조)를 포함하는 매퍼 엔트리 데이타(220a 내지 220d)를 선택하기 위해 멀티플렉서(108)(도 2a 참조)에 의해 사용된다.

도 5는 도 2b에 도시된 매퍼 레벨 2(112b)의 노드에 해당하는 매퍼(106b)를 도시한 도면이다. 상기 매퍼(106b)는 서브트리 메모리(400), 매퍼 어드레스 로직(402) 및 서브트리 매퍼(418)를 포함한다. 매퍼(106a)에 저장된 키(210a)의 첫번째 부분에 의해 선택된 서브트리 인덱스(312)는 서브트리 메모리(400)에 전달된다. 상기 서브트리 메모리(400)는 서브트리 인덱스(312)에 의해 선택된 서브트리 엔트리(404)를 포함한다. 상기 서브트리 엔트리(404)는 데이타 필드(406)와 포인터 필드(408)를 포함한다.

도 2b를 다시 참조하면, 서브트리 엔트리(404)는 매퍼 레벨 2(112b)에 도시된 서브트리중 하나의 하부 레벨에 해당한다. 매퍼 레벨 2(112b)가 8개의 레벨을 갖는 경우, 각각의 서브트리(미도시)의 하부 레벨은 각각의 로드에 대해 하나씩 최대 256개의 루트를 갖는다.

도 5를 참조하면, 서브트리 엔트리(404)는 당해 서브트리의 하부 레벨의 각 노드에 해당하는 256개의 예상 루트 인덱스(102)(도 3 참조)에 대한 악세스를 제공한다. 루트 인덱스(102)(도 3 참조)는 서브트리 매퍼(418)에 저장된다. 256개의 예상 루트 인덱스에 대한 악세스를 제공하기 위하여, 밀집 서브트리 디스크립터가 데이타 필드(406)에 저장된다. 상기 데이타 필드(406)의 폭은 256비트이며, 서브트리의 하부 레벨의 각 노드에 1비트를 제공한다. 상기 데이타 필드(406)는 도 6a 및 도 6b를 참조하여 후에 상세하게 설명한다. 포인터 필드(408)은 폭이 256비트이고 16개의 16비트 포이터의 저장을 허용하며, 각각의 포인터는 256개의 루트 인덱스에 대한 악세스를 제공하기 위해 서브트리 매퍼(418)에 16개의 연속된 매퍼 엔트리의 베이스 어드레스를 저장한다. 따라서, 상기 포인터 필드(408)는 서브트리 하부 레벨의 각 노드에 대하여 서브트리 매퍼(418)의 매퍼 엔트리에 포인터를 간접적으로 제공할 수 있다. 상기 포인터 필드(408)는 도 6을 참조하여 후에 상세하게 설명한다.

데이타 필드(406)의 밀집 서브트리 디스크립터에 저장된 서브트리 데이타(412)와 포인터 필드(408)에 저장된 서브트리 포인터(414)는 매퍼 어드레스 로직(402)에 전달된다. 또한, 상기 매퍼 어드레스 로직(402)은 키(210b)의 다음 부분(다음의 8비트)을 수신한다.

상기 매퍼 어드레스 로직(402)은 키(212b)의 다음 8비트, 서브트리 데이타(412) 및 그 서브트리와 관련된 서브트리 포인터(414)에 따라 서브트리 하부레벨의 노드와 관련된 매퍼 엔트리의 매퍼 어드레스(416)를 결정한다. 상기 매퍼 어드레스(416)는 서브트리 매퍼(418)에서 매퍼 엔트리를 선택한다. 서브트리 매퍼(418)는 직접 맵된 매퍼(106a)와 관련하여 도 4에서 설명한 것과 동일한 형식의 매퍼 엔트리를 포함한다. 매퍼 데이타 엔트리(220b)의 컨텐츠는 차후 검색이 필요한지의 여부를 결정한다. 상기 매퍼 엔트리 데이타(220b)가 다음 매퍼 레벨(112c)(도 2b 참조)에 다른 서브트리 엔트리(404)가 존재함을 나타내는 서브트리 인덱스(312)를 포함하는 경우, 다음의 검색이 필요하다.

상기 키(210b)의 두번째 부분은 선택된 서브트리의 하부 레벨에서 노드를 선택한다. 상기 서브트리 포인터(414)는 서브트리의 노드와 관련된 베이스 어드레스를 선택하고, 상기 서브트리 데이타(412)는 베이스 어드레스와 관련된 매퍼 엔트리의 블록 내에서 오프셋을 선택한다. 상기 매퍼 어드레스 로직(402)은 도 7을 참조하여 후에 상세하게 설명한다.

도 6a는 서브트리의 2진트리 표현이다. 도시된 서브트리는 5개의 레벨을 포함한다. 서브트리는 3개의 루트 인덱스(r1)(r2)(r3)와 2개의 서브트리 인덱스(s0)(s1)를 포함한다. 서브트리의 하부 레벨에는 32개의 노드(500^{1} 내지 500^{32})가 존재한다. 하부 레벨의 각 노드(500^{1} 내지 500^{32})와 관련된 루트 인덱스 또는 서브트리 인덱스가 하기된 표 1에 나타나 있다.

[표 1]

서브트리 비트	루트/서브트리
00000	r0
00001	r0
00010	r0
00011	r0
00100	r1
00101	r1
00110	r0
00111	r0
01000	r2

01001	s0
01010	r2
01011	r2
01100	r2
01110	r2
01111	r2
1xxxx	r3

도 6b는 도 6a에 도시된 서브트리의 하부 레벨의 노드에 해당하는 도 5에 도시된 데이타 필드(406)에 저장된 밀집 서브트리 디스크립터를 도시한 도면이다. 델타 필드(406)는 32비트를 포함하며, 도 6a에 도시된 서브트리의 하부 레벨에서 각 노드(500)에 대해 1비트이다. 데이타 필드(406)의 비트(502^1 내지 502^{32})는 다음과 같이 부여된다: 즉, 선행 노드의 루트 인덱스가 사용되면 데이타 필드(406)의 비트는 '0'으로 설정되며, 서브트리 매퍼(418)(도 5 참조)에 저장된 차기 루트 인덱스가 사용되면 차기 매퍼 엔트리 어드레스에 '1'을 증분하도록 설정된다. 데이타 필드(402)에서 첫번째 비트는 루트가 특정되지 않는다면 매퍼 엔트리(504^1)에 저장된 디폴트 루트(r0)를 선택한다. 따라서, 상기 비트(502^1)는 특정된 루트가 존재하지 않기 때문에 디폴트 루트를 선택하기 위해 '0'으로 설정된다. 상기 매퍼 엔트리(504^1)에 저장된 디폴트 루트(r0)는 다음의 3개 노드(500^2 내지 500^4)를 위해 선택되며, 따라서, 해당 비트(502^2 내지 502^4)는 502^1에 의해 사용된 선행 루트 인덱스를 사용하기 위해 데이타 필드(406)에서 '0'으로 설정된다. 노드(500^5)에서 루트의 변화가 존재한다.

매퍼 엔트리(504^2)에 저장된 노드(500^5)를 위해 사용된 루트(r1)는 노드(500^6)에 의해 공유된다. 따라서, 비트(502^5)는 '1'이며, 이는 서브트리 매퍼(418)(도 5 참조)에서 매퍼 엔트리(504^2)를 선택하기 위한 루트 변화를 나타낸다. 비트(502^6)는 '0'이며, 이는 502^5에 저장된 루트 인덱스가 이 노드를 위해 사용됨을 나타낸다. 노드(500^7)에는 어떠한 루트도 제공되지 않기 때문에, 루트 변화가 존재하며, 디폴트 루트(r0)를 저장하는 서브트리 매퍼(418)(도 5 참조)에서 매퍼 엔트리(504^3)이 필요한 비트(502^7)에는 '1'이 저장된다.

노드(500^8)는 선행 노드(500^7)와 동일한 루트를 공유하며, 새로운 매퍼 엔트리가 서브트리 매퍼(418)(도 5 참조)에는 필요하지 않다. 노드(500^8)에 해당하는 비트(502^8)는 '0'으로 설정된다. 노드(500^9)는 선행 노드(500^8)와는 다른 루트를 가지며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하다. 노드(500^9)에 해당하는 비트(502^9)는 '1'로 설정되며, r2를 저장하는 매퍼 엔트리(504^6)가 다음의 연속 메모리 로케이션의 서브트리 매퍼(418)(도 5 참조)에 추가된다.

노드(500^{10})는 선행 노드(500^9)와는 다른 루트를 가지며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 루트 엔트리가 필요하다. 노드(500^{10})에 해당하는 비트(502^{10})는 '1'로 설정되며, s0를 저장하는 매퍼 엔트리(504^5)가 다음의 연속 메모리 로케이션의 서브트리 매퍼(418)(도 5 참조)에 추가된다.

노드(500^{11})는 선행 노드(500^{10})와는 다른 루트를 가지며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하다. 노드(500^{11})에 해당하는 비트(502^{11})는 '1'로 설정되며, r2를 저장하는 매퍼 엔트리(504^6)가 다음의 연속 메모리 로케이션의 서브트리 매퍼(418)(도 5 참조)에 추가된다.

노드(500^{12})(500^{13})는 선행 노드(500^{11})와 동일한 루트를 공유하며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하지 않다. 노드(500^{12})에 해당하는 비트(502^{12})와 노드(500^{13})에 해당하는 비트(502^{13})는 데이타 필드(406)에서 '0'으로 설정된다.

노드(500^{14})는 선행 노드(500^{13})와는 다른 루트를 가지며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하다. 노드(500^{14})에 해당하는 데이타 필드(406)의 비트(502^{14})는 '1'로 설정되며, s1을 저장하는 매퍼 엔트리(504^7)가 서브트리 매퍼(418)(도 5 참조)에 추가된다. 노드(500^{15})는 선행 노드(500^{14})와는 다른 루트를 가지며, 서브트리 매퍼

(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하다. 노드(500^{15})에 해당하는 데이타 필드의 비트(502^{15})는 '1'로 설정되며, r2를 저장하는 매퍼 엔트리(504^{8})가 서브트리 매퍼(418)(도 5 참조)에 추가된다. 노드(500^{16})는 선행 노드(500^{15})와 동일한 루트를 공유하며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하지 않다.

노드(500^{17})는 선행 노드(500^{16})와는 다른 루트를 가지며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하다. 노드(500^{17})에 해당하는 데이타 필드(406)의 비트(502^{17})는 '1'로 설정되며, r3를 저장하는 매퍼 엔트리(504^{9})가 서브트리 매퍼(418)(도 5 참조)에 추가된다.

노드(500^{18} 내지 500^{32})는 모두 노드(500^{17})와 동일한 루트를 공유하며, 서브트리 매퍼(418)(도 5 참조)에는 새로운 매퍼 엔트리가 필요하지 않다. 해당 비트(502^{18} 내지 502^{32})는 '0'으로 설정된다. 따라서, 32 노드(500^{1} 내지 500^{32})에 대한 서브트리 엔트리(304)(도 4 참조) 또는 루트 엔트리(302)(도 4 참조)를 저장하기 위해서는 9개의 매퍼 엔트리(504^{1-9})가 필요하다.

데이타 필드(406)에 저장된 밀집 서브트리 디스크립터에 저장된 "1"의 수를 계산함으로써, 노드(500^{1} 내지 500^{32})에 해당하는 매퍼 엔트리(504^{1} 내지 504^{9})는 서브트리 매퍼(418)(도 5 참조)에 인덱스된다. 예를 들어, 노드(500^{28})에 해당하는 매퍼 엔트리(504^{1} 내지 504^{9})를 찾기 위하여, 데이타 필드(406)의 비트(502^{1} 내지 502^{28})에 저장된 '1'의 갯수가 카운트된다. '1'의 갯수가 8이면, 해당 매퍼 엔트리는 디폴트 루트로부터 8번째 로케이션, 즉 매퍼 엔트리(504^{9})에 존재한다.

루트 변화가 존재하는 경우에만 매퍼 엔트리를 저장함으로써, 서브트리 매퍼(418)(도 5 참조)에서 서브트리당 매퍼 엔트리(504^{1} 내지 504^{9})의 수를 줄인다.

도 7은 도 5에 도시된 포인터 필드(408)를 나타낸다. 포인터 필드(408)는 서브트리 매퍼(418)(도 5 참조)에서 16 연속 매퍼 엔트리 로케이션(504^{1} 내지 504^{16})(도 6b 참조)의 블록에 대한 베이스 어드레스를 저장하기 위한 블록 베이스 어드레스 필드(600^{1})(600^{2})를 포함한다. 메모리는 16 연속 매퍼 엔트리의 블록(602^{1})(602^{2})의 서브트리 매퍼(418)(도 5 참조)에 할당된다. 8 레벨 서브트리는 256개의 모든 루트를 저장하기 위해 16 블록(600^{1})(600^{2})이 필요한 최대 256개의 서로 다른 루트를 가질 수 있다. 필요한 블록(602)의 수는 서브트리에 대한 루트의 수에 좌우된다. 블록 베이스 어드레스(미도시)의 프리 리스트로부터 블록 베이스 어드레스(602^{1})(602^{2})를 제거함으로써, 블록(602)은 특정 서브트리에 할당된다. 메모리에 대해 어드레스 프리 리스트를 제공하는 방법은 당업계에 잘 알려져 있다.

16 매퍼 엔트리(504^{1-16})의 메모리 블록을 할당함으로써, 할당된 16 로케이션이 연속적이기 때문에, 서브트리 매퍼(418)(도 5)의 메모리는 관리가 더 용이하다.

도 8은 도 5에 도시된 매퍼 어드레스 로직(402)를 도시한 도면이다. 상기 매퍼 어드레스 로직(402)은 오프셋 로직(700), 포인터 로직(702) 및 추가 로직(704)을 포함한다. 상기 오프셋 로직(700)은 노드 선택 로직(706)과 1카운트 로직(708)을 포함한다. 상기 포인터 로직은 베이스 어드레스 선택 로직(710)을 포함한다.

상기 노드 선택 로직(706)은 키(210b)의 8비트에 해당하는 서브트리 데이타(412)에서 노드(500)(도 6b)를 선택한다. 해당 노드 수는 노드 선택(718)에서 1카운트 로직(708)으로 전달된다. 상기 1카운트 로직(708)은 선택된 노드(500)에 해당하는 비트까지 서브트리 데이타 필드(406)에 저장된 1의 수를 카운트한다. 1의 수는 블록 선택(712)에서 포인터 로직(702) 및 블록 오프셋(714) 그리고 추가 로직(704)으로 전달된다.

8비트 폭 카운트 필드가 필요한 256비트 서브트리 데이타 필드(406)에 저장된 최대 256개의 '1'이 존재할 수 있다. 상기 8비트 카운트 필드는 2개의 필드로 분할되며, 4MSB는 블록 선택(712)을 제공하며, 4최하위비트(LSB)는 블록 오프셋(714)을 제공한다.

예를 들어, 8비트 키(210b)가 '0100 0100'인 경우, 노드 수(68)를 선택하기 위해, 서브트리 데이타(412)의 첫번째 68비트에 저장된 27개의 '1'이 존재하며, 상기 카운트는 1C Hex(0001 1100), MSB(0001); 즉, 블록 선택(714), 선택 블록(602^1)(도 6 참조) 및 LSB(1100)이며; 즉, 베이스 블록 오프셋은 매퍼 엔트리(504^{11})(도 6 참조), 즉 블록(502^1)에서 12번째 엔트리를 선택한다.

상기 베이스 어드레스 선택 로직(710)은 오프셋 로직(700)으로부터 전달된 블록 선택(712)에 따라 서브트리 포인터(414)로부터 베이스 어드레스(716)를 선택한다. 상기 추가 로직(704)는 오프셋 로직(700)으로부터 전달된 블록 오프셋(714)을 베이스 어드레스(716)에 추가하며, 매퍼 어드레스(416)를 제공한다. 상기 매퍼 어드레스(416)는 매퍼(106b 내지 106d)에서 매퍼 엔트리(504)(도 6b 참조)의 인덱스이다.

도 9는 본 발명의 원리에 따른 검색표(200)에서 키(210)(도 3 참조)에 대한 최장의 정합 접두부를 검색하기 위한 단계를 도시한 흐름도이다.

단계 800에서, 키(210a)(도 3 참조)의 첫번째 부분이 인덱스로서 매퍼(106a)에 전달된다. 단계 802로서 프로세싱이 계속된다.

단계 802에서, 상기 키(210a)(도 3 참조)의 첫번째 부분에 의해 인덱스된 제 1 레벨 매퍼에서 매퍼 엔트리(504)(도 6b 참조)에 저장된 매퍼 엔트리 데이타(220a)(도 3 참조)는 다음 레벨에 대한 검색이 더 필요한지의 여부를 결정한다. 필요하다면, 단계 804로서 프로세싱이 계속된다. 필요하지 않다면, 제 1 레벨 매퍼에서 인덱스된 매퍼 엔트리(504)(도 6b 참조)의 루트 엔트리(302)(도 4 참조)는 키에 대한 해당 최장의 접두부 루트를 저장하고, 단계 808로서 프로세싱은 계속된다.

단계 804에서, 다음 레벨 매퍼(106b 내지 106d)가 검색된다. 다음 레벨 매퍼의 인덱스는 키(210b 내지 210d)의 다음 부분과 선행 레벨 매퍼에서 인덱스된 매퍼 엔트리(504)(도 6b)의 서브트리 엔트리 디스크립터(304)(도 4 참조)에 저장된 서브트리 인덱스(312)에 좌우된다.

단계 806에서, 다음 레벨 매퍼에서 인덱스된 매퍼 엔트리(504)(도 6b 참조)는 검색이 더 필요한지의 여부를 나타내는 서브트리 인덱스 또는 키에 대한 해당 최장의 접두부 루트 인덱스를 저장한다. 검색이 더 필요하다면, 단계 804로서 프로세싱은 계속된다. 필요하지 않다면, 단계 808로서 프로세싱은 계속된다.

단계 808에서, 매퍼(106a 내지 106d)중 하나에서 매퍼 엔트리(504)(도 6b)에 저장된 루트 인덱스(102)(도 3 참조)는 검색표(200)으로부터 루트 인덱스(102)(도 3 참조)로서 전달된다. 프로세싱은 완료된다.

깊이 확장

도 3에 도시된 검색표(200)에 저장될 수 있는 루트 인덱스(102)의 수는 서브트리 매퍼(418(도 5))에서의 유용한 매퍼 엔트리(504(도 6B))의 수에 의해 제한된다. 예를 들면, 각각의 서브트리 매퍼(418(도 5))가 128K 매퍼 엔트리를 포함하고 검색표에 2개의 서브트리 매퍼(418(도 5))가 존재하면, 최대의 256K 루트 인덱스(102(도 3))는 검색표(200)에 저장될 수 있다. 128K 매퍼 엔트리를 지닌 서브트리 매퍼(418(도 5))는 17-비트 인덱스를 요구한다. 512K 매퍼 엔트리를 지닌 서브트리 매퍼(418(도 5))는 19-비트 인덱스를 요구한다. 검색표(200)에서 2개의 512K 서브트리 매퍼(418(도 5))는 32-비트 IPv4 수신지 어드레스를 위해 가능한 4십억 루트 인덱스(102(도 3))에 대한 저장을 제공한다.

루트 인덱스(102(도 3))를 저장하기 위해 매퍼 엔트리(504(도 6B))의 수는 복수개의 검색표(200)를 제공함으로써 증가될 수 있다. 상기 복수개의 검색표은 검색표(200)의 하나에서 서브트리 매퍼(418(도 5))에서의 매퍼 엔트리(504(도 6B))에 저장된 검색키(210)에 상응하는 값에 대해 병렬로 검색된다.

도 10A는 깊이 확장에 대한 실시예를 도시한다. 2개의 검색표가 도시되며, 이는 마스터 검색표(200a)와 슬래브 검색표(200b)이다. 그러나, 검색표의 수는 도시된 2개로 제한되지 않으며, 하나 이상의 슬래브 검색표(200b)가 추가될 수 있다.

검색표(200a-b) 각각은 동일한 검색키(210)와 병렬로 검색된다. 상기 검색키(210)에 대응하는 루트 인덱스(102(도 3))는 검색표(200a-b)의 하나에서 서브트리 매퍼(418(도 5))에 저장되거나, 검색표(200a, 200b)의 모두에서 직접 매퍼(106a)에 저장된다. 최종 루트 인덱스(900)는 두개의 검색표(200a, 200b)와 병렬로 검색한 후 발견된다.

도 10B는 도 10A에 도시된 실시예에서 검색표(200a) 중 하나를 도시한다. 검색표(200a-b) 각각은 매퍼(106a-d) 및 파이프라인(208)을 포함하며, 상기 파이프라인은 도 3 및 드라이버 로직(902)과 관련하여 검색표(200)에서 이미 기술되었다. 검색표(200a)은 검색키에 대응하는 루트 인덱스를 위하여 매퍼(106a-d)에서 멀티레벨 검색을 수행한다. 각각의 레벨 검색의 결과는 매퍼 엔트리 데이타(220a-d) 상에서 파이프라인(208)으로 보내진다. 상기 파이프라인(208)은 검색 결과(904)를 드라이버 로직(902)으로 보낸다. 검색표(200a-b)의 각각에서의 매퍼(106a)에서 매퍼 엔트리(504(도 6B))는 서브트리 엔트리(304(도 4))를 저장하지만, 루트 엔트리(302(도 4))는 오직 마스터 검색표(200a)에서의 매퍼(106a)에 저장된다. 노-엔트리(300(도 4))는 루트 엔트리(302) 대신에 슬래브 검색표(200b)에서 매퍼(106a)에서의 매퍼 엔트리(504(도 6B))에 저장된다. 매퍼(106a)에서 루트 인덱스를 지닌 오직 하나의 검색표를 실장하는 것은 최종 루트 인덱스(900)를 제공하도록 검색표를 선택하지 않도록 한다. 이는 도 3과 관련하여 기술된 바와 같이, 루트 인덱스를 저장하도록 사용될 수 없지만 마스터 검색표 또는 슬래브 검색표로써 구성되도록 동일한 검색표를 허용하는 슬래브 검색표(200b)에서 64K 메모리로 나타난다. 선택적인 실시예로써, 매퍼(106a)없는 슬래브 룩업 장치가 제공될 수 있다. 루트 검색(102(도 3))이 매퍼(106a)에서 루트 엔트리(302(도 4))에 저장되면, 검색은 마스터 검색표(200a)에서의 매퍼(106a)에서 종결된다.

도 10A에 도시된 바와 같이, 마스터 검색표(200a) 및 슬래브 검색표(200b)은 최종 루트 인덱스(900)를 공유한다. 최종 루트 인덱스(900)에 저장된 검색표(200a, 200b)은 루트 인덱스(102(도 3))을 제공한다. 각각의 검색표(200a, 200b)이 분리된 장치라면, 최종 루트 인덱스(900)를 공유하는 것은 각 장치의 외부 핀 합계를 줄여준다. 검색표(200a, 200b) 중 하나만이 어떤 시간에라도 최종 루트 인덱스(900)를 구동하게 된다.

두개의 검색표가 동일한 시간에 최종 루트 인덱스(900)를 구동하는 두개의 검색표(200a, 200b)에 검색키(210)와 관련된 루트 인덱스가 저장되는 에러 상태를 피하기 위하여, 각 검색표(200a, 200b)는 장치 코드(906)를 저장한다. 3-비트 장치 코드는 8개의 장치를 포함하도록 확장된 검색표를 허용한다.

상기 드라이버 로직(902)은 검색 결과(904)가 루트 인덱스(102(도 3))을 포함하는지를 결정한다. 포함한다면, 검색표(200a)에서 드라이버 로직(902)은 버스 요구 신호(미도시) 상에 최종 루트 인덱스(900)를 구동하겠다는 신호를 보낸다. 2개 이상의 검색표(200a, 200b)가 동시에 루트 인덱스 신호를 구동하겠다는 신호를 보내면, 상기 루트 인덱스는 가장 낮은 장치 코드와 함께 검색표(200a, 200b)에 의해 제공된다. 버스 요구 신호의 사용을 통해 버스 콘플릭트(bus conflicts)를 결정하는 방법은 본 기술분야에 잘알려져 있다.

도 10C는 검색키(210)에 대응하는 값을 저장하기 위해 가능한 매퍼 엔트리의 수를 증가시키도록 깊이 확장을 제공하는 또 다른 실시예를 도시한다. 도 10C에 도시된 실시예에서, 2개의 검색표(200c-d)는 값, 마스터 검색표(200c) 및 슬래브 검색표(200d)를 저장하도록 제공된다. 그러나, 검색표의 수는 도시된 2개로 제한되지 않으며, 매퍼 엔트리의 수는 더 많은 슬래브 검색표(200d)를 추가함으로써 증가될 수 있다. 검색키[39:0](210)에 대응하는 검색표(200c-d)의 하나에서의 매퍼 엔트리에 저장된 값에 대한 검색은 검색표(200c-d)에 병렬로 수행된다.

도 10D는 도 10C에 도시된 실시예에서 슬래브 검색표(200d)를 도시한다. 각 검색표는 도 3과 관련된 검색표(200)에 기술된 바와 같이 매퍼(106a-d)를 포함한다. 검색표(200c-d)의 각각의 매퍼(106a)에서의 매퍼 엔트리는 서브트리 엔트리(304(도 4))를 저장한다. 각 검색표(200c-d)는 매퍼(106a)에서의 매퍼 엔트리(504(도 6B))에 저장된 서브트리 엔트리 디스크립터(304(도 4))에 저장된 서브트리 인덱스(312)를 다음 메퍼(106b-d)로 보낸다. 그러나, 루트 인덱스(102(도 3))는 오직 마스터 검색표(200c)에서의 매퍼(106a)에 저장된다. 노-엔트리는 슬래브 검색표(200d)에서의 매퍼(106a)에 저장되며, 이는 하나 이상의 검색표(200b, 200d)에서 키에 대응하는 루트 인덱스의 저장을 피하기 위함이다.

슬래브 룩업(200d)에서의 멀티레벨 검색(904)의 결과는 최종 인덱스 로직(1004)로 보내진다. 상기 최종 인덱스 로직(1004)은 출력 결과(1002a)로써 마스터 검색표(200c)로부터 보내진 입력 결과(1000a) 또는 멀티레벨 검색(904)의 결과를 보낸다. 루트 인덱스(102(도 3))가 멀티레벨 검색의 결과(904)로써 포함된다면, 멀티레벨 검색 결과는 출력 결과(1002a)로써 보내진다. 루트 인덱스(102(도 3))가 입력 결과(1000a)에 포함된다면, 입력 결과(1000a)는 출력결과(1002a)로써 보내진다. 루트 인덱스(102(도 3))가 입력 결과(1000a)도 멀티레벨 검색 결과(904)에도 포함되지 않으면, 멀티레벨 검색 결과(904)는 출력 결과(1002a)로써 보내진다.

도 10C에 도시된 바와 같이, 마스터 검색표(200c) 및 슬래브 검색표(200d)는 입력 결과(1000a)로 레벨된 공통 버스를 통해 연결된다. 상기 루트 인덱스(102(도 3))는 오직 출력 결과(1002a) 상에 슬래브 검색표(200d)로부터 보내진다. 하나 이상의 슬래브 검색표(200d)가 존재한다면, 확장된 검색표에 대한 루트 인덱스(102(도 3))는 최종 슬래브 검색표에 의해 제공된다. 이러한 실시예는 도 10A와 관련되어 기술된 멀티 드라이버 최종 루트 인덱스(900)의 실행을 피하게 되지만, 입력 결과(1000a)에 대한 더 많은 드라이버 외부 핀을 요구하게 된다.

도 11A-B는 검색표(200a-b(도 10A) 또는 200c-d(도 10C)) 사이에서 도 2B에 도시된 루트의 분포의 이진 트리 표현을 도시한다.

도 11A는 마스터 검색표(200a(도 10A) 또는 200c(도 10C))에 저장된 루트의 이진 트리 표현을 도시한다. 도 2B에 도시된 루트의 이진 트리 표현에 도시된 서브트리B는 마스터 검색표(200a)에 포함되지 않는다. 노드 130^{1}-130^{22}는 도 3과 관련하여 기술된 바와 같이 검색표(200a)에서 매퍼(106a)에 코드된다. 마스터 검색표(200a)에 저장된다면, 서브트리B가 인덱스될 수 있는 노드는 간결화된 서브트리를 나타내는 X로 그래트에 나타난다. 마스터 검색표(200a)에서 노드 130^{23}에 대응하는 매퍼 엔트리(504(도 6B))는 더이상 서브트리 인덱스(312(도 4))를 서브트리B에 저장하지 않는다. 대신에 노-엔트리(300(도 4))는 마스터 검색표(200a)에서의 노드 130^{23}에 대응하는 매퍼 엔트리(504(도 6B))에 저장되며, 이는 노드 130^{23}에 대응하는 매퍼 엔트리가 또다른 슬래브 검색표(200b)에서의 서브트리 매퍼(418(도 5))에 저장되는 것을 의미한다.

도 11B는 슬래브 검색표(200b(도 10A) 또는 200d(도 10C))에서의 서브트리 매퍼(418(도 5))에서 매퍼 엔트리(504(도 6B))에 저장된 루트의 이진 트리 표현을 도시한다. 슬래브 검색표(200b)에 저장된 루트의 이진 트리 표현은 서브트리A가 포함되지 않은 도 2B에 도시된 이진 트리 표현과는 다르다. 따라서, 노드 130^{1}-130^{3} 및 130^{5}-130^{32}는 도 2B와 관련되어 기술된 바와 같이 코드된다. 슬래브 검색표(200b)에서 노드 130^{4}에 대응하는 매퍼 엔트리(504(도 6B))는 더이상 서브트리 인덱스(312(도 4))를 서브트리A에 저장하지 않는다. 대신에 슬래브 검색표(200b)에서의 노드 130^{4}에 대응하는 매퍼 엔트리는 노-엔트리(300(도 4))저장되며, 이는 노드 130^{4}에 대응하는 매퍼 엔트리가 또다른 검색표에 저장되는 것을 의미한다. 따라서, 서브트리A에 대한 서브트리 인덱스 및 호스트에 대한 루트 인덱스(138(도 11A))는 마스터 검색표(200a)에 저장되며, 서브트리B에 대한 서브트리 인덱스 및 호스트에 대한 루트 인덱스(140)는 슬래브 검색표(200b)에 저장된다. 상기 슬래브 검색표(200b, 200d)는 오직 서브트리에 대한 결과를 저장한다; 즉, 슬래브 검색표(200b, 200d)는 제1레벨 매퍼(106a)에 결과를 저장하지 않는다.

도 11A 및 도 11B와 관련하여, 키(210a)의 제1부분을 지닌 슬래브 매퍼 레벨_1(1104a) 또는 마스터 매퍼 레벨_1(1102(도 3)a)에서 임의의 노드 130^{9}-130^{12}에 대한 검색은 마스터 검색표(200a, 200c)에서의 매퍼(106a)에서 매퍼 엔트리(504(도 6B))에서의 루트 엔트리(302(도 4))에 저장된 r1(116)이 되며, 슬래브 검색표(200b, 200d)에서의 매퍼(106a)에서 매퍼 엔트리(504(도 6B))에 저장된 노-엔트리(300(도 4))가 된다. 마스터 검색표(200a, 200c)에 저장된 루트 엔트리(302(도 4))는 입력 결과(1000a)를 슬래브 검색표(200b, 200d)로 전하고, 출력 결과(1002a) 상의 슬래브 검색표(200b, 200d)에 의해 전달된다.

키(210a)의 제1부분을 지닌 노드 130^{4}에 대한 검색은 마스터 검색표(200a)에서의 매퍼(106a)에서 매퍼 엔트리(504(도 6B))에서의 서브트리 엔트리 디스크립터(304(도 4))에 저장된 서브트리A에 대한 서브트리 인덱스(312(도 4))가 된다. 상기 서브트리 인덱스(312)는 마스터 검색표(200a)에서의 매퍼(106b)로 전달되며, 이는 마스터 검색표(200a)에 저장된 루트 엔트리(302(도 4))에 대한 검색을 계속하기 위함이다.

상기 키(210a)의 제1부분을 지닌 노드 130^{23}에 대한 검색은 마스터 검색표(200a)에서의 매퍼(106a)에서 매퍼 엔트리(504(도 6B))에 저장된 노-엔트리(300(도 4)) 및 슬래브 검색표(200b)에서의 매퍼(106a)에서 매퍼 엔트리(504(도 6B))에 저장된 서브트리 엔트리 디스크립터(304(도 4))가 된다. 따라서, 루트 엔트리(302(도 4))에 대한 검색은 슬래브 검색표(200b)에서의 매퍼(106b)에서 상기 키(210b)의 다음 부분을 지니고 계속 진행된다.

도 12는 도 10A에 도시된 검색표(200a-b) 사이에서 매퍼 엔트리(504(도 6B))에 저장되도록 루트 엔트리(302(도 4))를 분배하는 방법을 묘사하는 플로챠트이다. 동일한 방법은 도 10C에 도시된 검색표(200c-d)에 적용한다. 매퍼 엔트리에 저장되기 위하여 루트 엔트리(302(도 4))는 검색표(200a-b)에 저장되기 이전에 프로세서(미도시)에 의해 메모리에 처음에 저장된다.

상기 루트 엔트리(302(도 4))가 메모리에 저장되는 동안, 검색표(200a-b(도 10A))의 각각에 저장되도록 하기 위한 루트 엔트리(302(도 4))의 수는 산출된다. 매퍼 레벨_1(1104a(도 11B))에 대한 루트 엔트리(302(도 4))는 검색표(200a)에서 매퍼(106a)에 저장된다. 매퍼 레벨_1(1104a(도 11B))에 대한 루트 엔트리(304(도 4))는 검색표(200a-200b)의 각각에서 매퍼(106a)에 저장된다.

단계 1200에서, 검색표(200a-200b(도 10A))의 각각에서의 매퍼(106a)에서 서브트리 엔트리(304(도 4))의 각각에 대해 저장되기 위하여 루트 엔트리(302(도 4))의 수는 검색표(200a-b(도 10A)) 사이에서 루트 엔트리(302(도 4))를 분배하는 방법을 결정하기 위해 계산된다. 루트 엔트리(302(도 4))를 저장하도록 요구된 매퍼 엔트리(504(도 6B))의 수가 결정된 후, 프로세싱은 단계 1202로 진행된다.

단계 1202에서, 서브트리를 위해 저장되도록 매퍼 엔트리(504(도 6B))의 총수는 검색표(200a-b(도 10A))의 수에 의해 루트 엔트리(302(도 4))의 수를 결정하도록 분배되며, 이는 각 검색표(200a-b(도 10A))에 저장되기 위함이다. 프로세싱은 단계 1204로 진행된다.

단계 1204에서, 루트 엔트리(302(도 4))는 선택된 검색표(200a-b)에서의 서브트리 매퍼(418(도 5))에서 매퍼 엔트리(504(도 6B))에 저장된다. 프로세싱은 단계 1206로 진행된다.

단계 1206에서, 선택된 검색표(200a-b(도 10A))에서의 서브트리 매퍼(418(도 5))에서 매퍼 엔트리(504(도 6B))에 저장된 루트 엔트리의 수가 가능한 검색표(200a-b(도 10A))의 수가 n일때의 1/n보다 작다면, 프로세싱은 단계 1208로 진행된다. 만약 그렇지 않다면, 선택된 검색표(200a-b)는 이미 매퍼 엔트리의 총수의 1/n을 저장하고 단계 1208로 진행된다.

단계 1208에서, 상기 선택된 검색표(200a-b)가 매퍼 엔트리의 총수의 1/n을 저장하고, 노-엔트리(300(도 4))는 선택된 검색표(200a-b)에서 임의의 잔여 서브트리 노드를 위해 저장되며 이는 각 서브트리에 대한 루트 인덱스가 현재 선택된 검색표에 저장되지 않았기 때문이다. 프로세싱은 단계 1210로 진행된다.

단계 1210에서, 상기 모든 루트 엔트리가 저장되었다면, 프로세싱은 완료되고, 그렇지 않다면, 프로세싱은 단계 1212로 진행된다.

단계 1212에서, 다음 검색표(200a-b(도 10A))가 선택된다. 프로세싱은 단계 1204로 진행된다.

루트 엔트리는 IP 어드레스에 대응하는 루트 인덱스에 대한 검색 이전에, 검색표(200a-b(도 10A)) 사이에서 분포된다. 상기 검색은 각각의 검색표(200a-b(도 10A))에 병렬로 수행된다. 각 검색표에 병렬로 실행된 검색 방법은 검색표(200a-b(도 10A))의 하나에 대해 기술되었다.

도 13은 도 10C에 도시된 임의의 하나의 검색표(200c-d)에 저장된 검색키에 대응하는 값에 대한 검색키를 갖는 검색 방법을 도시하는 플로챠트이다.

단계 1300에서, 각 검색표(200c-d(도 10C))는 검색키(210)를 수용한다. 각 검색표(200c-d)에서 매퍼(106a)는 상기 키(210a)의 제1부분에 대응하는 값에 대해 검색된다. 프로세싱은 단계 1302로 진행된다.

단계 1302에서, 매퍼(106a)에서의 매퍼 엔트리(504(도 6B))에 저장된 엔트리는 판독된다. 마스터 검색표(200c)에서 매퍼 엔트리(504(도 6B))는 노-엔트리(300(도 4)), 루트 엔트리(302(도 4)) 또는 서브트리 엔트리 디스크립터(304(도 4))를 저장할 수 있다. 슬래브 검색표(200d)에서의 매퍼 엔트리(504(도 6b))는 노-엔트리(300(도 4)) 및 서브트리 엔트리 디스크립터(304(도 4))를 저장할 수 있다. 상기 각 검색표(200)에서의 매퍼 엔트리가 루트 엔트리(302(도 4))를 저장한다면, 상기 엔트리는 유효값이 되고, 검색표(200c-d)에서 후속되는 매퍼의 다음 검색은 더이상 요구되지 않으며, 프로세싱은 단계 1310로 진행된다. 만약 그렇지 않으면, 프로세싱은 단계 1304로 진행된다.

단계 1304에서, 상기 엔트리가 서브트리 엔트리 디스크립터(304(도 4))를 저장하면, 검색표(200c-d)에서 추가적인 검색이 요구되며 프로세싱은 단계 1306로 진행된다. 만약 그렇지 않다면, 상기 엔트리는 노-엔트리를 저장하고, 이는 추가적인 검색이 요구되지 않음을 의미하며, 프로세싱은 단계 1310로 진행된다.

단계 1306에서, 상기 검색은 선택된 서브트리에서 계속 진행된다. 다음 레벨 매퍼(106b-d(도 3))는 이전 레벨의 검색으로부터 야기된 서브트리 인덱스(312(도 4)) 및 상기 키(210b-d)의 다음 부분에 좌우되어 검색된다. 프로세싱은 단계 1308로 진행된다.

단계 1308에서, 현재 레벨 매퍼(106b-d)에서의 검색으로부터 야기된 매퍼 엔트리에 좌우되는 검색을 계속할지 아닐지를 결정한다. 상기 매퍼 엔트리(504(도 6B))가 서브트리 엔트리 디스크립터(304(도 4))를 저장한다면, 상기 검색은 다음 레벨 매퍼(106b-d)로 진행되고 프로세싱은 단계 1306로 진행된다. 상기 매퍼 엔트리(504(도 6B))가 서브트리 엔트리 디스크립터(304(도 4))를 저장하지 않으면, 추가적인 검색은 요구되지 않으며, 프로세싱은 단계 1310로 진행된다.

단계 1310에서, 상기 검색 결과는 또다른 검색표로부터 수용된 입력 결과(1000a)와 비교된다. 예를 들면, 상기 검색표가 슬래브 검색표(200d)이라면, 마스터 검색표(200c)에서의 검색으로부터의 입력 결과는 입력 결과(1000a) 상에서 검색표(200d)로 보내지며, 슬래브 검색표(200d)에서의 검색 결과와 비교된다. 프로세싱은 단계 1312로 진행된다.

단계 1312에서, 상기 입력 결과(1000a) 및 현재 검색표(200d)에서의 검색 결과가 다르면, 프로세싱은 단계 1314로 진행된다. 두개의 입력 결과(1000a) 및 현재 검색표(200d)에서의 검색 결과가 동일하면, 두개의 유효 결과는 분리된 검색표(200c-d)에서 매퍼 엔트리(504(도 6B))에 저장된다. 두개의 유효 결과는 동일한 키(210)에 대해 저장되지 않아야 하며, 프로세싱은 단계 1316로 진행된다.

단계 1314에서, 상기 입력 결과(1000a)는 그것이 유효한지를 결정하기 위해 조사된다. 입력 결과(1000a)는 그것이 루트 엔트리(302(도 4))라면 유효하다. 상기 입력 결과(100a)는 그것이 노-엔트리(300(도 4))이거나 서브트리 엔트리 디스크립터(304(도 4))이면 무효하며, 서브트리 엔트리 디스크립터(304(도 4)), 루트 엔트리(302(도 4)) 및 노-엔트리(300(도 4))은 이미 도 4와 관련되어 기술되었다. 상기 입력 결과(1000a)가 무효하면, 프로세싱은 단계 1318로 진행된다. 만약 그렇지 않다면, 프로세싱은 단계 1320로 진행된다.

단계 1318에서, 상기 입력 결과(1000a)는 유효하며 현재 검색표(200d)에서의 검색으로부터 야기된 결과는 무효하다. 상기 입력 결과(1000a)는 출력 결과(1002a) 상에서 현재 검색표(200d)로부터 보내진다. 현재 검색표(200d)가 최종 검색표와 동일하거나 입력 결과(1000a)로써 다음 검색표로 보내진다면, 상기 입력 결과(1000a)는 루트 인덱스(102(도 3))으로써 보내진다. 프로세싱은 완료된다.

단계 1316에서, 두개의 유효 결과 값은 다른 검색표에서 키에 대해 저장된다. 에러는 상기 검색표(200c-d)에서의 루트 엔트리의 저장 과정동안 발생되었다. 에러 코드는 에러가 보정될 수 있도록 생성된다. 프로세싱은 완료된다.

단계 1320에서, 현재 검색표(200d)의 검색으로부터의 결과 및 입력 결과(1000a) 모두 유효하지 않다. 비록 무효하다 하더라도, 현재 검색표(200d)에서 검색 결과는 입력 결과(1000a)로써 다음 검색표로 보내진다. 프로세싱은 완료된다.

도 14는 도 10A에 도시된 검색표(200a-b)의 하나에 저장된 검색키에 대응하는 값에 대한 검색 방법을 도시한 플로챠트이다.

단계 1340에서, 두개의 검색표(200a-b)에서의 제1레벨 매퍼(106a)는 키(210a)의 제1부분에 대응하는 값에 대해 검색된다. 프로세싱은 단계 1342로 진행된다.

단계 1342에서, 유효 결과 값이 상기 키의 제1부분(210a)을 지닌 제1레벨 매퍼(106a)를 검색한 후에 발견된다면, 프로세싱은 단계 1352로 진행된다. 만약 그렇지 않다면, 프로세싱은 단계 1344로 진행된다.

단계 1344에서, 상기 키(210a)의 제1부분을 지닌 제1레벨 매퍼(106a)의 검색으로부터 야기된 값이 서브트리 엔트리 디스크립터(304(도 4))라면, 프로세싱은 단계 1346로 진행된다. 만약 그렇지 않다면, 상기 키에 대한 유효 값은 현재 검색표에 저장되지 않으며, 프로세싱은 완료된다.

단계 1346에서, 유효 값에 대한 검색은 이전 레벨 매퍼의 검색 동안 발견된 서브트리 엔트리 디스크립터(304(도 4))에서 입증된 서브트리에서 계속 진행한다. 다음 레벨 매퍼는 다음 레벨의 검색으로부터 야기된 서브트리 셀렉트 (subtree select) 및 상기 키(210b-c)의 다음 부분에 좌우되는 값에 대해 검색된다. 프로세싱은 단계 1348로 진행된다.

단계 1348에서, 상기 검색 결과는 다음 레벨 매퍼의 검색이 요구되는지를 결정한다. 현재 검색으로부터 야기된 엔트리는 루트 엔트리(302), 노-엔트리(300(도 4)) 또는 서브트리 엔트리 디스크립터(304(도 4))를 저장할 수 있다. 상기 엔트리가 서브트리 엔트리 디스크립터(304(도 4))를 저장한다면, 추가적인 검색이 필요하며 프로세싱은 단계 1346로 진행된다. 상기 엔트리가 서브트리 엔트리 디스크립터(304(도 4))를 저장하지 않으면, 프로세싱은 단계 1350로 진행된다.

단계 1350에서, 상기 엔트리가 루트 인덱스(102(도 3))를 저장한다면, 프로세싱은 단계 1352로 진행된다. 만약 그렇지 않다면, 상기 엔트리는 또다른 검색표에 저장된다. 프로세싱은 완료된다.

단계 1352에서, 상기 키에 대응하는 유효 값이 현재 검색표에 저장된다. 상기 유효 값은 상기 키에 대응하는 루트 인덱스(102(도 3))로써 보내진다. 프로세싱은 완료된다.

희박 모드(Sparse Mode)

도 5로 되돌아 가서, 서브트리 엔트리(404)는 256 노트 서브트리에서 노드당 하나, 256 가능한 루트 인덱스까지 액세스하기 위해 제공된다. 상기 루트 인덱스는 서브트리 매퍼(418(도 5))에서의 매퍼 엔트리 504^1-504^n에 저장된다. 서브트리 매퍼(418(도 5))에서의 매퍼 엔트리(504(도 6B))에 대한 매퍼 어드레스(416)는 데이타 필드(406)에 저장된 조밀 서브트리 디스크립터 및 서브트리 엔트리(404)에서의 포인터 필드(408)에 저장된 서브트리 포인터에 좌우되어 결정된다. 조밀 서브트리 디스크립터의 포맷은 이미 도 6A-6B와 관련되어 기술되었다. 상기 조밀 서브트리 디스크립터는 상기 256 노드 서브트리에서의 각 노드에 대해 노드 비트(502(도 6B))를 저장한다. 그러나, 모든 서브트리는 각 256 노드에 대한 다른 루트 인덱스를 갖으며, 예를 들면, 서브트리는 오직 하나의 루트 인덱스를 가질 것이다.

도 15는 제1매퍼 레벨(112a)에서의 서브트리 엔트리(304(도 4))에 의해 인덱스된 제2매퍼 레벨(112b)에서 희박 서브트리A 및 조밀 서브트리B의 이진 트리 표현을 도시한다. 매퍼(106a)에서 s1에 대한 서브트리 엔트리 디스크립터(304(도 4))는 서브트리A의 서브트리 엔트리(404)에 대한 서브트리 인덱스(312)를 저장한다. 매퍼(106a)에서 s0에 대한 서브트리 엔트리 디스크립터(304(도 4))는 서브트리B의 서브트리 엔트리(404)에 대한 서브트리 인덱스(312)를 저장한다.

조밀하게 실장된 서브트리B는 11개의 루트 인덱스를 가진다; 즉, r6-r16 및 6개의 서브트리 엔트리; 즉, s2-s7이다. 서브트리B에 대한 서브트리 엔트리(304(도 4)) 및 루트 엔트리(302(도 4))를 저장하는 매퍼 엔트리(504(도 6B))에 대응하는 매퍼 어드레스(416)는 도 6B와 관련하여 이미 기술된 바와 같이 조밀 서브트리 디스크립터에 코드된다.

희박하게 실장된 서브트리A는 2개의 루트 인덱스를 저장한다; 즉, r1 및 r2이다. 그들이 조밀 서브트리 디스크립터에 저장된다면, 전체 서브트리 엔트리(404)는 매퍼 엔트리(504(도 6B))에 대한 3개의 매퍼 어드레스(416)를 제공하도록 사용된다; 즉, r0, r1, 및 r2이다.

검색표(200)에 저장된 루트의 수는 복수개의 희박 서브트리 디스크립터중 하나에서 희박 서브트리를 코딩하고 서브트리 엔트리(404)에서의 조밀 서브트리 디스크립터에서 조밀하게 실장된 서브트리를 코딩함으로써 증가될 수 있다.

조밀하게 실장된 서브트리는 16개 또는 그 이상의 매퍼 엔트리(504(도 6B))를 가지며, 서브트리 엔트리(404)에서의 데이타 필드(406)는 도 6A-6B와 관련하여 기술된 바와 같이 조밀 서브트리 디스크립터를 저장한다. 희박하게 실장된 서브트리는 15개 또는 그 이하의 매퍼 엔트리(504(도 6B))를 가진다; 서브트리 엔트리(404)에서의 데이타 필드는 복수개의 희박 서브트리 디스크립터를 저장한다. 희박 서브트리 디스크립터에서의 희박하게 실장된 서브트리를 저장하는 능력을 제공함으로써, 더 많은 서브트리는 서브트리 메모리(400)에 저장될 수 있으며, 따라서, 더 많은 루트 엔트리는 검색표(200)에 저장될 수 있다.

도 16A-C는 도 4에 도시된 서브트리 엔트리 디스크립터(304(도 4)) 및 도 5에 도시된 서브트리 엔트리(404)에서의 포인터 필드(408) 및 데이타 필드(406)에 대한 변경을 도시하며, 이는 서브트리 엔트리(404)에서의 복수개의 희박 서브트리 디스크립터의 저장을 허용하기 위함이다.

도 16A로 다시 돌아가면, 희박 모드에서 형성된 서브트리 엔트리(404)에서의 데이타 필드(406)는 도 6B와 관련하여 기술된 서브트리 노드당 1비트를 지닌 조밀 서브트리 디스크립터 대신에 복수개의 희박 서브트리 디스크립터 1400^1-1400^n을 저장한다. 각 희박 서브트리 디스크립터 1400^1-1400^n는 노드 디스크립터 1402^1-1402^n를 포함한다. 노드 디스크립터 1402^1-1402^n는 서브트리에서 와전히 부호화된 루트를 나타내는 9-비트 값이다. 상기 노드 디스크립터 1402^1-1402^n는 서브트리에서 단일 노드 또는 복수개의 노드를 나타낸다.

도 16B로 다시 돌아가면, 희박 서브트리 디스크립터의 저장을 지원하기 위하여, 모드 필드(1404)는 서브트리 엔트리(404)에서의 포인터 필드(408)에 추가된다. 또한, 포인터 필드(408)는 서브트리 엔트리(404)당 총 32개의 매퍼 어드레스

(416)를 제공하는 16개의 할당된 매퍼 어드레스(416)를 포함하는 각 블록을 지닌 블록 베이스 어드레스(600^2) 및 블록 베이스 어드레스(600^1)을 저장한다. 상기 모드 필드(1404)는 모드 값을 저장한다. 상기 모드 필드(1404)에 저장된 모드 값은 서브트리 엔트리(404)에 저장된 희박 서브트리 디스크립터(1400^1-1400^n)의 수 및 각 희박 서브트리 기술(1400^1-1400^n)어에 저장된 노드 디스크립터(1402^1-1402^n)의 수를 나타낸다. 표 2는 각 모드에 대한 서브트리 엔트리(404)의 구성을 도시한다.

[표 2]

	노드/ 서브트리		비트	서브트리/ 서브트리 엔트리	루트/ 서브트리 엔트리		미사용	서브트리 엔트리를 위해 요구된 매퍼 엔트리	
모드	최대	최소			최대	최소		최대	최소
0	1	1	9	16	16	16	112	32	32
1	2	2	18	10	20	20	76	30	30
2	3	3	27	8	24	24	40	32	32
3	4	4	36	6	24	24	40	30	30
4	7	5	63	4	28	20	4	32	24
5	15	8	135	2	30	16	-14	32	18

표 2와 관련하여, 예를 들어, 서브트리 엔트리(404)에서의 포인터 필드(408)에서 모드 필드(1404)에 저장된 모드 값이 '4'로 설정된다면, 서브트리 엔트리(404)에서의 각 희박 서브트리 디스크립터(1400)는 5와 7사이의 노드 디스크립터(1402^1-1402^n)를 저장한다. 각 노드 디스크립터(1402^1-1402^n)는 9비트를 저장한다. 희박 서브트리 디스크립터(1400)에 저장된 총 비트수는 희박 서브트리 디스크립터(1400^1-1400^n)당 노드 디스크립터(1402^1-1402^n)수 곱하기 9에 의해 계산된다. 모드 4에 대한 희박 서브트리 디스크립터(1400)당 비트수를 산출하는데 있어서, 7개의 노드 디스크립터(1402)를 지닌 희박 서브트리 디스크립터(1400)는 63비트(7 노드 디스크립터 * 9비트 = 63)를 저장한다.

서브트리 엔트리(404)당 희박 서브트리 디스크립터(1400)의 수는 데이타 필드(406)에서의 비트수를 희박 서브트리 디스크립터(1400^1-1400^n)에서의 비트수로 나눔으로써 계산된다. 모드 4에 대해, 데이타 필드(406)에서의 비트수는 256이고, 희박 서브트리 디스크립터에서의 비트수는 63이다. 따라서, 희박 서브트리 디스크립터(1400^1-1400^n)의 수는 4(int(256/63)=4)이다.

서브트리 엔트리(404)당 노드 디스크립터(1402^1-1402^n)의 총수는 서브트리당 노드수 곱하기 서브트리 엔트리(404)당 서브트리수이다. 희박 서브트리 디스크립터(1400^1-1400^n)에 저장된 7개의 노드 디스크립터(1402^1-1402^n)가 존재한다면(7*4=28), 모드 4에 대한 계산은 서브트리 엔트리(404)당 노드 디스크립터(1402)의 총수는 28이고, 희박 서브트리 디스크립터(1400^1-1400^n)당 5개의 노드 디스크립터(1402)가 존재한다면(5*4=20), 그 수는 20이 된다.

표 2에서 매퍼 엔트리 열은 얼마나 많은 서브트리 매퍼(418(도 5))에서의 매퍼 엔트리(504(도 6B))가 서브트리 엔트리(404)에 의해 사용되는지를 나타낸다. 상기 매퍼값은 하나에 의한 서브트리당 노드를 증가시키고 희박 서브트리 디스크립터에서의 서브트리수를 곱함으로써 계산된다. 서브트리당 노드는 서브트리당 노드수보다 하나 이상의 매퍼 엔트리가 서브트리에 대한 기본 엔트리를 저장하도록 요구되기 때문에 하나씩 증가된다.

표 2에서의 모드 4행과 관련하여, 희박 서브트리 디스크립터(1400)당 7개의 노드 디스크립터(1402)가 존재한다면, 32((7+ 1)*4=32)개의 매퍼 엔트리는 서브트리 엔트리(404)당 요구되며, 희박 서브트리 디스크립터(1400)당 5개의 노드 디스크립터(1402)가 존재한다면, 24((5+ 1)*4=24)개의 노드 디스크립터(1402)가 희박 서브트리 디스크립터(1400)당 요구된다. 서브트리당 노드수 및 서브트리 엔트리(404)당 서브트리수는 서브트리 엔트리(404)당 노드 디스크립터의 최대수가

30을 초과하지 않도록 선택되면, 이는 서브트리 매퍼(418(도 5))에서의 매퍼 어드레스(416)가 16개의 블록 증가에서 할당되기 때문이다. 32개의 매퍼 어드레스(416)는 포인터 필드(408)에 저장된 2개의 블록 베이스 어드레스(600^1, 600^2)를 저장함으로써 제공된다.

도 16C에서, 서브트리 메모리(400)에서의 각 서브트리 엔트리(404)는 도 6B와 관련하여 기술된 바와 같이 조밀 모드에서 또는 희박 모드에서 구성될 수 있다. 조밀 모드에 대해 도 4와 관련하여 기술된 서브트리 매퍼(418(도 5))에 저장된 서브트리 엔트리 디스크립터(304(도 4))는 서브트리 엔트리(404)가 조밀 모드 또는 희박 모드에서 코드되는지의 지시를 제공함으로써 희박 모드를 허용하도록 변형된다. 이러한 지시자는 타입 필드(1406)에 의해 제공된다.

타입 필드(1406)의 상태는 서브트리 엔트리(404)가 조밀 모드로 구성되었는지 희박 모드로 구성되었는지를 지시한다. 상기 서브트리 엔트리(404)가 희박 모드로 구성되었다면, 희박 서브트리 디스크립터 셀렉트 필드(1408) 및 서브트리 인덱스(312)에 저장된 값은 희박 서브트리 디스크립터(1400)를 선택하도록 사용된다. 상기 희박 서브트리 디스크립터 셀렉트(1408)는 도 16과 관련하여 더 자세히 뒤에서 기술될 것이다.

도 17은 블록 오프셋(714)을 제공하기 위해 도 8에 도시된 오프셋 로직(700)에서의 희박 모드 로직(1502)을 도시하며, 이는 희박하게 실장된 서브트리에서의 노드를 위해 매퍼 엔트리(504(도 6B))를 선택하기 위함이다. 상기 희박 모드 로직(1502)은 서브트리 엔트리(404)에서의 희박 서브트리 디스크립터(1400)에 저장된 노드 디스크립터(1402)에 좌우되는 블록 오프셋(714)을 제공한다. 또한 상기 오프셋 로직(700)은 조밀 모드 로직(1500)을 포함한다. 조밀 모드 로직(1500)은 조밀하게 실장된 서브트리에서의 루트를 위해 블록 오프셋(714)을 제공하기 위해 카운트 로직(708) 및 노드 셀렉트(706)를 포함한다. 조밀 모드 로직(1500)은 이미 도 8과 관련하여 기술되었다.

상기 타입 필드(1406)의 상태가 서브트리 엔트리(404)가 희박 모드에서 구성된 것을 나타내면, 서브트리 엔트리(404)로부터 서브트리 데이타(412)는 희박 모드 로직(1502)으로 전달된다. 상기 희박 모드 서브트리 로직(1502)은 도 18과 관련하여 기술된다.

도 18은 도 17에서의 오프셋 로직(700)에 도시된 희박 모드 로직(1502)을 도시한다. 상기 희박 모드 로직(1502)은 서브트리 셀렉트 로직(1600), 멀티플렉서(1602), 내용 주소화 기억장치("CAM") 및 변환 로직(1604)을 포함한다. 선택된 서브트리 엔트리(404)에서의 데이타 필드(406)에 저장된 희박 서브트리 디스크립터(1400^1-1400^n)는 서브트리 데이타(412) 상에서 오프셋 로직(700)으로 전해진다. 상기 오프셋 로직(700)은 희박 서브트리 디스크립터(1400^1-1400^n)를 희박 모드 로직(1502)에서의 멀티플렉서(1602)로 보낸다. 상기 서브트리 데이타(412)에서의 희박 서브트리 디스크립터(1400^1) 중 하나는 서브트리 셀렉트 로직(1600)에 의해 생성된 셀렉트(1614)에 의해 선택된다.

상기 서브트리 셀렉트 로직(1600)은 희박 서브트리 디스크립터 셀렉트(1408)의 상태에 좌우되는 희박 서브트리 디스크립터(1400^1)를 선택하도록 셀렉트(1614)를 생성하며, 상기 희박 서브트리 디스크립터 셀렉트는 선택된 서브트리 엔트리(404)에서의 포인터 필드(408)에 저장된 이전의 매퍼 레벨 및 모드(1404)에서 선택된 매퍼 엔트리로부터 전해진다. 표 3은 모드 4 서브트리 엔트리(404)에 대해 멀티플렉서(1602)로부터 선택된 희박 서브트리 디스크립터(1610)상에 전해진 각 서브트리 데이타 비트(412) 및 선택된 희박 서브트리 디스크립터(1400^1)를 도시한다. 표 2에서 모드 4행과 관련하여, 4개의 희박 서브트리 디스크립터는 모드 4 서브트리 엔트리(404)에 저장될 수 있다. 4개의 희박 서브트리 디스크립터(1400) 각각은 63비트이고 7개부터 5개의 노드 디스크립터(1402)를 저장할 수 있다. 따라서, 4개의 희박 서브트리 디스크립터(1400)의 각각은 63비트 이진수에서 시작한다. 상기 제1희박 서브트리 디스크립터(1400^1)는 데이타 필드(406)에서 비트 62:0에 저장된다. 상기 제2희박 서브트리 디스크립터(1400^2)는 데이타 필드(406)에서 비트 125:63에 저장된다. 상기 제3희박 서브트리 디스크립터(1400^3)는 데이타 필드(406)에서 비트 188:126에 저장되며, 상기 제4희박 서브트리 디스크립터(1400^4)는 데이타 필드에서 비트 251:189에 저장된다. 상기 데이타 필드(406)에서 각 비트는 희박 서브트리 디스크립터 셀렉트(1408)에 의해 선택된다. 예를 들어, 표 3을 보면, 희박 서브트리 디스크립터 셀렉트(1408)가 '0001'이라면, 제2희박 서브트리 디스크립터(1400^2)가 선택되며, 256비트 서브트리 데이타(412)의 비트 125:63은 선택된 희박 서브트리 디스크립터(1610)상의 멀티플렉서(1602)를 통해서 변환 로직(1604)으로 보내진다.

[표 3]

희박 서브트리 디스크립터 셀렉트	서브트리 데이타	희박 서브트리 디스크립터
0000	비트 62:0	1
0001	비트 125:63	2
0010	비트 188:126	3
0011	비트 251:189	4

상기 서브트리 메모리(400)에서 각 서브트리 엔트리(404)는 희박 모드 또는 조밀 모드로 구성될 수 있다. 희박 모드에서 구성된 각 서브트리 엔트리(404)는 모드(1404)를 통해 희박 서브트리 디스크립터(1400)당 다른 노드 디스크립터(1402)수를 저장하도록 구성된다. 희박 모드에서 구성된 서브트리 엔트리(404)에서의 모든 희박 서브트리 디스크립터(1400)는 희박 서브트리 디스크립터(1400)당 동일한 노드 디스크립터(1402)수를 저장한다.

노드 디스크립터(1402)는 서브트리에서 다중 노드를 나타내도록 코드될 수 있다. 노드 디스크립터(1402)에 의해 표시된 다중 8비트 노드는 8비트 중 일부를 마스킹함으로써 결정된다. 각 노드 디스크립터(1402)를 지닌 마스크 비트를 저장하는 대신에, 9개의 비트 노드 디스크립터(1402)는 노드 디스크립터(1402)에 의해 표시된 8비트의 광역 노드를 완전히 부호화하는데 사용된다. 상기 8비트 광역 노드는 런 비트 길이 부호화(run bit length encoding)를 사용하는 9개 비트 광역 노드에서 부호화된다. 런 비트 길이 부호와는 노드의 8비트가 마스트를 허용한다.

상기 변환 로직(1604)은 선택된 희박 서브트리 디스크립터(1400)에 저장된 9비트 광역 노드 디스크립터(1402^1-1402^n)를 'X'(don't care)에 대한 비트 셋을 포함하는 9비트 CAM 값으로 변환하며, 8비트 CAM 값을 CAM(1606)으로 로드한다. 변환 로직(1604)에 의한 8-비트 CAM 값으로의 9-비트 노드 디스크립터(1402)의 변환의 예는 하기 표 4에 도시된다.

[표 4]

8-비트 값	9-비트 코트
101100XX	101100100
100XXXXX	100100000
001111XX	001111100
0011XXXX	001110000

9비트 코트 컬럼은 노드 디스크립터(1402)내에 저장된 값을 나타낸다. 표 4에서 제1열을 보면, 노드 디스크립터(1402)내에 저장된 9비트 코드는 '101100100' 이며, 대응되는 8 비트 값 '101100XX'는 CAM 1606내에 저장된다. 변환 로직(conversion logic)(1604)은 9 비트 코드를 오른쪽에서 왼쪽으로 조사함으로써 9비트 코드의 첫번째 비트가 1로 시작되게 전환한다. 오른쪽에서 왼쪽으로 9비트 코드내의 비트를 보면, 첫번째 두 비트는 0으로 시작하고, 세번째 비트는 1로 시작된다. 변환 로직(1604)은 첫번째 1의 오른쪽에 두개의 '0'이 있기 때문에 '100'을 두개의 미지수('XX')로 변환한다. 첫번째 1은 무시되고, 남아 있는 비트는 8비트 값의 다음 비트로 직접 복사된다.

표 4에서 제2열을 보면, 노드 디스크립터(1402)에 저장된 9비트 코드는 100100000이다. 변환 로직(1604)은 9비트 코드를 오른쪽에서 왼쪽으로 조사함으로써, 첫번째 비트가 1로 시작되게 전환한다. 다섯번째 숫자는 1을 저장한다. 9비트 코드는 다섯개의 LSBs (Least Significant Bits)가 미지수 'X'로 시작되는 8비트 코드 값으로 전환된다. 비트 길이 암호를 운용하는 9비트를 사용하는 노드 디스크립터(1402)를 저장함으로써, 단위 노드 디스크립터(1402) 당 요구되는 비트 수는 최소화되며, 검색표(200)내에 저장될 수 있는 노드 디스크립터(1402)의 수는 증가한다.

9비트 노드 디스크립터(1402)를 8비트 값으로 전환한 후, 변환 로직(1604)은 8비트 값을 CAM(1606)내로 로드한다. 8비트 값은 노드 디스크립터(1402)가 선택된 희박 서브트리 디스크립터(1400)내에 저장된 순서와 동일하게 CAM내로 로딩된다. 즉, 최단에서 최장 정합(match)이다. CAM(1606)은 단위 희박 서브트리 디스크립터(1400)당 노드 디스크립터(1402)의 최대수를 저장하기 위한 기억장치를 제공한다. 그리하여, 챔(1606)은 디폴트 매퍼 어드레스와 모드(5) 희박 서브트리 디스크립터(1400)에 대비하여 노드 디스크립터(1402)의 최대수를 저장하기 위한 15개의 엔트리를 제공하는 8비

트의 폭과 16엔트리의 깊이를 구비한다. CAM(1606)은 제3의 성능과 다중 정합 리졸버(multi-match resolver)를 그 내에 구비한다. 소규모 CAM (1606)은 진 연산 기억 장치(true contents addressable memory)를 제공하는 것과는 대조적으로 게이트내에 실행될 수 있다. 즉 CAM(1606)은 CAM을 모방하는 하드웨어 회로내에서 실행될 수 있다.

희박 서브트리 디스크립터(1400)내에 저장되는 노드 디스크립터(1402)의 수는 희박 서브트리 디스크립터(1400)가 저장되는 서브트리 엔트리(404)를 결정한다. 특별 모드를 위한 범위내의 희박 서브트리 디스크립터(1400) 저장 노드 디스크립터(1402)는 동일한 서브트리 엔트리(404)내에 저장된다. 디폴트 매퍼 어드레스는 각 서브트리를 위해 디폴트 경로를 계산한다. 디폴트 8비트 값은 디폴트 매퍼 어드레스를 계산하기 위해 CAM내의 제1로케이션내에 영구적으로 저장된다.

선택된 희박 서브트리(1400)를 위한 8비트 값이 CAM(1606)내에 로딩된 후, CAM(1606)은 키(210b)의 다음 부분을 조사한다. 키(210b)의 다음부분에서 비트의 가장 큰 수를 매칭하는 CAM(1606)내의 엔트리가 선택된다. CAM의 검색으로부터 발생되는 정합 어드레스는 블록 오프셋(714)으로 전송된다. 블록 오프셋(714)은 서브트리 매퍼(418)내에 저장된 경로에 대응하는 매퍼 엔트리를 위해 매퍼 어드레스(416)를 결정하는 데 사용한다.

도 19 a-d는 희박 서브트리(sparse populated subtree)(1700)내의 노드를 위한 블록 오프셋(714)의 선택을 도시하고 있다. 도 17a는 희박 서브트리내의 경로의 도표다. 세개의 경로 r0,r1 및 r2,r1의 하나에 대응되는 서브트리(1700)내의 노드는 서브트리(1700)에 대해 디폴트 루트이다. 두개의 경로 r1,r2는 희박 서브트리 디스크립터(1400)내의 노드 디스크립터(1402^1, 1402^2)내에 암호화된다. 디폴트 경로 r0 값은 CAM(1606)내의 제1엔트리(1702)내에 영구적으로 저장된다. 표2에 의하면, 두개의 노드 디스크립터(1402)를 구비하는 희박 서브트리 디스크립터(1400)는 '1'로 시작되게 하는 모드 필드(1404)를 구비하는 서브트리 엔트리(404)내에 저장된다.

서브트리(1700)를 보면, r2는 10XXXXXX를 매칭하는 모든 노드에 대응하고, r1은 01XXXXXX을 매칭하는 모든 노드에 대응한다. 희박 서브트리 디스크립터 (1400)내의 각 경로를 규정하기 위한 각 노드 디스크립터 (1402^1,1402^2)에 의해 요구된 비트 수를 최소화 하기 위해, 노드 디스크립터(1402^1,1402^2) 는 운용 비트 길이 암호를 사용하여 코드화된다. 코딩 방법은 노드를 완전히 암호화하기 위해 사용된 비트의 수보다 하나 더 많은 비트를 사용한다. '1'은 첫번째 미지수 X의 로케이션내에 삽입되며, 남아있는 X는 0으로 코팅된다. 따라서, 경로 10XXXXXX 는 1010000으로 번역되고, 101XXXXX는 010100000으로 번역된다.

도 19b는 희박 서브트리 디스크립터(1400)내의 노드 디스크립터 (1402^1,1402^2)의 기억장치를 도시하고 있다. 희박 서브트리 디스크립터(1400)에 저장된 노드 디스크립터(1402^1, 1402^2)가 존재하기 때문에, 노드 디스크립터 (1402^1 ,1402^2)는 1로 시작되는 모드 필드(1404)를 구비하는 서브 엔트리(404)에 저장된다. 서브트리를 위한 가장 긴 정합은, r1이 3분의 1비트의 정합을 요구하고, r2는 2분의 1비트의 정합을 요구하기 때문에, r1이다. 노드 디스크립터 (1402^1,1402^2)는, 첫번째로 저장된 r2에 대한 노드 디스크립터(14021)와 그 다음으로 저장된 r1에 대한 노드 디스크립터(14022)와 더불어, 희박 서브트리 디스크립터(1400)내에 최장의 정합 순서(match order)에 대해 최단으로 저장된다.

도 19c는 노드 디스크립터(1402^2)가 8비트로 작업된 값으로의 변환되는 것을 도시하고 있다. 왼쪽에서 오른쪽으로 노드 디스크립터 비트(1708^1-1708^9)를 보면, 처음의'1'은 비트(17086)내에 저장된다. 이것은 8비트로 작업된 값에 대한 마스크 비트의 마지막을 표시한다. 노드 디스크립터(1402^2)를 8비트로 작업된 값으로 변환하기 위하여, 다음의 비트 변환이 수행되었다. 노드 디스크립터 비트(1708^1)에 저장된 '0'은 'X'로 변환되고, 8비트로 작업된 값 비트(1710^1)에 저장된다. 노드 디스크립터 비트(1708^2)에 저장된 '0'은 'X'로 변환되고, 8비트로 작업된 값 비트(1710^2)에 저장된다. 노드 디스크립터 비트(1708^3)에 저장된 '0'은 'X'로 변환되고, 8비트로 작업된 값 비트(1710^3)에 저장된다. 노드 디스크립터 비트(1708^4)에 저장된 '0'은 'X'로 변환되고, 8비트로 작업된 값 비트(1710^4)에 저장된다. 노드 디스크립터 비트(1708^5)에 저장된 '0'은 'X'로 변환되고, 8비트로 작업된 값 비트(1710^5)에 저장된다. 노드 스크립터 비트(1708^6)에 저장된 '1'은 무시된다. 노드 디스크립터 비트(1708^7)에 저장된 '0'은 8비트로 작업된 값 비트(1710^6)에 저장된다. 노드 디스크립터 비트(1708^8)에 저장된 '1'은 8비트로 작업된 값 비트(1710^7)에 저장된다. 노드 디스크립터 비트(1708^9)에 저장된 '0'은 8비트로 작업된 값 비트(1710^8)에 저장된다.

도 19d는 CAM(1606)내의 노드 디스크립터(1402^1,1402^2)의 저장과 선택된 희박 서브트리 디스크립터(1400)에 대한 서브트리 매퍼(418)(도 5)에 저장된 그에 상응하는 매퍼 엔트리(504^1-504^3)를 도시하고 있다. 선택된 서브트리 디스크립터(1400)에 저장된 9비트 노드 디스크립터(1402^1,1402^2)는 변환 로직(1604)(도 18)으로 변환되고, CAM(1606)에 의해 재생된다. CAM(1606)내의 최초 엔트리(1702)는 도 19a의 서브트리(1700)안에 도시된 r0에 대한 디폴트 엔트리이다. 두 번째 엔트리(1704)는 선택된 희박 서브트리 디스크립터(1400)에 저장된 첫번째 노드 디스크립터(1402^2)로부터 변환된다. 두 번째 엔트리(1704)는 r2로 변환되는 최단의 정합이다. 선택된 서브트리 디스크립터(1400)에 저장된 두 번째 노드 디스크립터(1402^2)는 010100000에서 010XXXXX으로 변환되고 CAM(1606)내의 세 번째 엔트리(1706)에 저장된다.

CAM(1606)의 검색은 블록 오프셋(714)(도 18)에 귀결된다. 블록 오프셋(714)은 서브트리 매퍼(418)(도 5)에 저장된 매퍼 엔트리(504^1-504^3)에 대한 매퍼 어드레스(416)를 결정하기 위해 사용된다. CAM(1606)은 가장 긴 정합을 저장하는 엔트리(1702,1704,1706)에 대한 키(210b)의 두 번째 부분으로 검색된다. CAM(1606)에 의해 제공되는 블록 오프셋(714)은 선택된 서브트리 엔트리(404)내의 포인터 필드내에 저장된 블록 베이스 어드레스(600^1,600^1)의 하나에 의존하는 서브트리 베이스 어드레스와 결합된다.

도 20은 도 8에 도시된 포인터 로직(702)내의 희박 모드 베이스 셀렉트 로직(1800)을 나타내는 블록도이다. 포인터 로직(702)은 서브트리 매퍼(418)(도 5)내의 매퍼 엔트리(504)(도 6B)에 대한 매퍼 어드레스(416)를 계산하기 위해 사용된 베이스 어드레스(716)를 선택한다. 포인터 로직(702)은 밀집 모드 베이스 셀렉트 로직(710)과 희박 모드 베이스 셀렉트 로직(1800)을 포함한다. 둘 중의 하나는 이전의 매퍼 레벨로 형성된 서브트리 엔트리 디스크립터(304)(도 4)에 저장된 타입(1406)의 상태에 따라 선택된다. 전술했듯이, 타입(1406)의 상태는 서브트리 엔트리(404)가 밀집 모드로 설정될 것인지의 여부를 나타낸다.

만약 서브트리 엔트리(404)가 대부분의 희박 서브트리 디스크립터(1400)를 저장한다면, 희박 모드 베이스 선택 로직(1800)은 희박 서브트리 디스크립터 (1400)에 대한 베이스 어드레스(716)를 계산한다. 희박 모드 베이스 셀렉트 로직(1800)은 모드 필드(1404)에 저장된 모드값(1608)을 사용하는 베이스 어드레스(716)와 서브트리 엔트리(404)안의 블록 베이스 어드레스 필드(600^1, 600^2)에 저장된 서브트리 포인터(414), 그리고 이전의 매퍼 레벨로부터 발생된 서브트리 엔트리 디스크립터(304)(도 4)에 저장된 희박 서브트리 디스크립터 셀렉트(1408)를 계산한다. 베이스 어드레스(716)는 다음과 같이 계산된다.

수학식 1

베이스 어드레스 (희박 서브트리 디스크립터의 경우)

= 블록 베이스 어드레스 + 베이스 오프셋

여기서, 베이스 오프셋 = ((1+ 노드/부분트리)*희박 서브트리 디스크립터 셀렉트)

예를 들면, 희소 모드(4)내에 형성된 서브트리 엔트리(404)에서 서브트리 번호 2의 시작인 베이스 어드레스(716)를 찾기 위해 베이스 오프셋이 먼저 계산된다. 서브트리 번호 2에 대한 희박 서브트리 디스크립터(1408) 셀렉트는'1'이고 노드/서브트리의 수는 7이다(표 2 참조). 베이스 오프셋은 8이다((1+ 7)*1). 각각의 블록 베이스 어드레스(600^1, 600^2)는 서브트리 엔트리(404)에 할당된 16개의 매퍼 어드레스의 블록에 대한 베이스 어드레스이다. 서브트리 번호 2에 대한 베이스 오프셋은 16보다 작은 8이며, 따라서 서브트리 2에 대한 블록 베이스 어드레스(600^1)는 블록 베이스 어드레스(600^1)이고 희박 서브트리 디스크립터에 대한 베이스 어드레스(716)는 블록 베이스 어드레스(600^1) + 8이다. 아래의 표 5는 모드(4)에 형성된 서브트리 엔트리(404)내의 4개 서브트리 각각에 대한 서브트리 베이스 어드레스를 보여준다.

[표 5]

서브트리 베이스 어드레스	서브트리
블록 베이스 어드레스 1+0	1
블록 베이스 어드레스 2+8	2

블록 베이스 어드레스 2+0	3
블록 베이스 어드레스 2+8	4

도 21은 서브트리 메모리(400)에 저장된 밀집 서브트리 디스크립터와 희박 서브트리 디스크립터를 도시하고 있다. 도 21은 도 15와 결합하여 설명된다. 서브트리 B(도 21)에 대한 밀집 서브트리 디스크립터 서브트리 엔트리(404^1)는 서브트리(404^1)내의 데이타 필드(406^1)에 저장된다. 서브트리 A(도 21)에 대한 희박 서브트리 디스크립터(1400^1)는 서브트리 엔트리(404^2)내의 데이타 필드(406^2)에 저장된다. 밀집 서브트리 디스크립터는 이미 도 6b와 결합하여 도시된 것처럼, 서브트리 B의 최저 레벨에서 각각의 노드에 대한 노드 비트를 저장한다. 희박 모드 디스크립터(1400^1)는 도 19b와 결합하여 도시된 것처럼 경로 r4와 r5에 대응하는 노드 디스크립터(1402^1,1402^2)를 포함한다. 서브트리 인덱스(312)는 서브트리 엔트리 (404^1, 404^2)를 선택한다.

s0에 대한 매퍼(106a)의 매퍼 엔트리(504)(도 6b)의 서브트리 엔트리(304)(도 4)에 저장된 서브트리 인덱스(312)는 서브트리 엔트리(404^1)를 선택한다. s1에 대한 매퍼(106)에서 매퍼 엔트리(504)(도 6b)의 서브트리 엔트리(304)(도 4)에 저장된 서브트리 인덱스(312)는 서브트리 엔트리(404^2)를 선택한다. 그리하여, 서브트리 메모리(400)는 희박 서브트리와 밀집 서브트리에 대한 서브트리 엔트리(404^1, 404^2)를 저장할 수 있다.

도 22는 희박 서브트리와 밀집 서브트리내의 노드에 대한 경로를 저장하는 서브트리 매퍼(418)(도 5)의 매퍼 엔트리(504)(도 6b)에 대한 매퍼 어드레스 (416)(도 5)를 제공하는 방법을 설명하는 흐름도이다. 어느 서브트리 엔트리(404)는 다수개의 희박 서브트리 디스크립터 또는 하나의 밀집 서브트리 디스크립터를 저장할 수 있다. 희박 서브트리 디스크립터와 밀집 서브트리 디스크립터의 임의의 조합은 경로가 두개의 트리내에 분포되는 방법에 의존한다. 서브트리 메모리(400)내의 서브트리 엔트리(404)에서 희박 모드와 밀집 서브트리 디스크립터를 혼합하고 정합하는 가능성은 서브트리 메모리(400)를 보다 다양하게 이용가능하게 한다.

단계(1900)에서, 선택된 서브트리 엔트리(404)의 형상은 이전의 매퍼 레벨에서 선택된 서브트리 엔트리 디스크립터(304)(도 4)에 저장된 타입(1406)(도 16c)의 상태에서 결정된다. 만약 서브트리 엔트리(404)타입이 희박 모드로 설정되면 프로세싱은 단계(1902)로 계속된다. 그렇지 않다면 단계(1914)로 계속 처리된다.

단계(1902)에서, 서브트리 엔트리(404)는 희박 모드로 설정된다. 서브 엔트리(404)내에 저장된 희박 서브트리 디스크립터(1400)의 수는 모드 필드(1404)의 상태에 의존한다. 오프셋 로직(700)내에 희박 모드 로직(1502)은 도 14와 결합하여 전술한 이전의 매퍼 레벨과 모드 필드(1404)로부터 형성된 서브트리 엔트리 디스크립터(304)(도 4)에 저장된 희박 서브트리 디스크립터 셀렉트(1408)에 의존하는 서브트리 엔트리(404)로부터 희박 서브트리 디스크립터(1400)를 선택한다. 프로세싱은 단계(1904)로 계속된다.

단계(1904)에서, 선택된 희박 서브트리 디스크립터(1400)내의 노드 디스크립터(1402)에 저장된 9비트 코트값은 8비트 값으로 변환되고, CAM(1606)내에서 최단에서 최장의 정합순으로 저장된다.

단계(1906)에서, CAM(1606)은 최장정합을 저장하는 CAM 엔트리를 위한 키 (210b)의 다음 부분과 함께 검색된다. 프로세싱은 단계(1908)로 계속된다.

단계(1908)에서, 키(210b)의 다음 부분을 위한 최장정합을 저장하는 CAM (1606)에서 어드레스의 로케이션은 블록 오프셋(714)으로 형성된다. 블록 오프셋(714)은 서브트리 매퍼(418)(도 5)내의 매퍼 엔트리(504)의 매퍼 어드레스(416)(도 5)를 계산하는 데 사용된다. 프로세싱은 단계(1910)로 계속된다.

단계(1910)에서, 선택된 희박 서브트리 디스크립터(1400)를 위한 베이스 어드레스(716)는 이전의 매퍼 레벨로부터 형성된 서브트리 엔트리 디스크립터(304)(도 4)내에 저장된 희박 서브트리 디스크립터 셀렉트(1408)와 선택된 서브트리 엔트리(404)내에 저장된 모드 필드(1404)의 내용에 의존하여 계산된다. 프로세싱은 단계(1912)로 계속된다.

단계(1912)에서, 매퍼 어드레스(416)는 블록 오프셋(714)과 어드레스 로직 (704)(도 8)내의 베이스 어드레스(716)를 더함으로써 계산된다. 매퍼 엔트리(504)(도 6b)는 서브트리 매퍼(418)(도 5)내의 경로 엔트리(302)나 서브트리 엔트리 디스

크립터(304)를 저장하는 서브트리 매퍼(418)(도 5)내의 매퍼 어드레스(416)에 의해 인식된다. 만약 매퍼 엔트리(504)(도 6b)가 경로 엔트리(302)를 저장한다면, 검색은 종료된다. 만약 매퍼 엔트리(504)(도 6b)가 서브트리 엔트리 디스크립터(304)(도 4)를 저장한다면, 키(210)에 상응하는 값을 위한 검색은 다음 매퍼 레벨로 계속된다.

단계(1914)에서, 서브트리 엔트리(404)는 밀집 모드에서 형성되고, 데이타 필드(406)내의 단순 밀집 서브트리 디스크립터를 저장한다. 블록 오프셋(714)은 도 6b와 결합하여 전술한 바와 같이, 서브트리 엔트리(404)내의 데이타 필드(406)내에 저장된 밀집 서브트리 디스크립터에 저장된 '1'의 수를 카운팅하여 계산된다. 프로세싱은 단계(1916)로 계속된다.

단계(1916)에서, 서브트리 엔트리(404)는 그 내의 포인터 필드(408)내에 16개의 블록 베이스 어드레스(600)를 저장한다. 블록 베이스 포인터(600)의 하나는 도 8과 결합하여 전술된 포인터 로직(702)내의 밀집 모드 베이스 셀렉트 로직 (710)에 의해서 선택된다. 프로세싱은 단계(1912)로 계속된다.

도 23은 검색표(200)에 추가된 새로운 경로의 두개의 트리를 도시하고 있다. 두개의 트리는 매퍼 레벨1(2000)과 매퍼 레벨2(2002)와 매퍼 레벨3(2004)에 대한 검색표(200)에 저장된 경로를 나타낸다. 매퍼 레벨2(2002)는 서브트리 A와 B에 대한 경로를 저장한다. 매퍼 레벨3(2004)는 서브트리 A_1,A_2,B_1 및 B_2 에 대한 경로들을 저장한다. s5는 서브트리 매퍼(418)(도 5)에 저장된 서브트리 엔트리 디스크립터(304)(도 4)를 나타낸다. s5에 대한 서브트리 엔트리 디스크립터(304)(도 4)는 키(210)에 대한 최장 정합 경로의 검색이 레벨3(2004)내에서 계속되게 하는 서브트리 B_2 에 포인터를 저장한다. 서브트리 B_2(2006)는 단지 r6과 h1의 두 개의 경로만 가지고 있기 때문에 희박 서브트리이다. 따라서, 도 14a와 결합하여 전술한 바와 같이, 노드 r6과 h1를 위한 노드 디스크립터(1402)(도 2)는 희박 서브트리 디스크립터 (1400)에 저장된다. 희박 서브트리 디스크립터(1400)에 저장된 두 개의 노드 디스크립터(1402)가 존재하기 때문에, 서브트리 B_2(2006)에 대한 희박 서브트리 디스크립터(1400)는 서브트리 메모리(400)내의 '1'로 시작된 노드 필드 (1404)와 함께 서브트리 엔트리(404)에 저장된다. 서브트리 B_2`(2008) 에서 도시된 새로운 경로 h2는 검색표(200)에 추가된다. 서브트리 B_2(2006)에 경로를 추가하는 것은 2에서 3까지 희박 서브트리 디스크립터에 저장된 노드 디스크립터 (1402)의 개수를 증가시키기 때문에, 새로운 경로 h1은 검색표에서 서브트리 B_2(2006)에 즉시 추가될 수 없다. 희박 서브트리 디스크립터(1400)에 노드 디스크립터(1402)를 추가하는 것은'2'로 시작된 노드 필드(1404)와 함께 서브트리 엔트리(404)내의 새로운 희박 서브트리 디스크립터(1400)의 할당을 요구한다. 따라서, 새로운 경로 h1의 추가는 서브트리 B_2(2006)를 서브트리 B_2`(2008)로 교체하는 것이 필요하다.

도 24는 프로세서 메모리(2400)에 저장된 개정 경로들을 설명한다. 또한 검색표(200)에 저장된 두개 트리의 복사는 검색표(200)에서 분리된 프로세서 메모리(2400)내에 저장된다. 서브트리 B_2(2006)를 위해 저장된 경로는 프로세서 메모리(2400)내에 서브트리 B_2`에 복사되고 새로운 경로 h2는 서브트리 $B_2$`(2008)에 추가된다.

루트 업데이트 경로(route update routine)(2402)는 검색표(200)에 서브트리 B_2(2008)를 추가하기 위해 연속된 루트 업데이트 지시(2404)를 하며, 표 업데이트 경로(2406)에 루트 업데이트(2404)를 진행시킨다. 표 업데이트 경로(2406)는 루트 업데이트(2402)를 위해 표 업데이트(2410)를 발생시키고 루트 업데이트(2404)와 함께 검색표(200)를 업데이트하기 위해 업데이트 싸이클(2412)을 진행시킨다.

업데이트 사이클(2412)은 서브트리 메모리(400)(도 5)와 서브트리 매퍼(418)(도5)의 바람직한 메모리 로케이션에 업데이트 경로를 기록한다.

도 23을 다시 참고하면, 업데이트 싸이클(2412)은 매퍼 엔트리(504)(도 6b)내의 새로운 서브트리B_2`(2008)를 위한 경로를 저장하기 위해 서브트리 매퍼(418)(도5)의 일부를 할당하는 지시를 포함하고 있다. 서브트리 $B_2$`(2008)는 매퍼엔트리(504)(도6b)에 저장된 h1,h6 및 새로운 경로 h2의 경로 엔트리들을 포함하고 있다. 서브트리 B_2`(2008)를 위한 경로 엔트리가 서브트리 매퍼(418)(도5)의 매퍼 엔트리(504)(도6B)내에 저장된 후, 경로를 위한 노드 디스크립터(1402)가 만들어지며 희박 서브트리 디스크립터(1400)내에 저장된다. 희박 서브트리 디스크립터(1400)는 서브트리 엔트리(404)에 저장된다. 서브트리 엔트리(404)의 모드(1404)는 희박 서브트리 디스크립터(1400)내에 저장된 노드 디스크립터(1402)의 수와 관계된다.

서브트리 B_2`(2008)를 위한 희박 디스크립터(1400)가 검색표(200)의 서브트리 메모리(400)에 있는 서브트리 엔트리(404)내에 저장된 후, s5로 표시되는 서브트리 엔트리 디스크립터(304)(도4)는 서브트리 $B_2$(2006) 대신 서브트리 $B_2$`(2008)를 지시하도록 변경된다. 서브트리 B_2`(2008)가 검색표에 추가되는 동안, 서브트리 B2(2006)내에 저장 된 경로 r6와 h1은 s5를 통해 접근될 수 있다. 서브트리 $B_2$`(2008)가 검색표에 저장된 후에, s5가 서브트리 B_2`(2008)과 경로 r6를 지시하도록 변경된 후에 경로h1과 새로운 경로h2에 접근될 수 있다. 따라서 새로운 경로 h2가 검색표(200)에 추가되는 동안 서브트리 B_2(2006)은 경로 r6과 h1에 상응하는 경로 인덱스들에 대해 계속적으로 검색이 가능해진다.

도 25는 검색표(200) 내의 서브트리 매퍼(418b)의 매퍼 엔트리($504c^4$)에 저장되며, 도 23에 도시된 새로운 경로 h2를 도시하고 있다. 도 25는 도 24에 도시된 두개의 트리와 결합하여 도시된다.

매퍼 레벨2(2002)내의 서브트리 B는 r3, s4 및 s5의 세 경로를 가지고 있다. 서브트리B는 16개 이하의 경로를 갖고 있으므로 희박 서브트리이다. 서브트리 B r3, s4 및 s5에 대한 노드 디스크립터($1402a^1$-$1402a^3$)는 서브트리 메모리(400a)내의 서브트리 엔트리(404a)에 속해있는 희박 서브트리 디스크립터($1400a^1$)내에 저장된다. 매퍼 엔트리($504a^2$)-($504a^4$)는 서브트리 B의 각 경로에 대한 서브트리 매퍼 418a에 저장된다. 각각의 매퍼 엔트리($504a^2$-$504a^4$)는 노드에 대한 경로 엔트리 302(도 4)와 엔트리 디스크립터(304)(도4)를 저장한다. 서브트리 엔트리 디스크립터(304)(도4)는 매퍼 엔트리($504a^3$, $504a^4$)내의 경로 s4, s5를 위해 저장된다. s5에 대한 매퍼 엔트리 $504a^4$내에 저장된 서브트리 엔트리 디스크립터(304)(도4)는 다음 단계의 검색 즉, 매퍼 레벨3(2004)을 시작하기 위한 서브트리 메모리(400b)에 대한 서브트리 인덱스(312b)를 제공한다.

서브트리 B_2는 경로 h1와 r6의 두 경로를 갖고 있으므로 서브트리 B_2 또한 희박 서브트리이다. 노드 디스크립터($1402b^1$-$1402b^2$)는 서브트리 메모리(400b)내의 서브트리 엔트리(404b)에 속해 있는 희박 서브트리 디스크립터($1400b^1$)에 저장되어 있다. 서브트리 B_2내의 각 경로는 매퍼 엔트리($504b^2$-$504b^3$)에, 그리고 서브트리 B_2의 디폴트 경로는 매퍼 엔트리($504b_1$)에 저장된다.

서브트리 B_2(2006)내의 경로 h1을 검색하기 위해, 경로 s5에 대한 노드 디스크립터(1402)를 저장하는 희박 서브트리 디스크립터(1400a)를 저장하고 있는 서브트리 엔트리(404a)의 어드레스는 서브트리 메모리(400a)보다 서브트리 인덱스(312a)에 앞선다. 선택된 서브트리(404a)에 저장된 데이타 필드(406)와 포인터 필드(408)는 매퍼 어드레스 로직(402a)보다 서브트리 데이타(412a)와 서브트리 포인터(414a)에 앞선다. 매퍼 어드레스 로직(402a)는 s5에 대한 서브트리 엔트리를 저장하는 매퍼 엔트리($504a^4$)에 대한 매퍼 어드레스(416a)를 발생시킨다. 매퍼 어드레스(416a)는 서브트리 데이타(412a), 서브트리 포인터(414a) 및 키(210b)의 다음 부분에 의존한다. s5에 대한 서브트리 엔트리는 서브트리 메모리(400b)보다 서브트리 인덱스(312b)에 앞선다.

서브트리 메모리(400b)는 서브트리 B_2(2006)에 대한 노드 디스크립터 ($1402b^1$, $1402b^2$)를 저장한다. B_2에 대한 희박 서브트리 디스크립터($1400b^1$)는 서브트리 엔트리 404b)에 저장된다. 서브트리 엔트리(404b)에 저장된 데이타 필드 (406)와 포인터 필드(408)는 매퍼 어드레스 로직(402b)보다 서브트리 데이타(412b)와 서브트리 포인터(414b)에 앞선다. 매퍼 어드레스 로직(402b)는 h1에 대한 경로 엔트리를 저장하고 있는 매퍼 엔트리($504b^3$)에 대한 매퍼 어드레스(416b)를 발생시킨다. 매퍼 어드레스(416b)는 서브트리 데이타(412b)와 서브트리 포인터(414b) 및 키(210c)의 다음 부분에 의존한다.

서브트리 B_2(2006)에 경로 h2를 더하기 위해, 서브트리 매퍼(418b)내의 이전에 사용되지 않은 매퍼 엔트리(602c)의 블록은 서브트리 B_2`(2008)에 대한 경로 r6, h1 및 h2를 저장한 매퍼 엔트리($504c^2$-$504c^4$)를 저장하기 위해 할당된다. 매퍼 엔트리($504c^1$)는 서브트리 $B_2$`(2008)에 대한 디폴트 엔트리를 저장한다. 즉 매퍼 엔트리($504b^1$)에 저장된 동일한 값이다. 매퍼 엔트리($504c^2$)는 경로 r6에 대한 경로 엔트리를 저장한다. 그리고 그 값은 매퍼 엔트리 ($504b^2$)에 저장된 값과 같다. 매퍼 엔트리($504c^3$)는 h1경로에 대한 경로 엔트리를 저장하고 그 값은 매퍼 엔트리($504b^3$)에 저장된 값과 같다. 매퍼

엔트리(504c^4)는 새로운 경로 h2에 대한 경로 엔트리를 저장한다. 매퍼 엔트리(504c$^{1-4}$)의 블록이 기록되는 동안, 매퍼 엔트리(504b^1-504b^3)내에 저장된 경로 엔트리는 서브트리 매퍼(418a)의 엔트리(504a^4 내의 경로 s5에 저장된 서브트리 엔트리를 통해 접근될 수 있다.

서브트리 매퍼(418b)내에 서브트리B_2`에 대한 매퍼 엔트리(504c$^{1-4}$)를 저장하는 동안, 희박 서브트리 디스크립터(1400c^1)는 서브트리 메모리(400b)에 더해진다. 노드 디스크립터(1402c$^{1-3}$)의 수는 16보다 작으므로, 노드 디스크립터(1402c$^{1-3}$)는 희박 서브트리 디스크립터(1400c^1)에 저장된다. 서브트리 메모리(400b)내의 서브트리 디스크립터(1400^1)의 로케이션은 희박 서브트리 디스크립터(1400c^1)와 관련되는 노드 디스크립터(1402c$^{1-3}$)의 수에 의존한다. 새로운 경로를 서브트리B_2(2006)에 더함으로써, 희박 서브트리 디스크립터(1400c^1)에 저장된 노드 디스크립터(1402c^1-1402c^3) 수는 2에서 3으로 증가한다. 희박 서브트리 디스크립터(1400c^1)는 희박 서브트리 디스크립터당 3개의 노드 디스크립터와 2로 시작된 모드 필드(1404)와 함께 서브트리 엔트리(404c)에 저장된다. 희박 서브트리 디스크립터(1400c^1)는 공간이 충분하거나 새로운 모드(3) 서브트리 엔트리가 할당되어 있다면 서브트리 엔트리(404c) 현재의 모드(3)에 저장된다. B_2'(2008)내의 경로를 위한 노드 디스크립터는 모드(3)서브트리 엔트리(404c)내의 희박 서브트리 디스크립터(1400c^1)의 노드 디스크립터(1402c$^{1-3}$)에 저장된다.

희박 서브트리 디스크립터(1400c^1)와 노드 디스크립터(1402c$^{1-3}$)가 서브트리 메모리(400b)에 저장되면 서브트리B$^{2'}$(2008)는 접근 가능하다. 서브트리 B$^{2'}$(2008)로 접근을 제공하기 위해, 서브트리 엔트리(504a^4)는 서브트리 엔트리(404b)내의 희박 서브트리 디스크립터(1400b^1) 대신에 서브트리 엔트리(404c)내의 인덱스 희박 서브트리 디스크립터(1400c^1)로 변경된다. 매퍼 엔트리(504c^4)에 저장된 경로 h2와 경로 r5 및 각각의 매퍼 엔트리(504c^2, 504c^3)에 저장된 h1에 대한 경로 엔트리에 접근할 수 있게 된다.

매퍼 엔트리(504b^1-504b^3)는 더 이상 접근 및 할당될 수 없고, 장래의 할당을 위해 프리 리스트(미도시됨)에 위치하게 된다. 또한 희박 서브트리 디스크립터 (1400b^1)도 더 이상 접근할 수 없게 된다. 따라서 희박 서브트리 디스크립터 (1400b^1)는 할당되지 않고 장래의 할당을 위해 프리 리스트(미도시됨)에 놓여진다.

희박 서브트리에 경로를 추가하는 것은 서술되었다. 경로는 또한 새롭게 할당된 서브트리 엔트리(404)와 대응되는 서브트리 매퍼(418)내의 서브트리 디스크립터의 새로운 밀집 서브트리 디스크립터를 저장함으로써 밀집 서브트리에 추가할 수 있으며, 매퍼 엔트리(504a^4)에 저장된 서브트리 엔트리를 새롭게 할당된 서브트리 엔트리(404) 인덱스로 변경하여 밀집 서브트리에 추가될 수 있다.

도 26은 도 25에 도시된 검색표(200)에 경로를 추가하기 위한 증가 개정을 수행하기 위한 단계를 도시하는 흐름도이다.

단계(2200)에서, 서브트리당 경로의 수는 루트 업데이트가 희박 서브트리가 되는지 또는 밀집 서브트리가 되는지를 결정하기 위해 계산된다. 서브트리가 경로 계정후에 밀집해지면 과정은 단계(2218)로 이어진다. 서브트리가 경로 계정후에 희박해지면 과정은 단계(2202)로 이어진다.

단계(2202)에서, 서브트리는 희박이다. 희박 서브트리 모드는 결정된다. 과정은 단계(2204)로 이어진다.

단계(2204)에서, 서브트리 매퍼(418)(도 5)내에 저장된 부분적으로 채워진 서브트리 엔트리(404)의 목록은 새로운 희박 서브트리 디스크립터(1400c1)가 미리 할당된 서브트리 엔트리(404)에 저장될 수 있는지를 결정하기 위해 검색된다. 예를 들어, 4개의 희박 서브트리 디스크립터(1400c^1-1400c^4)는 모드(4)서브트리 엔트리 (404)에 저장될 수 있다. 오직 3개의 디스크립터가 저장되었다면 서브트리 엔트리 (404)는 부분적으로 채워졌고 부분적으로 채워진 서브트리 엔트리(404)의 리스트에 저장된다. 부분적으로 채워진 서브트리 엔트리(404)가 이용될 수 있다면, 과정은 단계(2208)로 이어지며, 그렇지 않으면 과정은 단계(2206)로 이어진다.

단계(2206)에서, 새로운 서브트리 엔트리(404c)는 희박 서브트리 디스크립터($1400c^1$)를 저장하기 위해 할당되며, 매퍼 엔트리($504c^1$-$504c^4$)는 새롭게 할당된 서브트리 엔트리(404c)내 희박 서브트리 디스크립터($1400c^1$)에 저장된 노드 디스크립터($1402c^1$-$1402c^3$)를 위한 매퍼 엔트리(504)(도 6b)를 저장하기 위해 서브트리 매퍼에 할당된다. 과정은 단계(2208)로 계속한다.

단계(2208)에서, 희박 서브트리 디스크립터($1400c^1$)를 위한 서브트리 매퍼에서의 첫번째 매퍼 엔트리($504c^1$)의 로케이션은 서브트리 엔트리(404c)내의 포인터 필드(408)에 저장된 포인터와 서브트리 엔트리(404c)에서 모드 필드(1404)에 저장된 모드로부터 결정된다. 과정은 단계(2210)로 계속된다.

단계(2210)에서 희박 서브트리를 위한 경로 엔트리는 서브트리 매퍼(418b)내의 매퍼 엔트리($540c^1$-$504c^4$)에 저장된다. 과정은 단계(2212)로 계속된다.

단계(2212)에서, 노드 디스크립터($1402c^1$-$1402c^2$)를 저장하는 서브트리 디스크립터($1400c^1$)는 서브트리 엔트리(404C)내에 저장된다. 과정은 단계(2214)로 이어진다.

단계(2214)에서, 매퍼 엔트리($504a^4$)내에 저장된 서브트리 엔트리 디스크립터(304)(도 4)는 서브트리 엔트리(404c)내에 저장된 새로운 희박 서브트리 디스크립터($1400c^1$)를 인덱스 하기 위해 변경된다. 매퍼 엔트리($504c^4$)내에 저장된 h2에 대한 경로 엔트리는 이제 접근될 수 있다. 과정은 단계(2216)로 이어진다.

단계(2216)에서, 매퍼 엔트리($504b^1$- $504b^3$)와 희박 서브트리 디스크립터 (1400b)에 더 이상 접근될 수 없다. 매퍼 엔트리($504b^1$-$504b^3$)는 서브트리 매퍼(418b)를 위한 매퍼 엔트리(504)(그림 6B)의 프리 리스트상에 위치되고, 다른 경로를 저장하기 위해 할당될 수 있다. 서브트리 엔트리(404b)내의 가장 이용가능한 로케이션은 부분적으로 채워진 서브 엔트리의 리스트에서 계정된다.

단계(2218)에서, 새로운 서브트리 엔트리(404)는 프로세서 메모리(2400)(도 24)내에 저장된 프리 서브 엔트리(404)의 리스트에서 할당된다. 새로운 서브트리 엔트리(404)는 새로운 밀집 서브트리 디스크립터를 저장하기 위해 할당된다. 서브트리 매퍼(418b)내의 매퍼 엔트리(504)의 블록은 경로를 저장하기 위해 할당된다. 할당된 매퍼 엔트리(504)(도 6b)의 블록의 포인터는 서브트리 엔트리(404)(도 5)내의 포인터 필드(408)내에 저장된다. 과정은 단계(2220)로 이어진다.

단계(2220)에서, 새로운 서브트리 엔트리(404)는 도 6a 내지 도 6b와 결합하여 전술된 바와 같이 새로운 서브트리 엔트리(404)내의 데이타 필드(406)내에 기록된다. 과정은 단계(2222)로 이어진다.

단계(2222)에서, 밀집 서브트리를 위한 경로 엔트리는 서브트리 엔트리 (404)내의 포인터 필드(408)에 저장된 포인터에 의해 인식된 서브트리 매퍼(418)(도 5)내의 매퍼 엔트리(504)(도 6)에 저장된다. 과정은 단계(2224)로 이어진다.

단계(2224)에서, 매퍼 엔트리($504a^4$)에 저장된 서브트리 엔트리 디스크립터 (304)(도 4)는 새로운 서브트리 엔트리(404c)내에 저장된 새로운 밀집 서브트리 디스크립터를 인덱스하기 위해 변경된다. 매퍼 엔트리($504c^4$)에 저장된 h2에 대한 경로 엔트리는 이제 접근될 수 있다. 과정은 단계(2226)로 이어진다.

단계(2226)에서, 이전의 서브트리 엔트리(404)내 포인터 필드(408)에 저장된 포인터에 의해 인덱스된 매퍼 엔트리(504)(도 6b)는 프로세스 메모리(2400)(도 24)에 저장된 매퍼 엔트리의 프리 리스트에 복귀된다. 이전의 서브트리 엔트리(404b)는 프로세서 메모리(2400)(도 24)에 저장된 서브트리 엔트리의 프리 리스트에 추가된다.

프로세스는 경로 추가를 위해 검색표에 기록된다. 유사한 프로세서는 검색표로부터 경로를 제거하기 위해 실행된다. 예를 들면, 매퍼 엔트리($504c^4$)에 저장된 h2를 서브트리 B_2`로부터 제거하기 위해, 경로 r6와 h1에 대한 2개의 노드 디스크립터를 구비하는 새로운 희박 서브트리 디스크립터를 저장하는 것이 요구되며, 모드(2) 서브트리 엔트리 디스크립터내에 희

박 서브트리 디스크립터를 저장하는 것이 요구되며, 대응하는 서브트리 매퍼를 계정하는 것이 요구되며, 새로운 서브트리 엔트리(404)내에 저장된 계정 서브트리 엔트리를 인덱스하기 위한 매퍼 엔트리(504a[4])내에 저장된 서브트리 엔트리 디스크립터(304)(도 4)를 변경하는 것이 요구된다.

본 발명분야의 기술자는 본 발명의 사상 및 범주를 벗어나지 않고 본 발명의 다양한 변형 및 변경을 할 수 있음은 명백할 것이다. 따라서, 본 발명은 첨부된 청구범위 및 그 균등물의 범위 내에서 제공되는 본 발명의 변형 및 변경을 포함한다.

도면의 간단한 설명

도 1a는 2진트리의 제 1 단계를 나타내기 위한 종래의 비트 맵을 도시한 도면이고,

도 1b는 캐시 메모리에 구비된 종래의 검색표를 도시한 도면이며,

도 2a는 본 발명의 원리에 따른 최장의 정합 접두부 검색표를 도시한 도면이고,

도 2b는 도 2a에 도시된 검색표에 저장된 루트 인덱스의 2진트리 표현을 도시한 도면이며,

도 3은 본 발명의 원리에 따른 40비트 키에 대한 최장의 정합 접두부 검색표를 도시한 도면이고,

도 4는 도 2a에 도시된 직접 맵된 매퍼에 저장될 수 있는 매퍼 엔트리의 형식을 도시한 도면이며,

도 5는 도 2b에 도시된 매퍼 레벨 2(112b)의 노드에 대응하는 매퍼를 도시한 도면이고,

도 6a는 서브트리의 2진트리 표현이며,

도 6b는 도 6a에 도시된 서브트리의 하부 레벨의 노드에 대응하는 도 5에 도시된 데이타 필드에 저장된 밀집 서브트리 디스크립터를 도시한 도면이고,

도 7은 도 5에 도시된 ptr 필드를 도시한 도면이며,

도 8은 도 5에 도시된 매퍼 어드레스 로직을 도시한 도면이고,

도 9는 최장의 정합을 검색하기 위한 단계를 도시한 흐름도이며,

도 10a는 깊이 확장을 제공하기 위한 실시예를 도시한 도면이고,

도 10b는 도 10a에 도시된 실시예에서 검색표중 하나를 도시한 도면이며,

도 10c는 값의 저장에 이용될 수 있는 매퍼 엔트리의 수를 증대시키기 위해 깊이 확장을 제공하는 다른 실시예를 도시한 도면이고,

도 10d는 도 10c에 도시된 실시예의 슬레이브 검색표를 도시한 도면이며,

도 11a 및 도 11b는 도 10a 및 도 10c에 도시된 검색표의 매퍼 엔트리중 도 2b에 도시된 루트 인덱스의 2진트리 표현에서 노드 분배의 2진트리 표현을 도시한 도면이고,

도 12는 도 10a 및 도 10c에 도시된 검색표의 매퍼 엔트리중 값을 분배하기 위한 방법을 도시한 흐름도이며,

도 13은 도 10c에 도시된 검색표에서 매퍼 엔트리중 하나에 저장된 검색 키에 대응하는 값을 검색하는 방법을 도시한 흐름도이고,

도 14는 도 10a에 도시된 검색표에서 매퍼 엔트리중 하나에 저장된 검색 키에 대응하는 값을 검색하는 방법을 도시한 흐름도이며,

도 15는 제 1 매퍼 레벨에 의해 인덱스된 제 2 매퍼 레벨에서의 희박 서브트리와 밀집 서브트리의 2진트리 표현을 도시한 도면이고,

도 16a 내지 도 16c는 서브트리 엔트리에 다수개의 희박 서브트리 디스크립터를 저장할 수 있도록 하기 위한 도 4에 도시된 서브트리 엔트리와 도 5에 도시된 서브트리 엔트리의 데이타 필드와 포인터 필드의 변형을 도시한 도면이며,

도 17은 희박하게 배치된 서브트리에서 노드의 블록 오프셋을 선택하기 위해 도 8에 도시된 오프셋 로직의 희박 모드 서브트리 로직을 도시한 도면이고,

도 18은 도 17의 오프셋 로직에 도시된 희박 모드 로직을 도시한 도면이며,

도 19a 내지 도 19d는 희박하게 배치된 서브트리에서 노드에 대한 블록 오프셋의 선택을 도시한 도면이고,

도 20은 도 8에 도시된 포인터 로직에서 희박 모드 베이스 선택 로직을 도시한 블럭도이며,

도 21은 서브트리 메모리에 저장된 희박 서브트리 디스크립터와 밀집 서브트리 디스크립터를 도시한 도면이고,

도 22는 희박하게 배치된 서브트리와 밀집되게 배치된 서브트리에 노드의 루트를 저장하는 서브트리 매퍼에서 매퍼 엔트리에 매퍼 어드레스를 제공하기 위한 방법을 도시한 흐름도이며,

도 23은 검색표에 추가될 새로운 루트의 2진트리 표현을 도시한 도면이고,

도 24는 프로세서 메로리에 저장된 업데이트 루트를 도시한 도면이며,

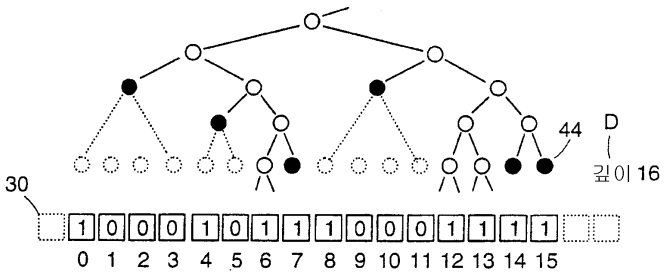
도 25는 검색표에 저장되는 도 23에 도시된 새로운 루트를 도시한 도면이고,

도 26은 도 25에 도시된 검색표에 새로운 루트를 추가하는 단계를 도시한 흐름도이다.

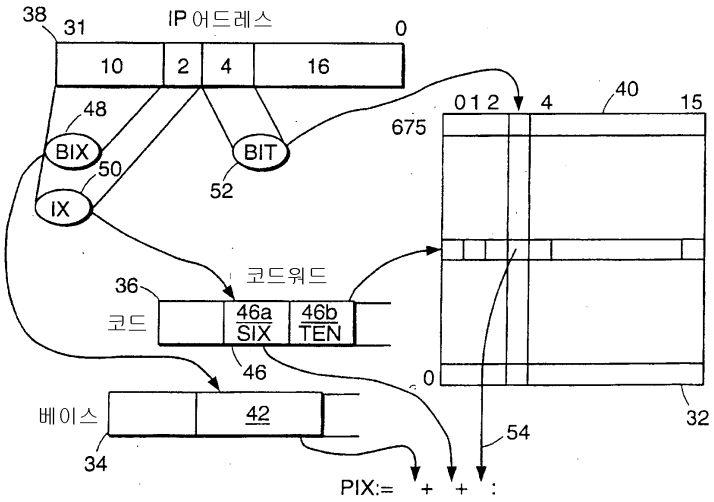
본 발명의 여타 목적, 특징 및 장점은 첨부도면에 도시된 본 발명의 바람직한 실시예에 대한 하기된 설명으로부터 이해할 수 있을 것이며, 도면에서 동일부분에는 유사한 참조번호를 부여하였다. 상기 도면은 척도에 따르지 않는 대신, 본 발명의 원리를 나타내기 위해 강조되었다.

도면

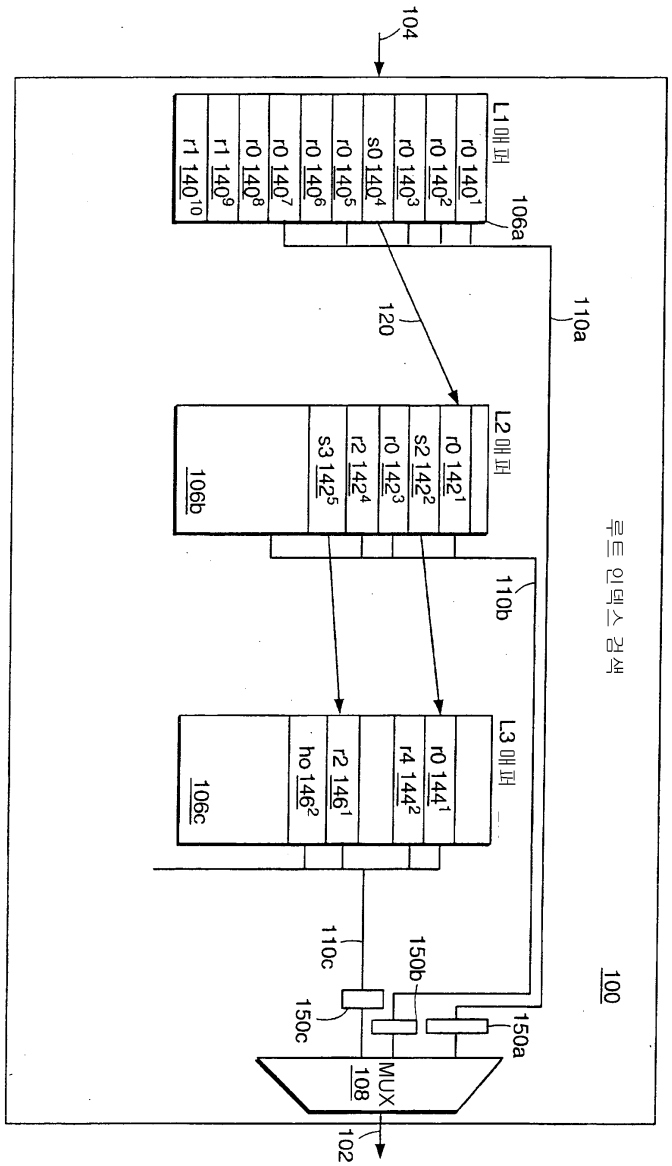
도면1a



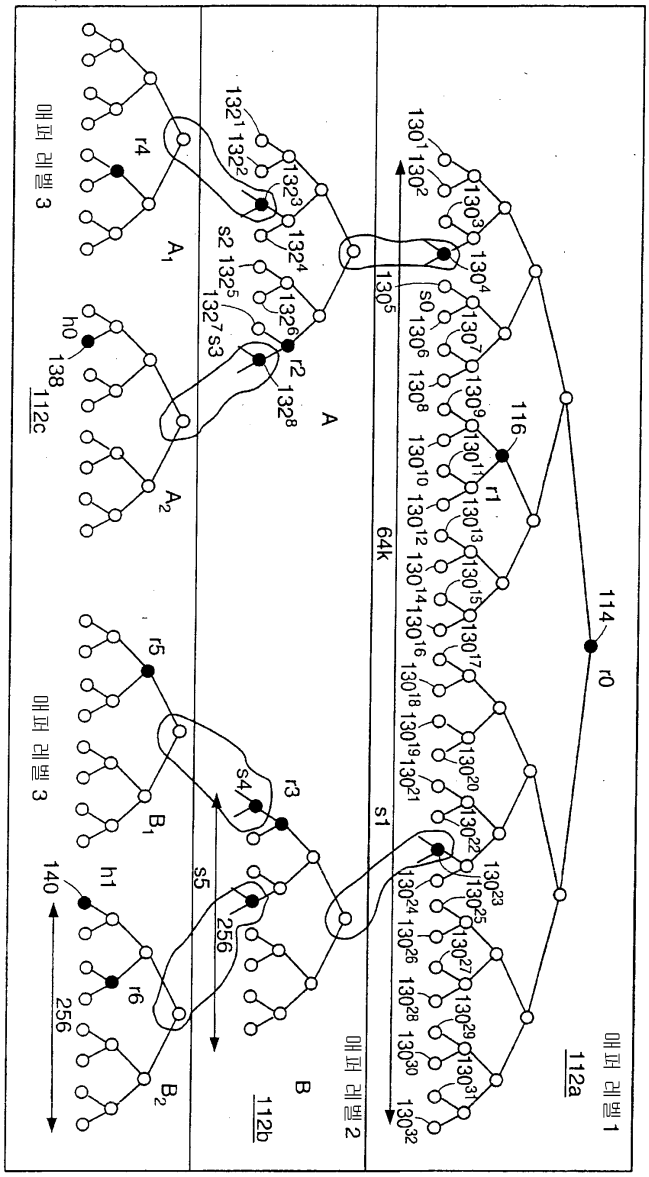
도면1b


38
31
IP 어드레스
0
10
2
4
16
48
BIX
50
IX
BIT
52
0 1 2
4
40
15
675
코드워드
36
코드
46a
SIX
46b
TEN
46
0
32
베이스
42
34
54
PIX:=
+
+
:

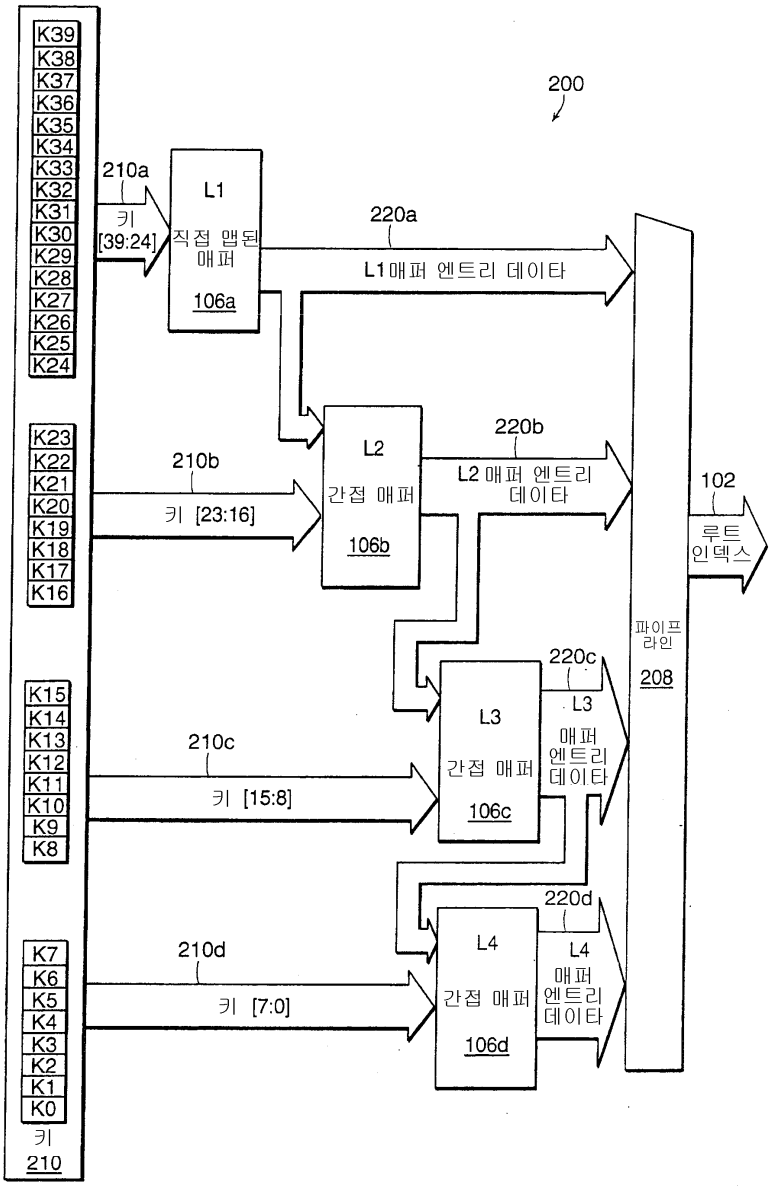
도면2a


104
L1 매퍼
r0 140¹
r0 140²
r0 140³
s0 140⁴
r0 140⁵
r0 140⁶
r0 140⁷
r0 140⁸
r1 140⁹
r1 140¹⁰
106a
110a
120
L2 매퍼
r0 142¹
s2 142²
r0 142³
r2 142⁴
s3 142⁵
106b
110b
룩트 인덱스 검색
L3 매퍼
r0 144¹
r4 144²
r2 146¹
ho 146²
106c
110c
150b
150c
150a
MUX 108
100
102

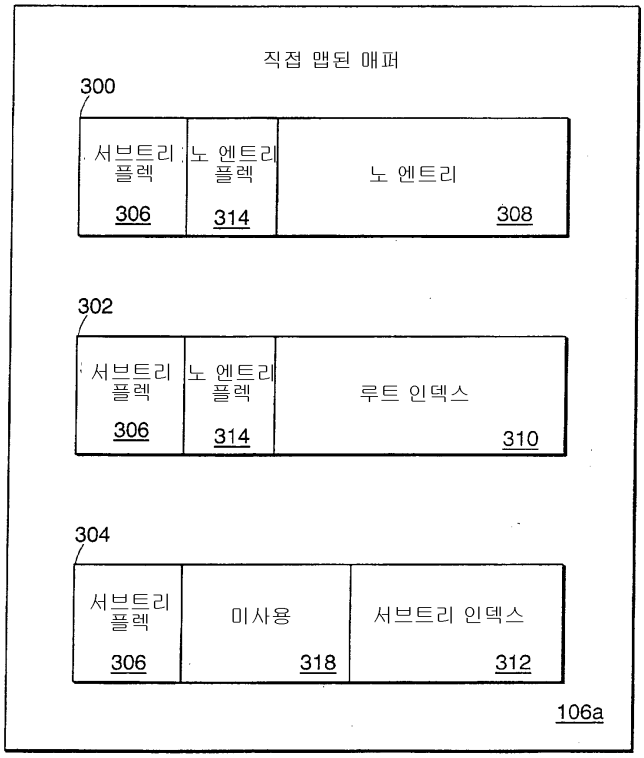
도면2b


버퍼 레벨 1
112a
버퍼 레벨 2
112b
버퍼 레벨 3
112c
r0
r1
r2
r3
r4
r5
r6
116
114
64k
s0
s1
s2
s3
s4
s5
256
256
A
B
A1
A2
B1
B2
h0
h1
138
140

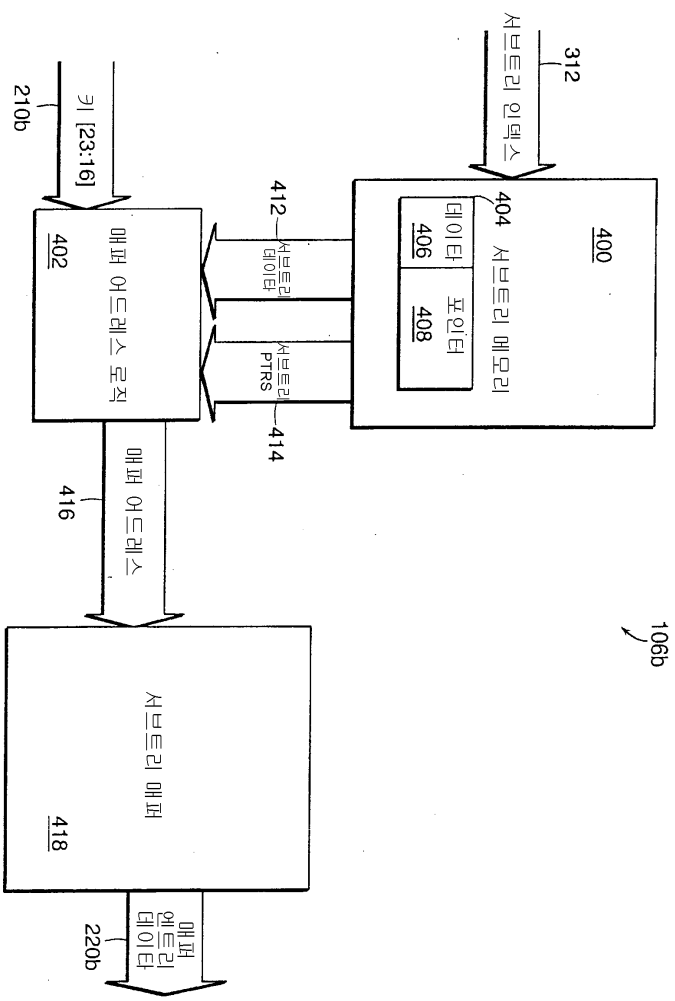
도면3


200
K39
K38
K37
K36
K35
K34
K33
K32
K31
K30
K29
K28
K27
K26
K25
K24
K23
K22
K21
K20
K19
K18
K17
K16
K15
K14
K13
K12
K11
K10
K9
K8
K7
K6
K5
K4
K3
K2
K1
K0
키
210
210a
키
[39:24]
L1
직접 맵된
매퍼
106a
220a
L1매퍼 엔트리 데이타
210b
키 [23:16]
L2
간접 매퍼
106b
220b
L2 매퍼 엔트리
데이타
210c
키 [15:8]
L3
간접 매퍼
106c
220c
L3
매퍼
엔트리
데이타
210d
키 [7:0]
L4
간접 매퍼
106d
220d
L4
매퍼
엔트리
데이타
파이프
라인
208
102
루트
인덱스

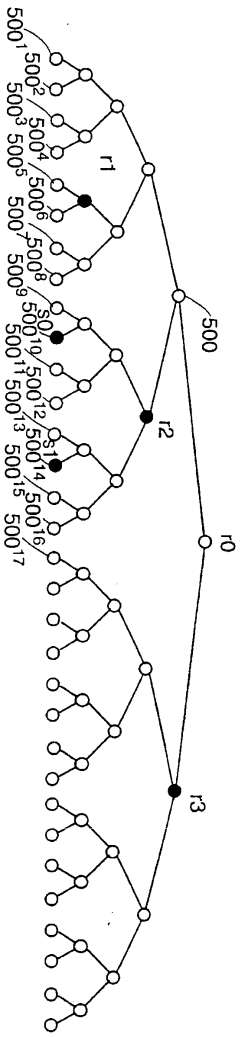
도면4


직접 맵된 매퍼
300
서브트리 플렉
306
노 엔트리 플렉
314
노 엔트리
308
302
서브트리 플렉
306
노 엔트리 플렉
314
루트 인덱스
310
304
서브트리 플렉
306
미사용
318
서브트리 인덱스
312
106a

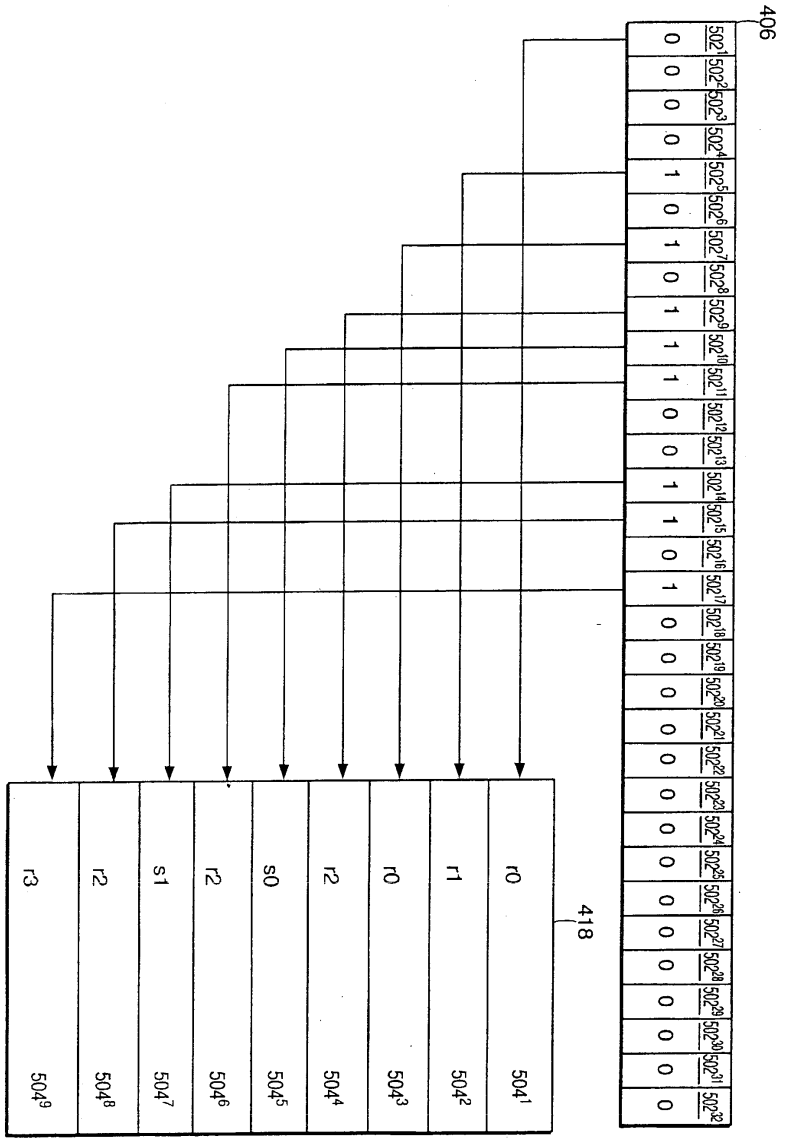
도면5


서브트리 인덱스
312
키 [23:16]
210b
400
404 서브트리 메모리
데이타
406
포인터
408
412
서브트리 데이타
서브트리 PTRS
414
매퍼 어드레스 로직
402
416
매퍼 어드레스
서브트리 매퍼
418
매퍼 엔트리 데이타
220b
106b

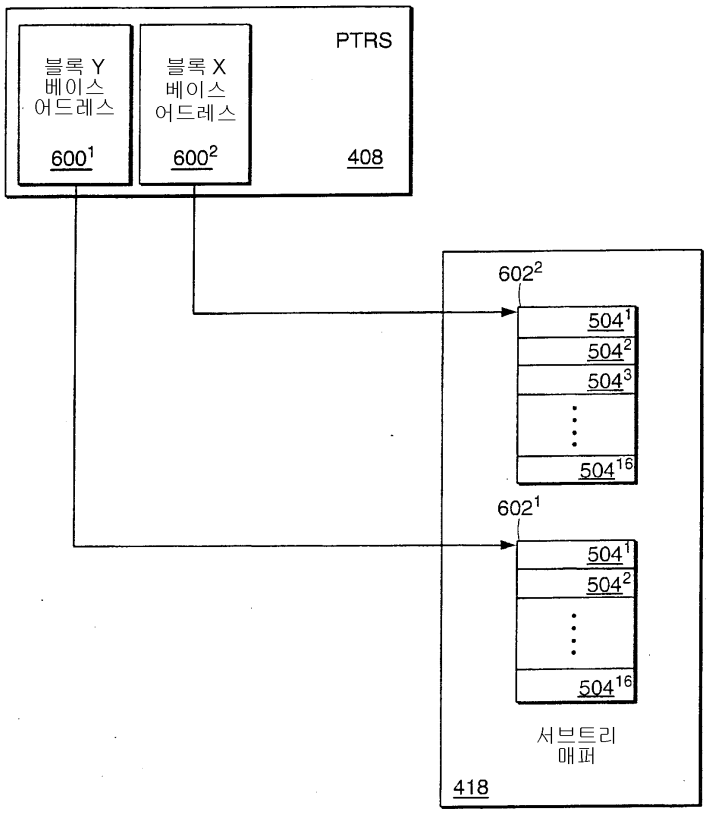
도면6a


500
r0
r1
r2
r3
500^{1}
500^{2}
500^{3}
500^{4}
500^{5}
500^{6}
500^{7}
500^{8}
500^{9}
500^{10}
500^{11}
500^{12}
500^{13}
500^{14}
500^{15}
500^{16}
500^{17}
s0
s1

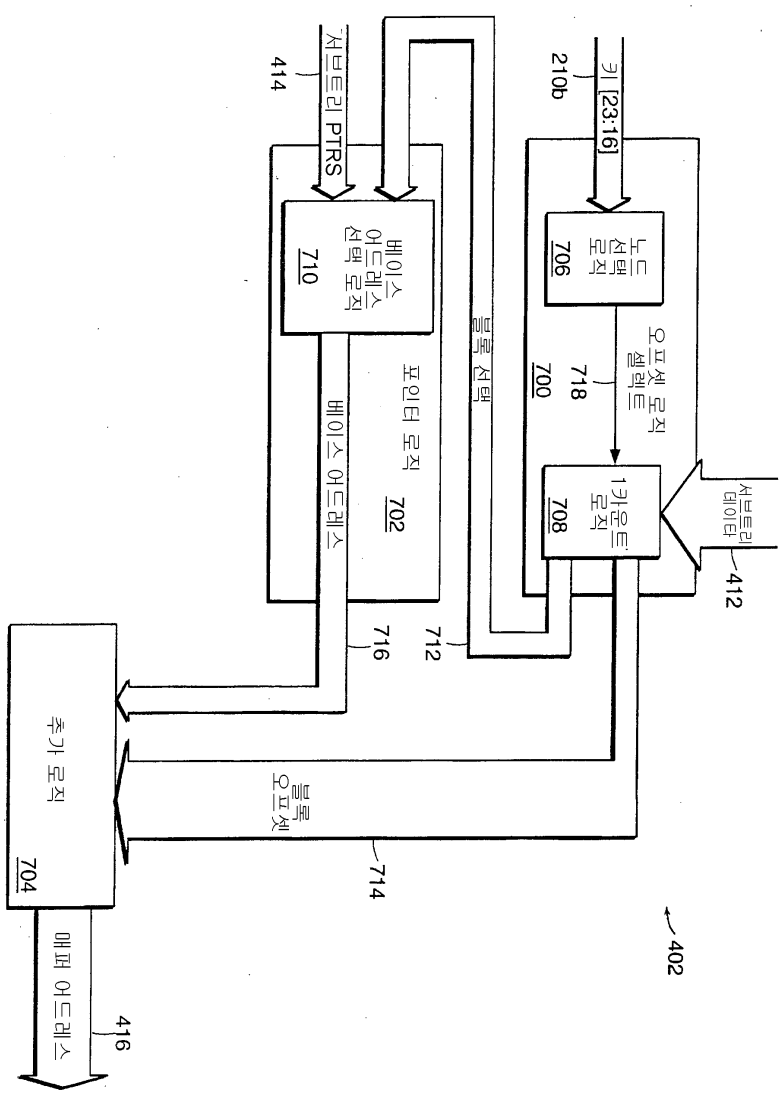
도면6b


406
0 0 0 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
418
r0
r1
r0
r2
s0
r2
s1
r2
r3

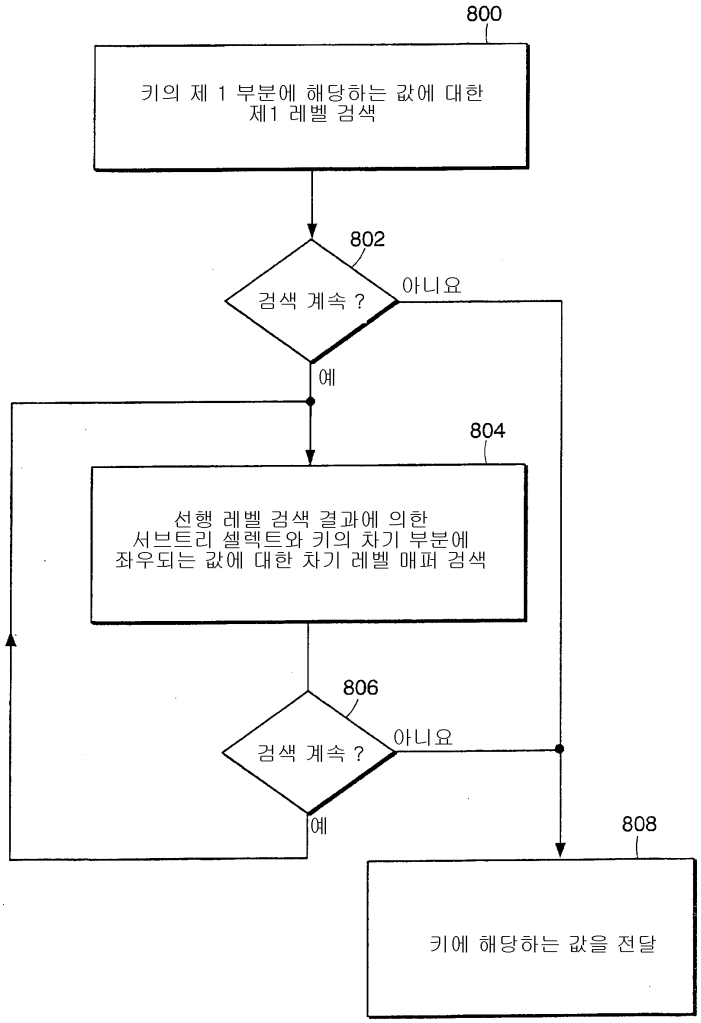
도면7


PTRS
블록 Y 베이스 어드레스
600¹
블록 X 베이스 어드레스
600²
408
602²
504¹
504²
504³
504¹⁶
602¹
504¹
504²
504¹⁶
서브트리 매퍼
418

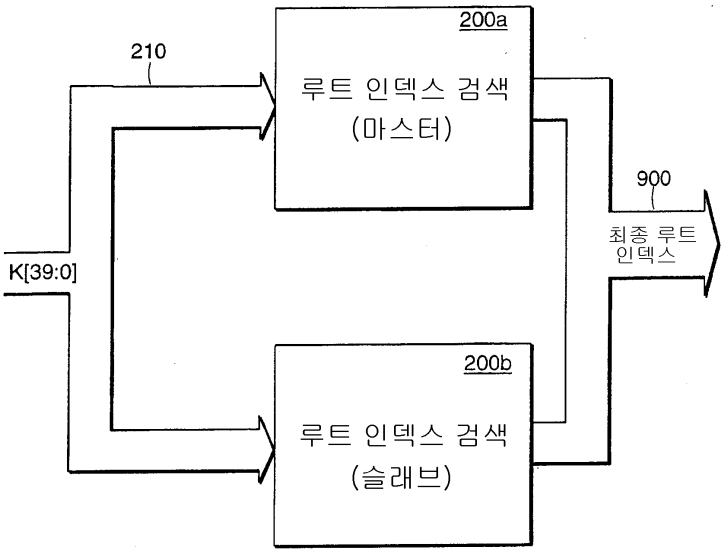
도면8


키 [23:16]
210b
서브트리 PTRS
414
노드 선택 로직
706
베이스 어드레스 선택 로직
710
오프셋 로직 셀렉트
718
700
블록 선택
포인터 로직
702
베이스 어드레스
716
712
1카운트 로직
708
서브트리 데이터
412
추가 로직
704
블록 오프셋
714
402
매퍼 어드레스
416

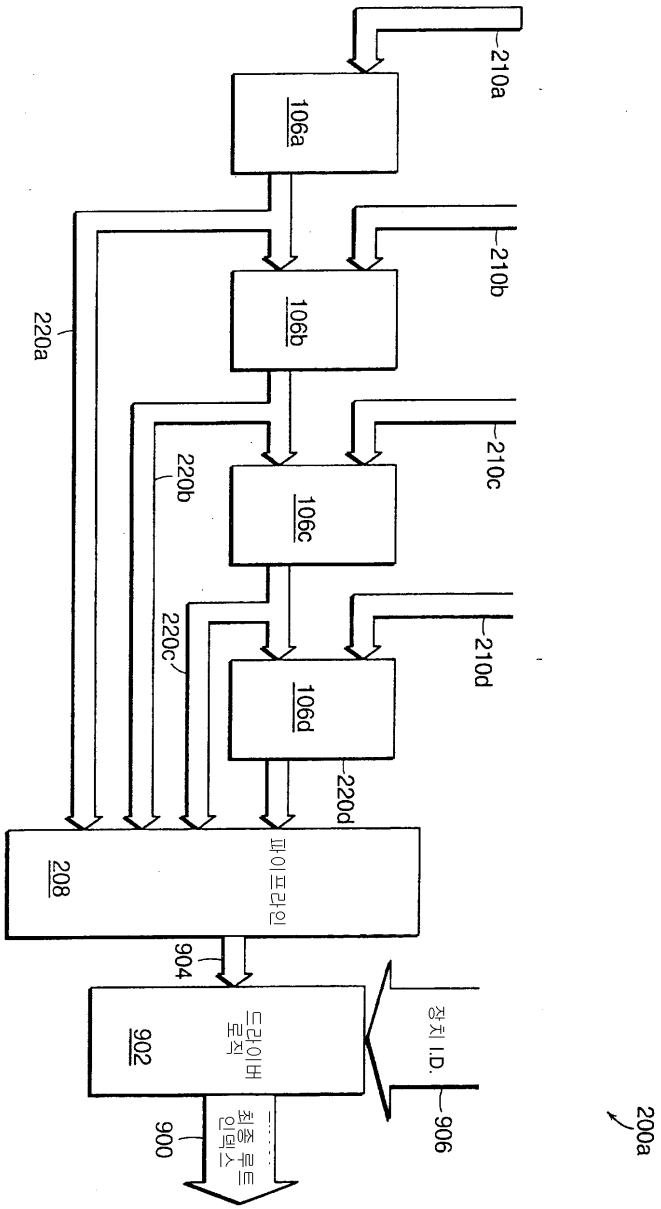
도면9


800
키의 제 1 부분에 해당하는 값에 대한
제1 레벨 검색
802
검색 계속 ?
아니요
예
804
선행 레벨 검색 결과에 의한
서브트리 셀렉트와 키의 차기 부분에
좌우되는 값에 대한 차기 레벨 매퍼 검색
806
검색 계속 ?
아니요
예
808
키에 해당하는 값을 전달

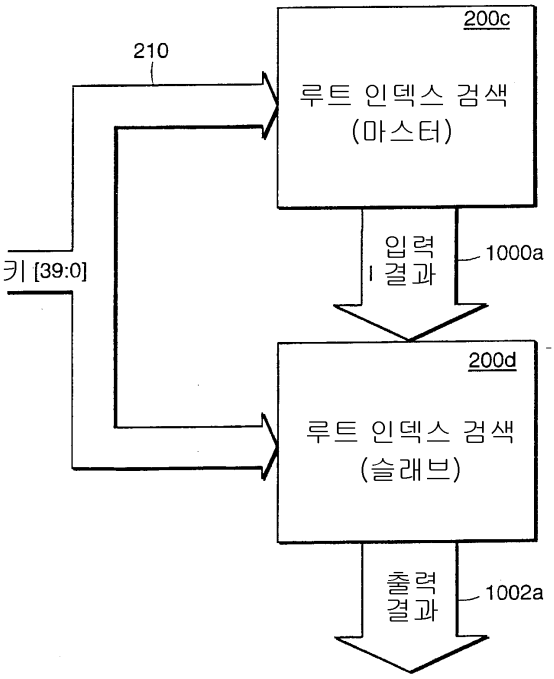
도면10a


200a
210
루트 인덱스 검색
(마스터)
900
최종 루트
인덱스
K[39:0]
200b
루트 인덱스 검색
(슬래브)

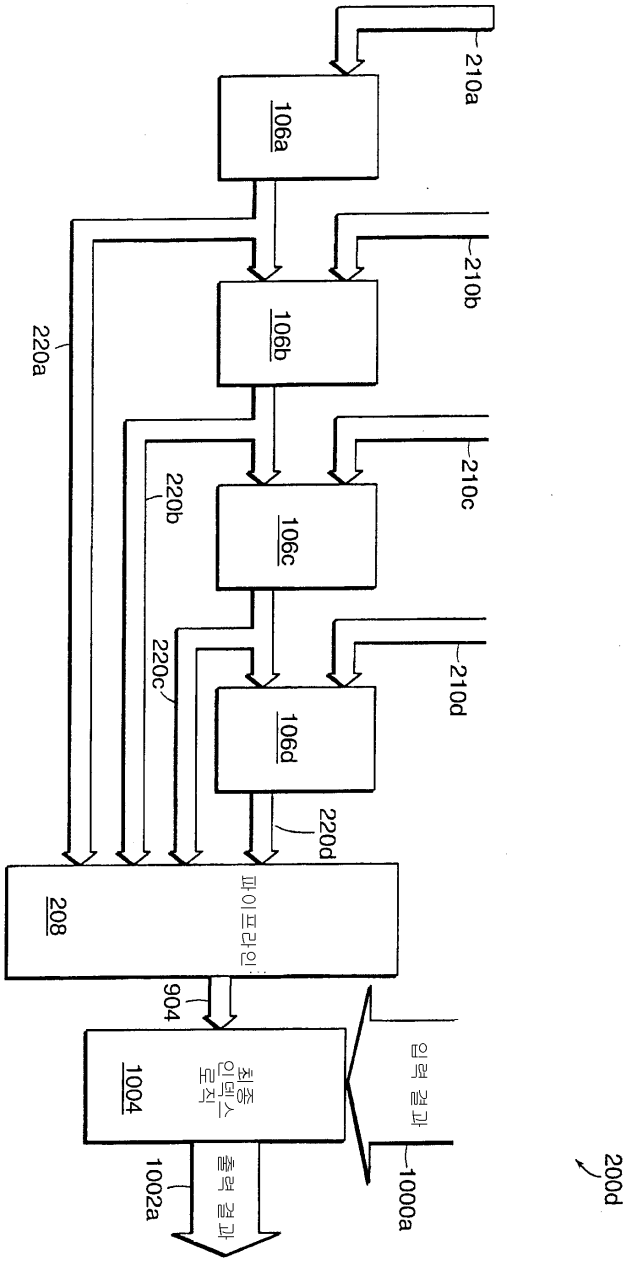
도면10b


210a
106a
210b
106b
220a
210c
220b
106c
210d
220c
106d
220d
208
파이프라인
904
902
드라이버
로직
장치 I.D.
906
900
최종 루트
인덱스
200a

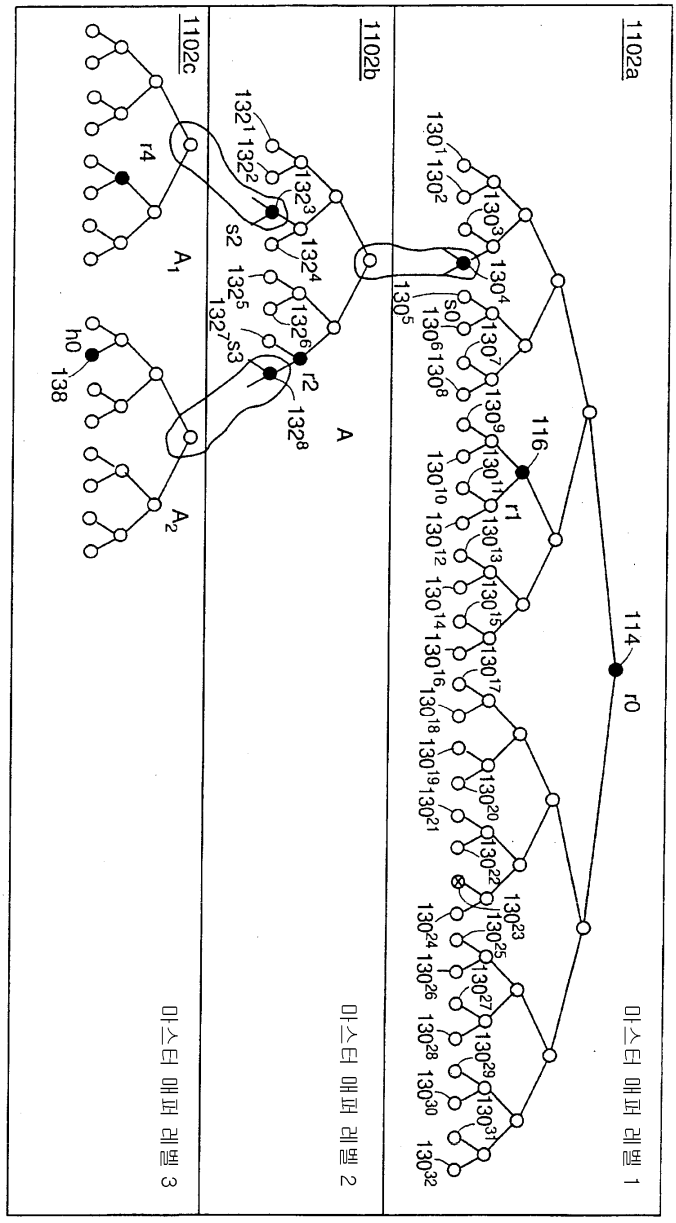
도면10c


200c
210
루트 인덱스 검색
(마스터)
입력
결과
1000a
키 [39:0]
200d
루트 인덱스 검색
(슬래브)
출력
결과
1002a

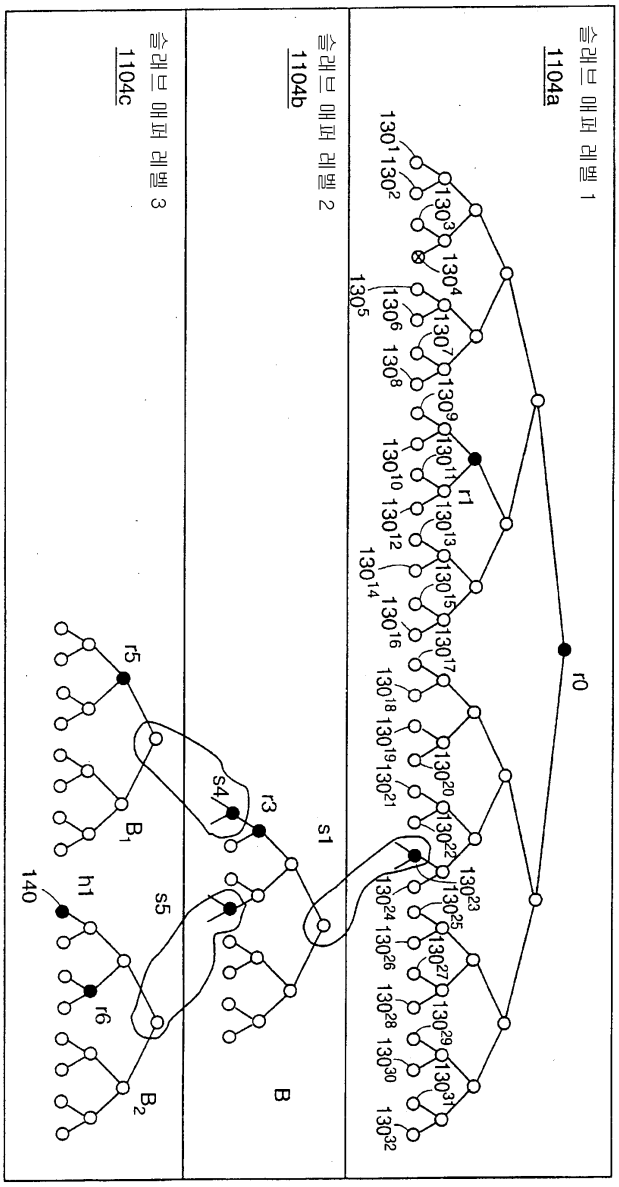
도면10d


210a
106a
210b
106b
220a
210c
220b
106c
220c
210d
106d
220d
208
파이프라인
904
최종 인덱스 로직
1004
입력 결과
1000a
출력 결과
1002a
200d

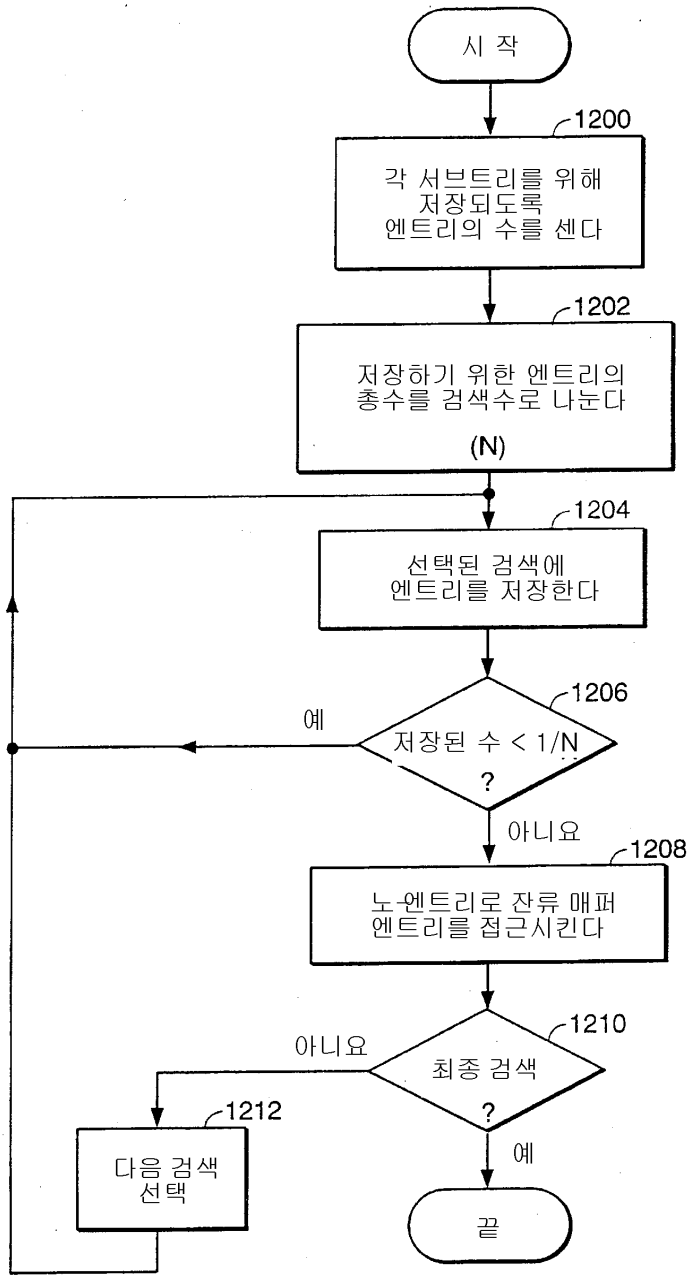
도면11a


1102a
1102b
1102c
마스터 매퍼 레벨 1
마스터 매퍼 레벨 2
마스터 매퍼 레벨 3
r0
r1
r2
r4
s0
s2
s3
h0
116
114
138
A
A1
A2

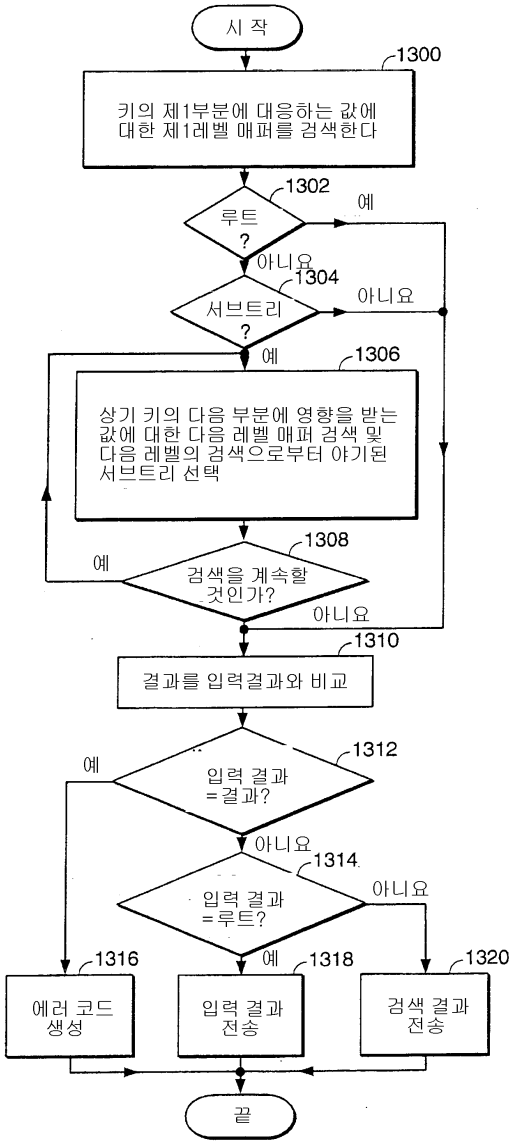
도면11b


슬래브 매퍼 레벨 1
1104a
슬래브 매퍼 레벨 2
1104b
슬래브 매퍼 레벨 3
1104c
r0
r1
r3
r5
r6
s1
s4
s5
h1
B
B1
B2
140

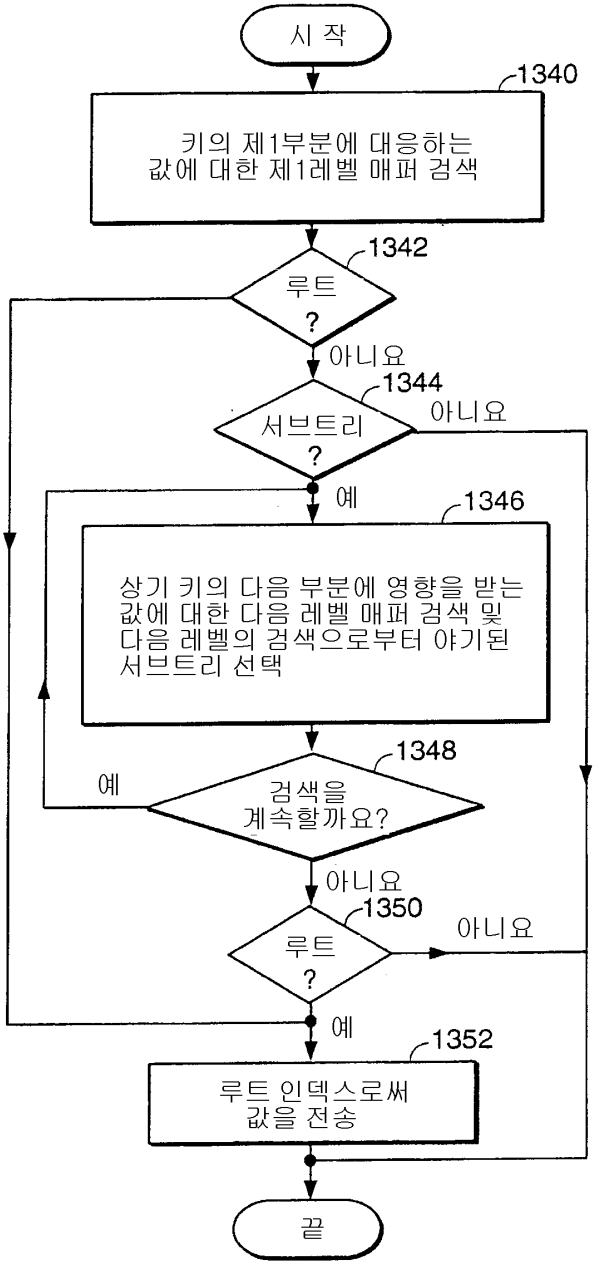
도면12


시 작
1200
각 서브트리를 위해
저장되도록
엔트리의 수를 센다
1202
저장하기 위한 엔트리의
총수를 검색수로 나눈다
(N)
1204
선택된 검색에
엔트리를 저장한다
1206
예
저장된 수 < 1/N
?
아니요
1208
노-엔트리로 잔류 매퍼
엔트리를 접근시킨다
1210
아니요
최종 검색
?
예
1212
다음 검색
선택
끝

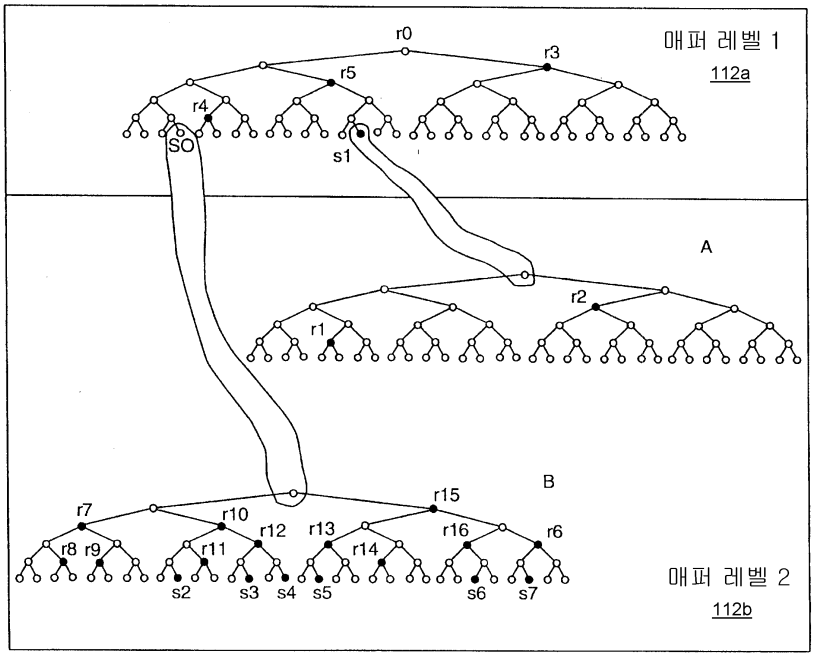
도면13


시 작
1300
키의 제1부분에 대응하는 값에 대한 제1레벨 매퍼를 검색한다
1302
루트 ?
예
아니요
1304
서브트리 ?
아니요
예
1306
상기 키의 다음 부분에 영향을 받는 값에 대한 다음 레벨 매퍼 검색 및 다음 레벨의 검색으로부터 야기된 서브트리 선택
1308
예
검색을 계속할 것인가?
아니요
1310
결과를 입력결과와 비교
1312
예
입력 결과 = 결과?
아니요
1314
입력 결과 = 루트?
아니요
예
1316
에러 코드 생성
1318
입력 결과 전송
1320
검색 결과 전송
끝

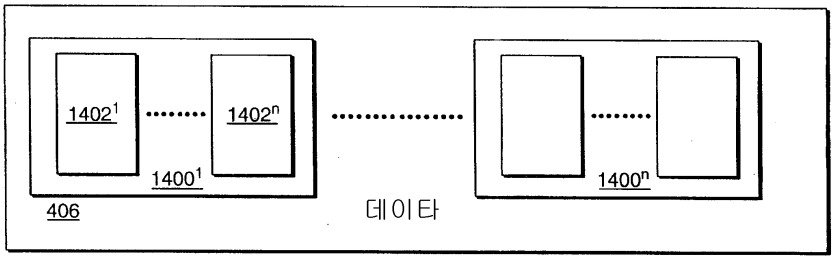
도면14


시 작
1340
키의 제1부분에 대응하는 값에 대한 제1레벨 매퍼 검색
1342
루트 ?
아니요
1344
서브트리 ?
아니요
예
1346
상기 키의 다음 부분에 영향을 받는 값에 대한 다음 레벨 매퍼 검색 및 다음 레벨의 검색으로부터 야기된 서브트리 선택
1348
예
검색을 계속할까요?
아니요
1350
루트 ?
아니요
예
1352
루트 인덱스로써 값을 전송
끝

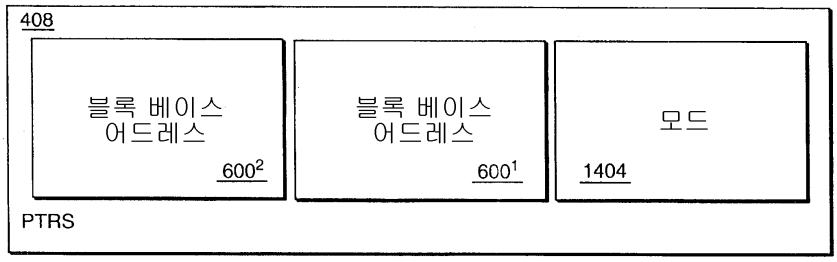
도면15



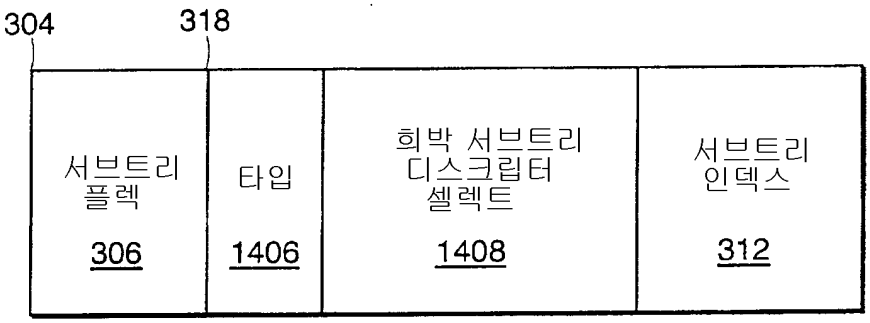
도면16a



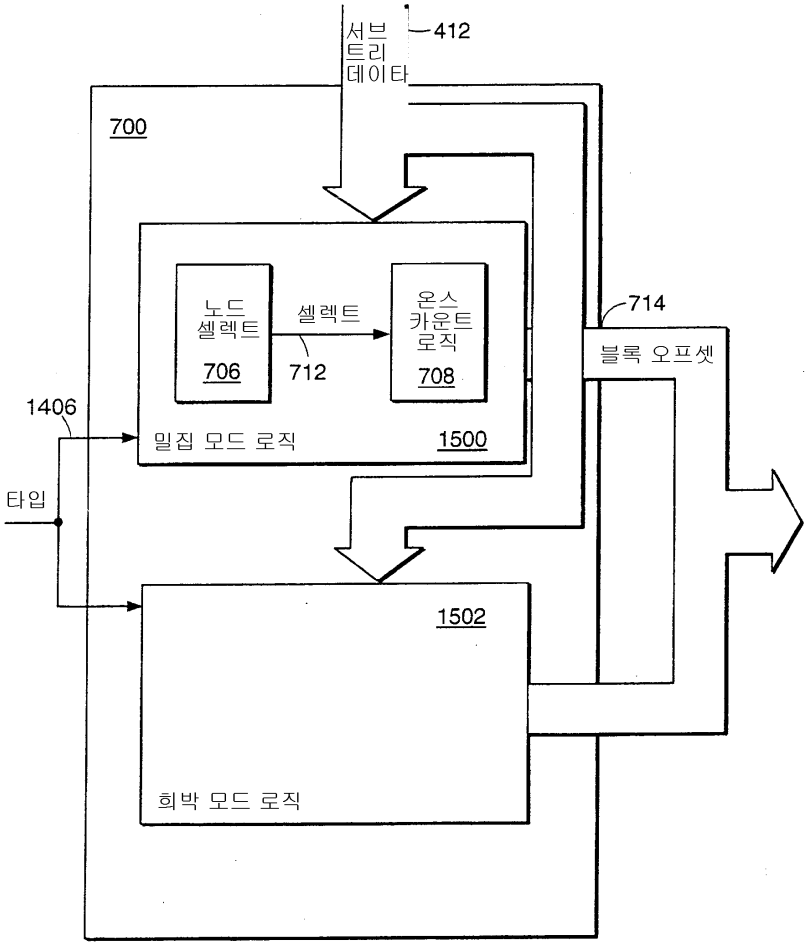
도면16b



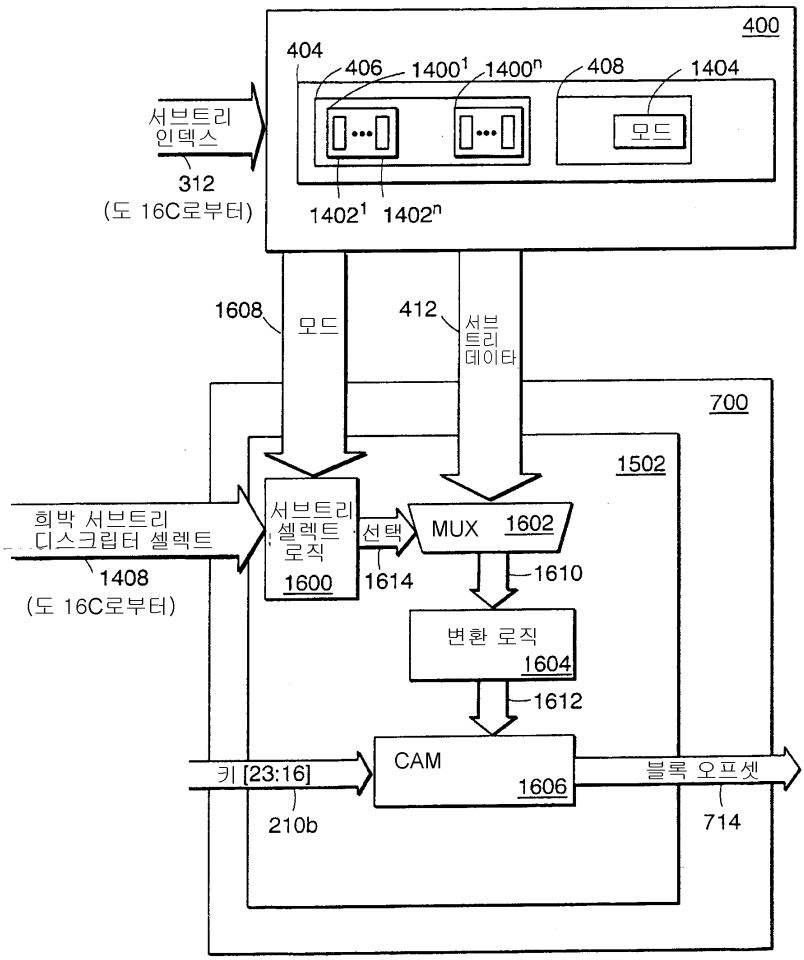
도면16c


304
318
서브트리 플렉
306
타입
1406
희박 서브트리 디스크립터 셀렉트
1408
서브트리 인덱스
312

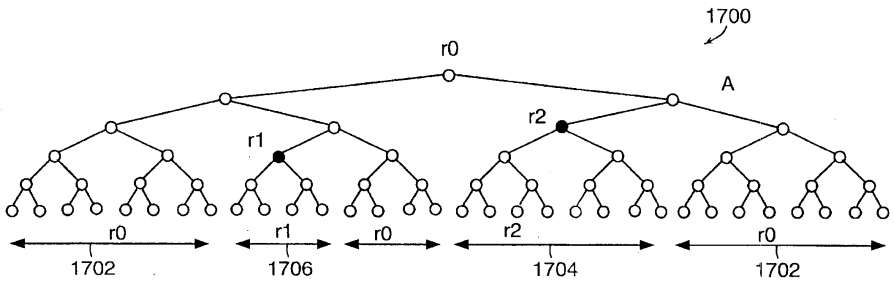
도면17


서브 트리 데이타
412
700
노드 셀렉트
706
셀렉트
712
온스 카운트 로직
708
714
블록 오프셋
1406
밀집 모드 로직
1500
타입
1502
희박 모드 로직

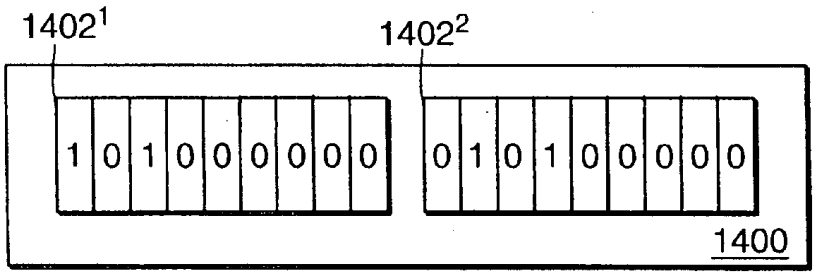
도면18



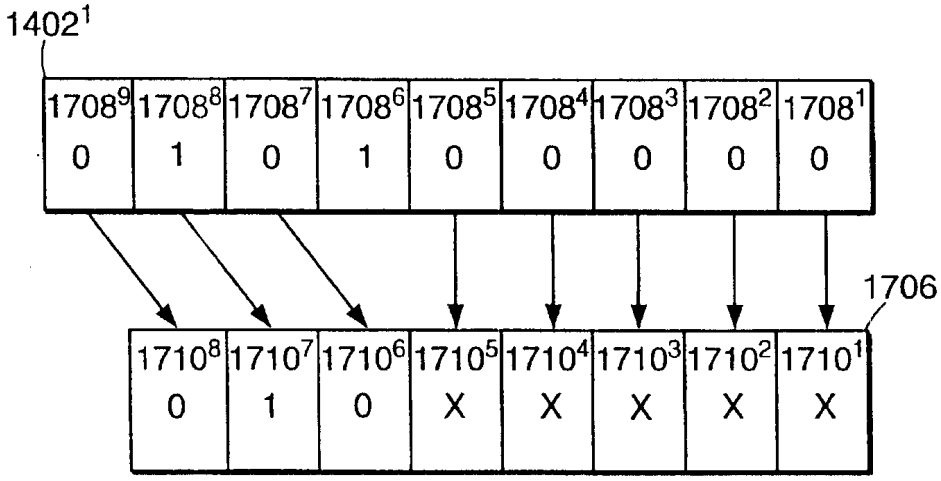
도면19a



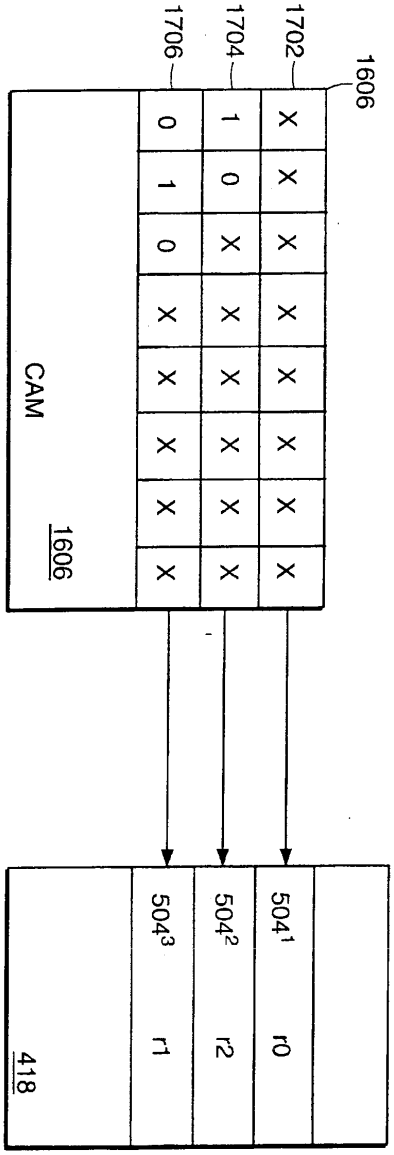
도면19b



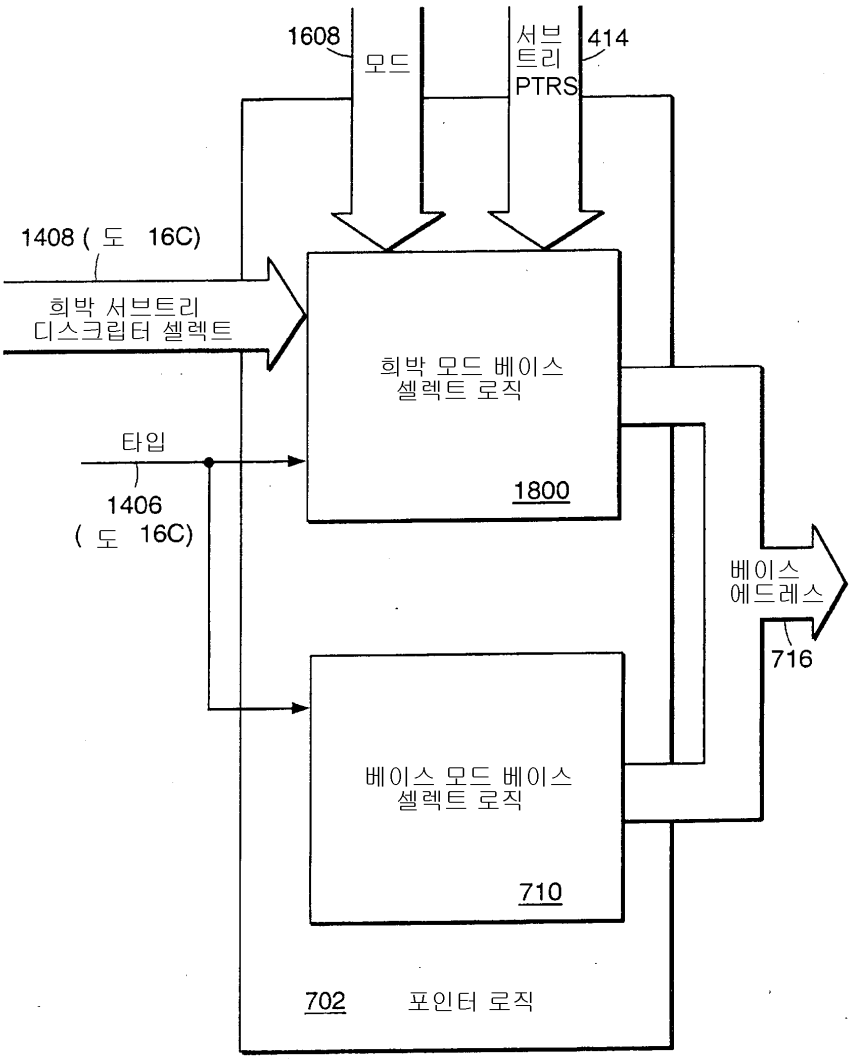
도면19c


1402[1]
1708[9] 1708[8] 1708[7] 1708[6] 1708[5] 1708[4] 1708[3] 1708[2] 1708[1]
0 1 0 1 0 0 0 0 0
1706
1710[8] 1710[7] 1710[6] 1710[5] 1710[4] 1710[3] 1710[2] 1710[1]
0 1 0 X X X X X

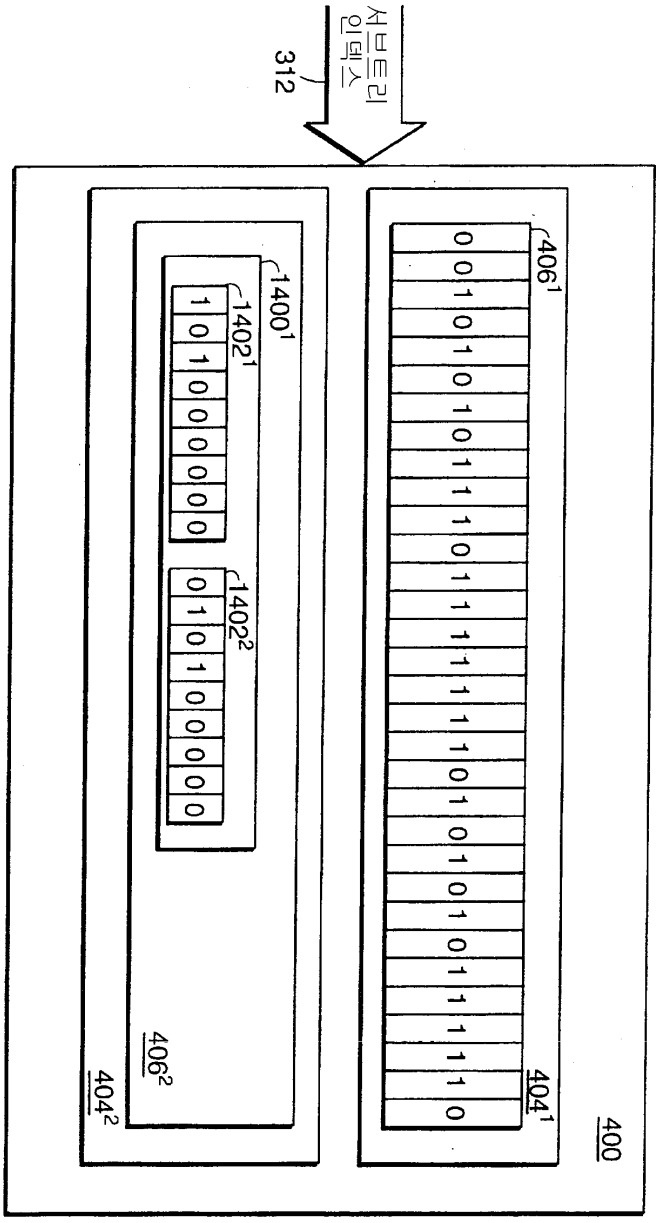
도면19d


1706
1704
1702
1606
CAM
1606
418
504[3] r1
504[2] r2
504[1] r0

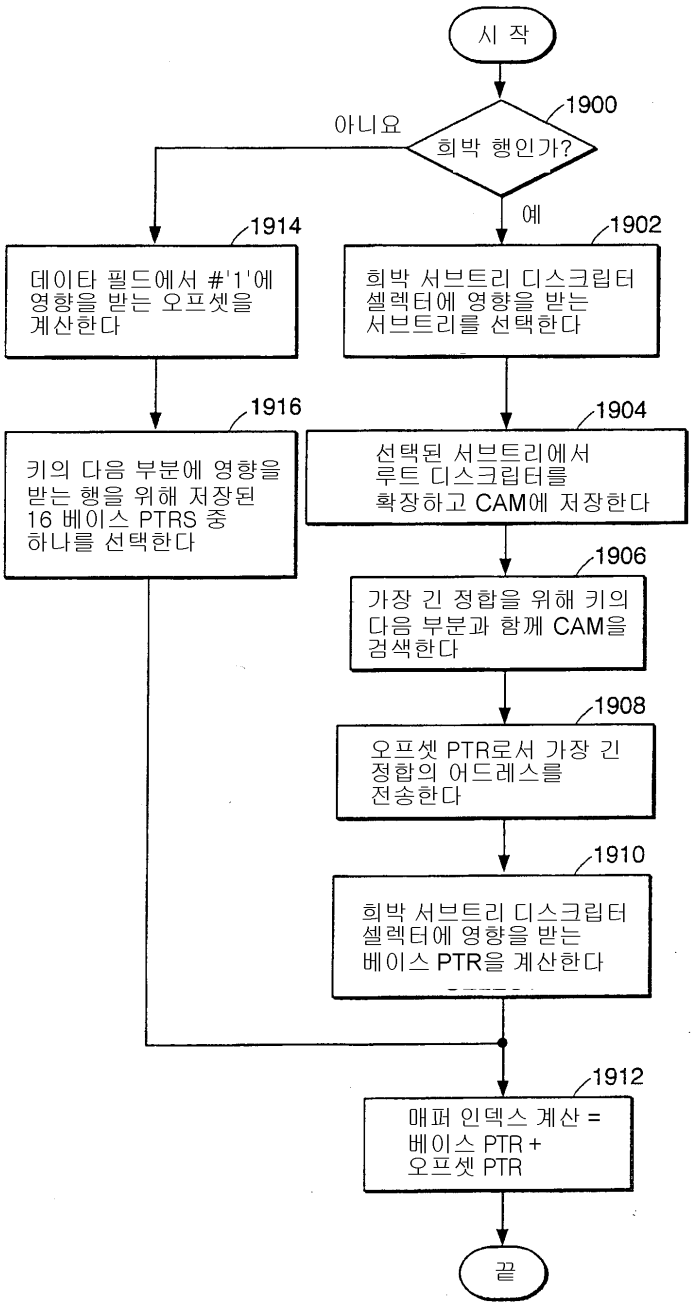
도면20


1608
모드
서브
트리
PTRS
414
1408 (도 16C)
희박 서브트리
디스크립터 셀렉트
희박 모드 베이스
셀렉트 로직
1800
타입
1406
(도 16C)
베이스
에드레스
716
베이스 모드 베이스
셀렉트 로직
710
702
포인터 로직

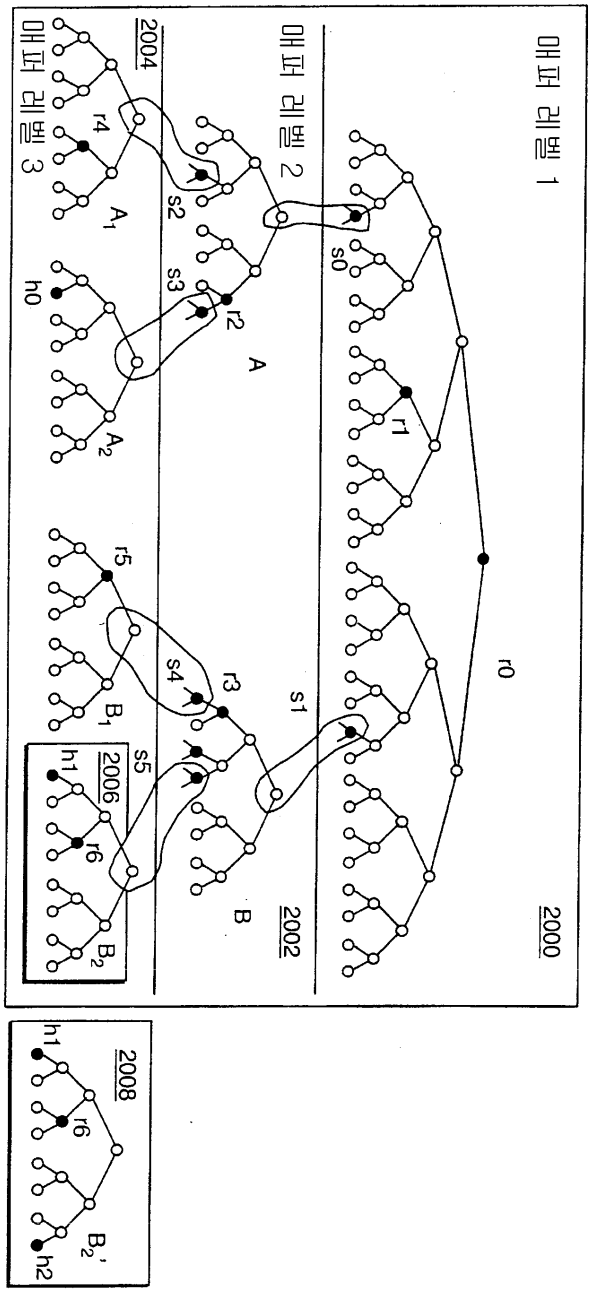
도면21



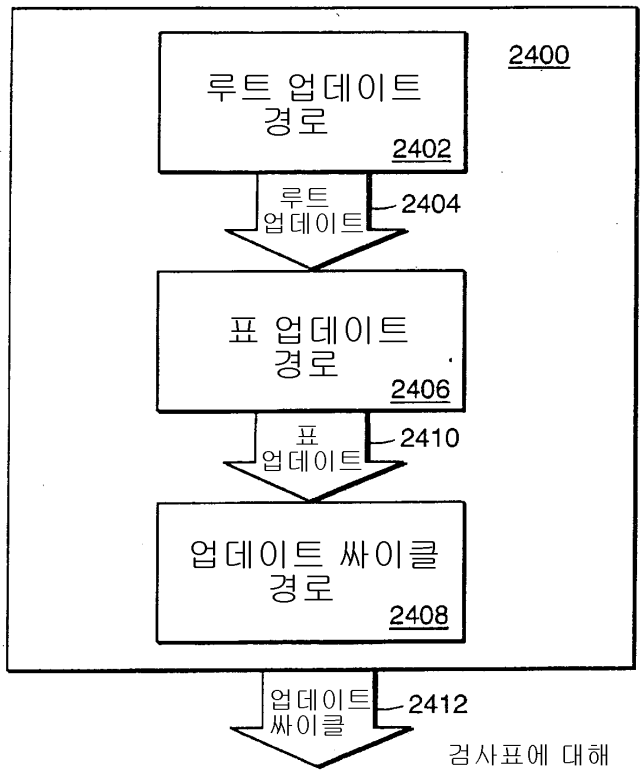
도면22


시 작
1900
희박 행인가?
아니요
예
1914
데이타 필드에서 #'1'에 영향을 받는 오프셋을 계산한다
1902
희박 서브트리 디스크립터 셀렉터에 영향을 받는 서브트리를 선택한다
1916
키의 다음 부분에 영향을 받는 행을 위해 저장된 16 베이스 PTRS 중 하나를 선택한다
1904
선택된 서브트리에서 루트 디스크립터를 확장하고 CAM에 저장한다
1906
가장 긴 정합을 위해 키의 다음 부분과 함께 CAM을 검색한다
1908
오프셋 PTR로서 가장 긴 정합의 어드레스를 전송한다
1910
희박 서브트리 디스크립터 셀렉터에 영향을 받는 베이스 PTR을 계산한다
1912
매퍼 인덱스 계산 = 베이스 PTR + 오프셋 PTR
끝

도면23

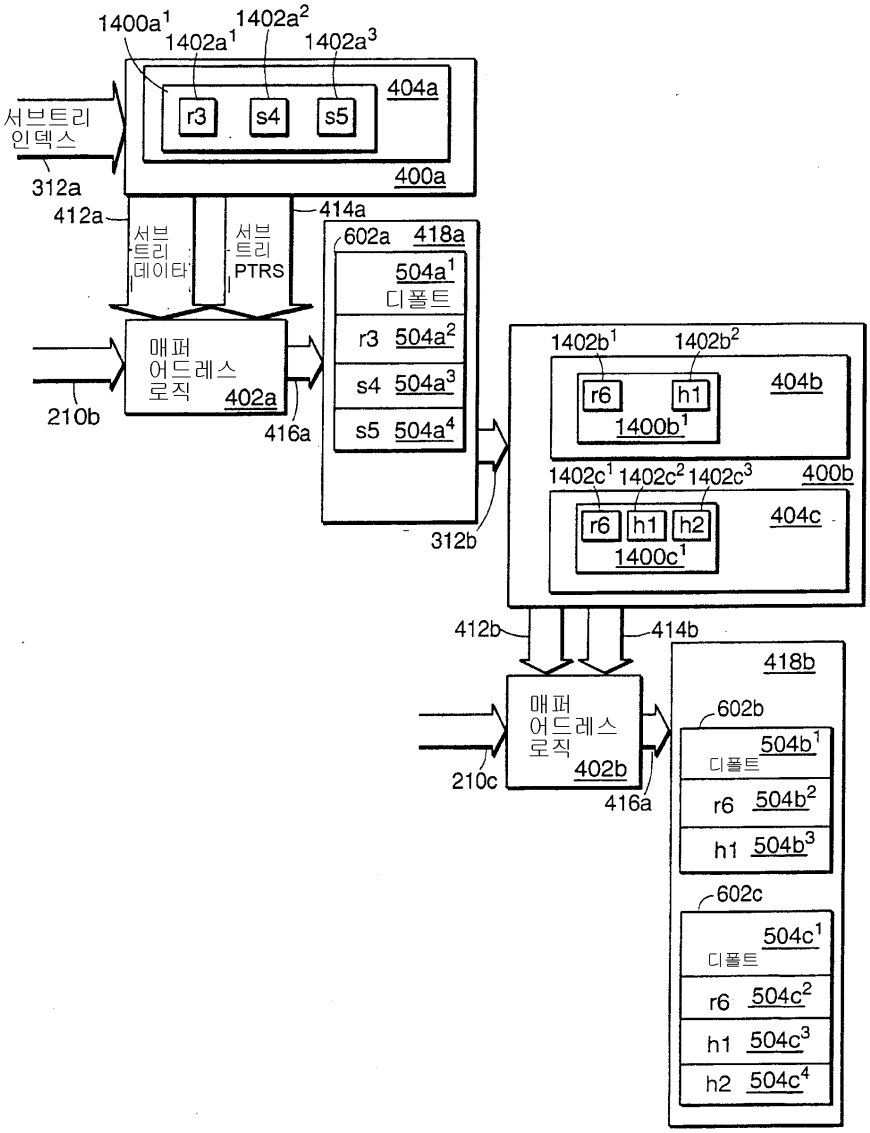

매퍼 레벨 1
매퍼 레벨 2
매퍼 레벨 3
2000
2002
2004
2006
2008
r0
r1
r2
r3
r4
r5
r6
s0
s1
s2
s3
s4
s5
h0
h1
h2
A
B
A1
A2
B1
B2
B2'

도면24

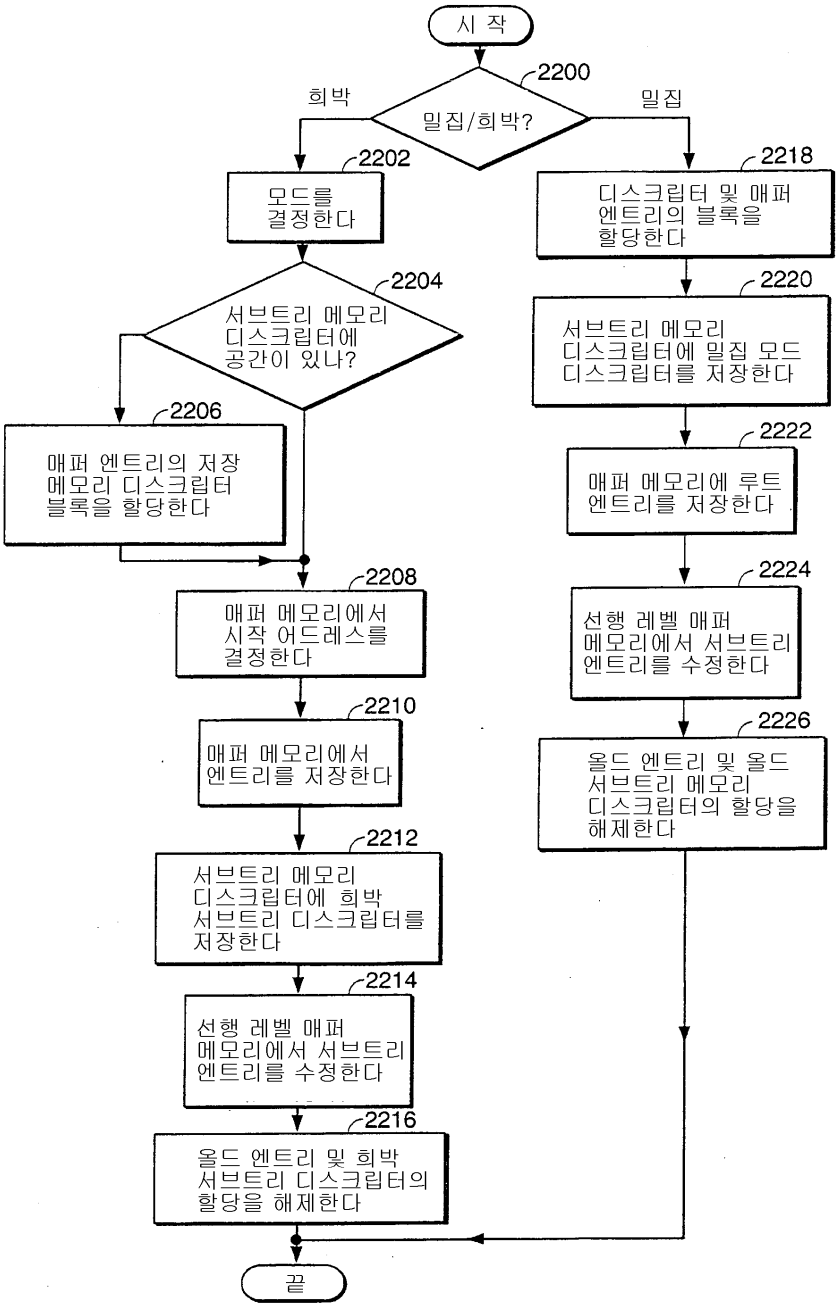


검사표에 대해
200 (도3)

도면25


1400a1
1402a2
1402a1
1402a3
r3
s4
s5
404a
400a
서브트리 인덱스
312a
412a
414a
서브 트리 데이타
서브 트리 PTRS
602a
418a
504a1 디폴트
r3 504a2
s4 504a3
s5 504a4
매퍼 어드레스 로직
402a
210b
416a
312b
1402b1
1402b2
r6
h1
1400b1
404b
1402c1
1402c2
1402c3
400b
r6
h1
h2
1400c1
404c
412b
414b
418b
매퍼 어드레스 로직
402b
210c
416a
602b
504b1 디폴트
r6 504b2
h1 504b3
602c
504c1 디폴트
r6 504c2
h1 504c3
h2 504c4

도면26


시 작
2200
밀집/희박?
희박
밀집
2202
모드를 결정한다
2204
서브트리 메모리 디스크립터에 공간이 있나?
2206
매퍼 엔트리의 저장 메모리 디스크립터 블록을 할당한다
2208
매퍼 메모리에서 시작 어드레스를 결정한다
2210
매퍼 메모리에서 엔트리를 저장한다
2212
서브트리 메모리 디스크립터에 희박 서브트리 디스크립터를 저장한다
2214
선행 레벨 매퍼 메모리에서 서브트리 엔트리를 수정한다
2216
올드 엔트리 및 희박 서브트리 디스크립터의 할당을 해제한다
2218
디스크립터 및 매퍼 엔트리의 블록을 할당한다
2220
서브트리 메모리 디스크립터에 밀집 모드 디스크립터를 저장한다
2222
매퍼 메모리에 루트 엔트리를 저장한다
2224
선행 레벨 매퍼 메모리에서 서브트리 엔트리를 수정한다
2226
올드 엔트리 및 올드 서브트리 메모리 디스크립터의 할당을 해제한다
끝