



- (51) 국제특허분류:  
H04L 29/08 (2006.01) H04B 1/00 (2006.01)  
H04W 4/00 (2009.01) G06F 9/455 (2006.01)
- (21) 국제출원번호: PCT/KR2017/001504
- (22) 국제출원일: 2017년 2월 10일 (10.02.2017)
- (25) 출원언어: 한국어
- (26) 공개언어: 한국어
- (30) 우선권정보:  
10-2016-0016342 2016년 2월 12일 (12.02.2016) KR  
10-2016-0087107 2016년 7월 8일 (08.07.2016) KR  
10-2016-0106773 2016년 8월 23일 (23.08.2016) KR  
10-2016-0111321 2016년 8월 31일 (31.08.2016) KR
- (71) 출원인: 한양대학교 산학협력단 (INDUSTRY-UNIVERSITY COOPERATION FOUNDATION HANYANG UNIVERSITY) [KR/KR]; 04763 서울시 성동구 왕십리로 222, Seoul (KR).
- (72) 발명자: 최승원 (CHOI, Seung Won); 04753 서울시 성동구 마장로 35길 76, 111동 501호, Seoul (KR). 김용 (JIN, Yong); 04763 서울시 성동구 왕십리로 222, 102호, Seoul (KR). 금동현 (KUEM, Dong Hyun); 15002 경

기도 시흥시 진말로 36, 113동 1202호, Gyeonggi-do (KR). 김경훈 (KIM, Kyung Hoon); 05526 서울시 송파구 풍성로 6가길 7, 502호, Seoul (KR). 안홍섭 (AHN, Heung Seop); 08202 서울시 구로구 신도림로 87, 101동 804호, Seoul (KR).

(74) 대리인: 특허법인 이상 (E-SANG PATENT & TRADE-MARK LAW FIRM); 06747 서울시 서초구 바우피로 188, 3층, Seoul (KR).

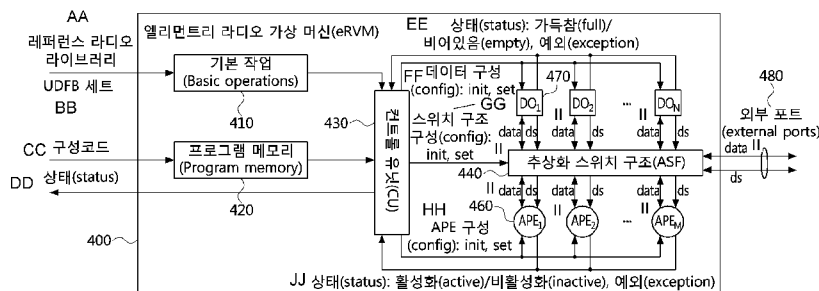
(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG,

[다음 쪽 계속]

(54) Title: METHOD AND TERMINAL DEVICE FOR EXECUTING RADIO APPLICATION

(54) 발명의 명칭: 라디오 어플리케이션을 실행하는 방법 및 단말 장치



- 400 ... Elementary radio virtual machine (eRVM)  
410 ... Basic operations  
420 ... Program memory  
430 ... Control unit (CU)  
440 ... Abstract switch framework (ASF)  
480 ... External ports  
AA ... Reference radio library  
BB ... UDFB set  
CC ... Configuration code  
DD ... Status  
EE ... Status: full/empty, exception  
FF ... Data configuration: init, set  
GG ... Switch structure configuration: init, set  
HH ... APE configuration: init, set  
II ... Data  
JJ ... Status: active/inactive, exception

(57) Abstract: A method and a terminal device for executing a radio application are disclosed. The method for executing a radio application comprises the steps of: allowing a radio virtual machine (RVM) of a reconfigurable mobile device to reference a radio programming interface (RPI); and replacing at least one abstract processing element (APE) of the RVM with at least one elementary RVM according to an RVM hierarchical structure referenced by the RPI.

(57) 요약서: 라디오 어플리케이션을 실행하는 방법 및 단말 장치가 개시된다. 라디오 어플리케이션 실행 방법은, 재구성 가능 모바일 장치의 라디오 가상 머신(RVM, Radio Virtual Machine)에서 라디오 프로그래밍 인터페이스(RPI, Radio Programming Interface)를 참조하는 단계, 및 상기 라디오 프로그래밍 인터페이스에서 참조한 라디오 가상 머신의 계층 구조에 따라 라디오 가상 머신의 적어도 하나의 추상화 처리 엘리먼트(APE, Abstract Processing Element)를 적어도 하나의 엘리먼트 라디오 가상 머신(elementary RVM)으로 대체하는 단계를 포함한다.



ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, RU, TJ, **공개:**

TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

— 국제조사보고서와 함께 (조약 제 21 조(3))

## 명세서

### 발명의 명칭: 라디오 어플리케이션을 실행하는 방법 및 단말 장치 기술분야

- [1] 본 발명은 소프트웨어 정의 라디오(SDR, Software Defined Radio), 디지털 무선 통신(Digital Wireless Communications), 라디오 프로세서(RP, Radio Processor), 어플리케이션 프로세서(AP, Application Processor), 멀티 라디오 어플리케이션(Multi-Radio Application), 재구성 가능 라디오 시스템(RRS, Reconfigurable Radio Systems), 또는 모바일 장치(MD, Mobile Device)의 라디오 재구성 관련 요구조건과 관련된 기술 분야에서 라디오 앱 스토어(Radio App Store)나 재구성 가능 모바일 장치(Reconfigurable MD)에 라디오 어플리케이션 패키지를 공급하는 라디오 어플리케이션 제공자(Radio Application Provider)의 라디오 프로그래밍 인터페이스(RPI, Radio Programming Interface)와 이를 이용하는 재구성 가능 모바일 장치에 관한 것이다.
- [2] 특히, 본 발명은 재구성 가능 모바일 장치의 라디오 가상 머신(RVM, Radio Virtual Machine)에 관한 것으로, 더욱 상세하게는 라디오 어플리케이션 제공자나 라디오 앱 스토어로부터의 라디오 어플리케이션 구성코드(Radio Apps. configcode)에 포함된 사용자 정의 기능 블록(UDFB, User Defined Functional Block)을 실행할 수 있는 단말 장치의 구조 및 이 구조에 적용할 수 있는 라디오 프로그래밍 인터페이스(RPI)를 이용한 라디오 어플리케이션 실행 방법에 관한 것이다.

### 배경기술

- [3] 통신기술이 발달함에 따라 사용자의 기호나 목적에 따라 많은 새로운 종류의 라디오 통신 기술들이 사용되고 있다. 롱텀에볼루션(LTE), 롱텀에볼루션 어드벤스드(LTE-advanced), 광대역 코드분할다중접속(WCDMA), 모바일 와이맥스(Mobile WiMAX), GSM(Global System for Mobile Communications) 등과 같은 무선 통신 기술들 대부분은 모뎀과 기지국이 상호 작용하면서 단말기 상에서 실행된다.
- [4] 단말기 내부의 모뎀은 제조사별로 고유의 명령어들을 가지고 이를 통해서 각각의 라디오 통신 기술들을 구현한다. 라디오 어플리케이션들이 모뎀을 제어하기 위해서는 제조사 또는 모델에 따라 모뎀이 갖는 고유의 명령어를 이해하고 이에 맞춘 모듈을 개발하여 적용해야 한다. 이는 결과적으로 어떤 라디오 어플리케이션은 오직 특정 제조사의 단말기나 특정 모뎀에서만 실행되는 결과를 초래한다. 이를 해결하기 위해 여러 종류의 모뎀에 맞춘 각기 다른 제어 명령을 모든 라디오 어플리케이션에 포함시키거나 각 모뎀별로 서로 다른 실행 파일을 제작하여 배포하여야 한다.
- [5] 그러나, 이러한 방법은 현재 시장에 출시되고 있는 다양한 모뎀의 하드웨어에

맞추어 일일이 최적화 작업을 별도로 수행해야 하므로 모든 단말기에서 작동 가능한 라디오 어플리케이션을 제작하는 것은 사실상 불가능하다. 그 뿐만 아니라 기존 기술에서는 하나의 멀티 라디오 어플리케이션을 제작하기 위하여 막대한 인력과 비용을 투입해야 하는 문제가 있다.

- [6] 이와 같은 문제를 해결하기 위해 하드웨어 독립적인 멀티 라디오 어플리케이션을 제작하고자 하는 시도가 있으나 이를 제어하려면 각각 제조사별 고유의 명령어가 아닌 통일된 명령어를 사용해야 한다.
- [7] 즉, 무선 기지국과 단말기에서 하드웨어로 고주파(HF, High Frequency)를 지원하던 방식을 소프트웨어 형태로 바꿔주는 기술이 필요하다. 소프트웨어 정의 라디오(SDR, Software Defined Radio)는 다중 모드(Multimode), 다중 대역(Multi band), 다중 환경의 무선 통신 환경에서 하나의 단말기로 장소, 시간에 관계없이 서비스를 경제적으로 제공하기 위한 무선 통신 기술로, 소프트웨어의 조작으로 무선 장치 및 서비스를 제공하기 위해 제안된 기술이다.
- [8] 이동 전화, 개인 휴대 정보 단말기(PDA, Personal Digital Assistants), 노트북 등 휴대 단말기에 소프트웨어 정의 라디오 모듈을 탑재하면 하나의 단말기에 서로 다른 주파수 대역과 2개 이상의 시스템을 동시에 지원하는 것이 가능해진다. 이러한 SDR 기술은 All-IP 기반 무선 멀티미디어 통신을 추구하는 4세대 통신에서 다양한 무선 네트워크, 다양한 무선 통신 방식, 국가마다 다른 주파수 밴드, 고속 데이터 통신을 위한 새로운 통신 방식을 제공할 수 있다.
- [9] 소프트웨어 정의 라디오 기술과 관련하여 SCA(Software Communication Architecture)라는 사실상의 표준 기술이 존재한다. 이는 소프트웨어 정의 라디오를 위하여 필요한 프레임워크(Framework), 미들웨어(Middleware), 실시간 운영 시스템(Real Time Operating System)과 관련된 규약을 모아놓은 것으로 소프트웨어 정의 라디오 시스템 간의 인터페이스 호환(Interface Compatible)을 보장한다. SCA의 핵심은 프레임워크 규약인 코어 프레임워크(Core Framework)로 이를 통해 라디오 어플리케이션을 구성하는 각 부분을 컴포넌트화할 수 있도록 하고 이 컴포넌트들을 재사용하고 조합하여 새로운 라디오 어플리케이션을 만들 수 있도록 이루어진다.
- [10] SCA의 경우 단말기에 미리 설치된 블록들에 한하여 재조합이 가능하지만 특정 라디오 어플리케이션에서의 사용을 위한 사용자 정의(User Defined) 형태의 블록을 하드웨어 구성이 서로 다른 SCA 호환 단말기에서는 설치할 수 없다. 따라서 단일의 실행 파일로 모든 SCA 호환 단말기에서 사용하는 것이 불가능하다.
- [11] 이는 모든 SCA 호환 단말기에 탑재된 하드웨어의 사양에 따라 최적화된 실행 파일을 개별적으로 만들어 배포해야 함을 의미한다. 이는 매우 많은 시간과 비용을 필요로 하여 라디오 어플리케이션의 상업적 이용을 매우 어렵게 한다. 또한, SCA 호환 단말기는 라디오 어플리케이션의 구현을 위한 베이스밴드 API(Application Programming Interface)를 제공하지 않아 선택적 하드웨어 가속

기능의 활용이 어렵다.

## 발명의 상세한 설명

### 기술적 과제

- [12] 상기 종래 기술의 문제점을 해결하기 위한 본 발명의 목적은 라디오 프로그래밍 인터페이스(RPI, Radio Programming Interface)를 참조하여 라디오 어플리케이션 구성코드에 포함된 사용자 정의 기능 블록(UDFB, User Defined Functional Block)을 실행할 수 있는, 라디오 가상 머신을 이용한 하드웨어 독립적인 라디오 어플리케이션 실행 방법을 제공하는데 있다.

- [13] 또한, 본 발명의 다른 목적은 라디오 프로그래밍 인터페이스(RPI, Radio Programming Interface)를 참조하여 라디오 어플리케이션 구성코드에 포함된 사용자 정의 기능 블록(UDFB, User Defined Functional Block)을 하드웨어 독립적으로 실행할 수 있는, 라디오 가상 머신의 구조를 구비한 단말 장치를 제공하는 데 있다.

### 과제 해결 수단

- [14] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따르면, 라디오 어플리케이션 실행 방법은, 재구성 가능 모바일 장치의 라디오 가상 머신(RVM, Radio Virtual Machine)에서 라디오 프로그래밍 인터페이스(RPI, Radio Programming Interface)를 참조하는 단계 및 상기 라디오 프로그래밍 인터페이스에서 참조한 라디오 가상 머신의 계층 구조에 따라 라디오 가상 머신의 적어도 하나의 추상화 처리 엘리먼트(APE, Abstract Processing Element)를 적어도 하나의 엘리먼트리 라디오 가상 머신(eRVM, elementary RVM)으로 대체하는 단계를 포함한다.

### 발명의 효과

- [15] 상술한 바와 같은 본 발명에 따른 라디오 어플리케이션을 실행하는 방법 및 단말 장치를 이용하면, 라디오 어플리케이션 제공자가 제공하는 다양한 라디오 어플리케이션을 하드웨어 플랫폼에 독립적으로 단말 장치에 설치하여 이용하는 것이 가능해진다.
- [16] 또한, 이동통신 사업자의 측면에서는, 필요에 따라 자신의 망 가입자들이 사용하고 있는 다양한 라디오 플랫폼을 구비한 단말 장치를 원하는 통신망 규격으로 전환하는 것이 가능해지므로, 유연성 있는 망 운영이 가능해진다.
- [17] 또한, 사용자의 측면에서는 새로운 라디오 통신 방식으로 전환하고자 하는 경우에 새로운 모바일 장치를 구입할 필요 없이 새로운 라디오 어플리케이션 패키지를 다운로드 받아 자신의 단말 장치에 설치하는 것만으로 새로운 통신망을 사용하는 것이 가능해진다.

### 도면의 간단한 설명

- [18] 도 1은 본 발명의 일 실시예에 따른 단말 장치에서 온라인 앱 스토어(On-Line App. Store)에서 배포되는 라디오 어플리케이션 패키지를 다운로드하는 경우를

설명하기 위한 예시도이다.

[19] 도 2는 본 발명의 일 실시예에 따른 단말 장치의 소프트웨어 아키텍처를 설명하기 위한 블록도이다.

[20] 도 3은 도 2의 단말 장치의 라디오 재구성을 위한 사용 환경에서 단말 장치와 관련된 인터페이스들을 설명하기 위한 개념도이다.

[21] 도 4는 표준 기능 블록(SFB, Standard Function Block)을 도시한 개념도이다.

[22] 도 5는 라디오 가상 머신(RVM, Radio Virtual Machine) 처리에 관한 다이어그램(Diagram)을 도시한 개념도이다.

[23] 도 6은 본 발명의 일 실시예에 따른 엘리먼트리 RVM(eRVM)을 도시한 개념도이다.

[24] 도 7은 데이터 오브젝트 및 데이터 오브젝트의 인터페이스를 도시한 개념도이다.

[25] 도 8은 APE 및 APE의 인터페이스를 도시한 개념도이다.

[26] 도 9는 추상적 스위치 구조(Abstract Switch Fabric)를 도시한 개념도이다.

[27] 도 10은 RVM 계층 구조를 도시한 개념도이다.

[28] 도 11은 eRVM의 수평 확장을 도시한 개념도이다.

[29] 도 12는 RVM 입력 파일을 생성하기 위한 프로세싱 방법을 도시한 흐름도이다.

[30] 도 13은 구성코드를 위한 바이너리 포맷을 도시한 개념도이다.

[31] 도 14는 특정 데이터 오브젝트의 구성코드 포맷을 도시한 개념도이다.

[32] 도 15는 데이터 오브젝트 세트 코드의 포맷을 도시한 개념도이다.

[33] 도 16은 데이터 오브젝트 초기화 코드의 포맷을 도시한 개념도이다.

[34] 도 17은 특정 추상적 처리 추상화 처리 엘리먼트 구성코드(APE Configcode)의 포맷을 도시한 개념도이다.

[35] 도 18은 ASF Configcode의 포맷을 도시한 개념도이다.

[36] 도 19는 구성코드들로 구성되는 XML 엘리먼트들을 도시하는 블록도이다.

[37] 도 20은 도 2의 단말 장치에서 라디오 컴퓨터의 소프트웨어 아키텍처를 설명하기 위한 블록도이다.

[38] 도 21은 도 2의 단말 장치에서 통합 라디오 어플리케이션의 동작 구조의 일 예를 설명하기 위한 계층 구조도이다.

[39] 도 22는 도 2의 단말 장치에서 통합 라디오 어플리케이션의 동작 구조의 다른 예를 설명하기 위한 계층 구조도이다.

[40] 도 23은 도 2의 단말 장치의 라디오 플랫폼의 기능 블록 라이브러리 구현들을 설명하기 위한 개념도이다.

[41] 도 24는 도 2의 단말 장치에서 다운로드할 수 있는 라디오 어플리케이션 패키지의 구성예를 설명하기 위한 블록도이다.

[42] 도 25는 도 2의 단말 장치의 아키텍처를 개략적으로 나타낸 블록도이다.

[43] 도 26은 본 실시예에 따른 표준 기능 블록들을 결정하는 방법에 대한 흐름도이다.

- [44] 도 27은 본 실시예에 따른 방법에 있어서 네이티브 라디오 라이브러리를 생성하는 과정을 나타낸 흐름도이다.

#### **발명의 실시를 위한 최선의 형태**

- [45] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따르면, 라디오 어플리케이션 실행 방법은, 재구성 가능 모바일 장치의 라디오 가상 머신(RVM, Radio Virtual Machine)에서 라디오 프로그래밍 인터페이스(RPI, Radio Programming Interface)를 참조하는 단계 및 상기 라디오 프로그래밍 인터페이스에서 참조한 라디오 가상 머신의 계층 구조에 따라 라디오 가상 머신의 적어도 하나의 추상화 처리 엘리먼트(APE, Abstract Processing Element)를 적어도 하나의 엘리먼트라 라디오 가상 머신(eRVM, elementary RVM)으로 대체하는 단계를 포함한다.
- [46] 본 발명의 다른 실시예에 따르면, 재구성 가능 모바일 장치 즉, 단말 장치는, 라디오 컴퓨터 또는 라디오 프로세서 상에서 동작하며 라디오 프로그래밍 인터페이스를 참조하는 라디오 가상 머신, 및 상기 라디오 프로그래밍 인터페이스에서 참조한 라디오 가상 머신의 계층 구조에 따라 라디오 가상 머신에 의해 라디오 가상 머신의 적어도 하나의 추상화 처리 엘리먼트(APE, Abstract Processing Element)와 대체되는 적어도 하나의 엘리먼트라 라디오 가상 머신(elementary RVM)을 포함한다.

#### **발명의 실시를 위한 형태**

- [47] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세한 설명에 상세하게 설명하고자 한다. 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 각 도면을 설명하면서 유사한 참조부호를 유사한 구성요소에 대해 사용하였다.
- [48] 제1, 제2, A, B 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 본 발명의 권리 범위를 벗어나지 않으면서 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.
- [49] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는

것으로 이해되어야 할 것이다.

- [50] 본 출원에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함한다" 또는 "가진다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [51] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥 상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.
- [52] 본 발명을 설명하기 위해서 전반적으로 사용되는 용어들에 대한 간략한 정의들을 정리한다. 아래 용어들 이외의 용어들에 대해서는 본 명세서 내의 적절한 부분에서 정의를 제공한다.
- [53] 라디오 어플리케이션(RA, Radio Application): 단말 장치의 구체적인 하드웨어 구성 및 사용자 어플리케이션에 비의존적인 라디오 통신 환경을 제공하기 위한 어플리케이션이다. 라디오 어플리케이션은 라디오 프로세서 상에서 동작하거나, 라디오 프로세서 실행 부분과 어플리케이션 프로세서 실행 부분으로 구성되어 두 개의 프로세서 상에서 동작하도록 구성될 수 있다. 라디오 어플리케이션은 라디오 컨트롤러 및 기능 블록들로 구성된다. 기능 블록에는 표준 기능 블록과 사용자 정의 기능 블록이 포함될 수 있다.
- [54] 라디오 어플리케이션 패키지(RAP, Radio Application Package): 라디오 어플리케이션의 배포 형태로서, 라디오 어플리케이션의 구성요소인 라디오 컨트롤러, 기능 블록과 함께 파이프라인 구성 메타 데이터를 포함한다. 또한, 라디오 어플리케이션 패키지는 추가적으로 라디오 라이브러리(Radio Library)를 포함할 수 있다.
- [55] 표준 기능 블록(SBF, Standard Function Block): 표준 기능 블록은 각 블록의 기능과, 해당 블록을 실행하기 위한 함수의 이름이 표준화된 표준적 기능 블록들이다. 표준 기능블록은 라디오 플랫폼 칩 벤더가 표준기능 블록을 제작하는 경우 하드웨어 제조사가 구현한 표준 기능 블록의 집합체가 될 것이며 드라이버와 함께 제공될 수 있다. 표준 기능 블록은 전용의 하드웨어 가속기(Hardware Accelerator)를 이용하여 구현되거나, 라디오 프로세서의 코어에서 동작하는 실행 코드로 구현될 수 있다. 라디오 프로세서의 코어에서 동작하는 실행 코드로 구현될 경우 라디오 라이브러리(Radio Library)로 지칭할 수 있다. 표준기능블록들은 각 함수의 이름과 기능이 표준화 되어 있으며, 표준

라디오 라이브러리 헤더(Standard Radio Library Header) 파일에 의해서 정의될 수 있다.

- [56] 사용자 정의 기능 블록(UDFB, User Defined Function Block): 표준 기능 블록으로 제공되지 않거나, 표준 기능 블록으로 존재하는 기능을 보다 커스터마이징할 필요성이 있는 경우, 라디오 어플리케이션 제공자에 의해서 제공될 수 있는 기능블록들로서, 라디오 프로세서의 코어에서 실행되도록 구현될 수 있다. 사용자 정의 기능 블록은 실행 코드, 소스 코드, 중간 표현 형태의 코드로 제공될 수 있다.
- [57] 사용자 정의 기능 블록(UDFB, User Defined Function Block) 집합: 라디오 어플리케이션 제공자가 제공한 사용자 정의 기능 블록의 집합체일 수 있다.
- [58] 라디오 HAL(Hardware Abstract Layer): OS의 관점에서 많은 종류의 HW를 추상화하는 계층이다. 표준화된 추상화 가속기 인터페이스는 하드웨어에 독립적이지만 HAL 때문에 모든 하드웨어에 OS가 접근할 수 있게 된다. 드라이버의 역할과 비슷하지만, 하드웨어가 바뀔에 따라 바뀌는 드라이버와는 달리 HAL은 OS에 포함되어 있다.
- [59] 라디오 플랫폼 드라이버(Radio Platform Driver): OS가 하드웨어를 인식하기 위해 필요한 소프트웨어이다. 하드웨어에 독립적인 OS의 명령어로서 하드웨어의 명령어 체계를 서로 매칭시키기 위한 소프트웨어이며 일반적인 하드웨어 드라이버의 역할을 할 수 있다.
- [60] 라디오 플랫폼(Radio Platform): 고정된(Fixed) 및/또는 프로그래머블한(Programmable) 하드웨어 가속기(Hardware Accelerator), RF 트랜시버, 안테나와 같은 RF 기능에 관한 모바일 장치 하드웨어의 일부일 수 있다. 즉, 라디오 플랫폼은 RF 신호를 생성하거나 수신할 수 있는 하드웨어의 일부일 수 있다. 본질적으로, 라디오 플랫폼은 예를 들어 주문형 집적 회로(ASIC)와 같은 고정 가속기(Fixed Accelerator)나 FPGA와 같은 재구성 가능 가속기(Reconfigurable Accelerator) 같은 다른 프로세싱 요소를 포함하는 이기종(Heterogeneous) 하드웨어일 수 있다.
- [61] 라디오 컴퓨터(Radio Computer): 재구성가능(Reconfigurable) 모바일 장치(MD, Mobile Device) 내에서 라디오 프로세서(RP, Radio Processor) 및 라디오 플랫폼의 컬렉션일 수 있다. 모바일 장치에서는, 개별 라디오 어플리케이션들이 보다 범용의 컴퓨팅 요소로 볼 수 있는 라디오 프로세서 상에서 실행하는 소프트웨어 요소로서 설계될 수 있다.
- [62] 이하, 본 발명에 따른 바람직한 실시예를 첨부한 도면들을 참조하여 상세하게 설명한다. 본 발명을 설명함에 있어 전체적인 이해를 용이하게 하기 위하여 도면상의 동일한 구성요소에 대해서는 동일한 참조부호를 사용하고 동일한 구성요소에 대해서 중복된 설명은 생략한다.
- [63]
- [64] 도 1은 본 발명의 일 실시예에 따른 단말 장치에서 온라인 앱 스토어(On-Line

App. Store)에서 배포되는 라디오 어플리케이션 패키지를 다운로드하는 경우를 설명한 예시도이다.

- [65] 도 1을 참조하면, 본 실시예에 따른 단말 장치(10)는 라디오 어플리케이션 스토어 또는 라디오 어플리케이션 제공자로부터 라디오 어플리케이션을 다운로드 받아 설치하여 다양한 무선 방식의 네트워크 환경에서 동작할 수 있다.
- [66] 즉, 단말 장치(10)는 온라인 앱 스토어(20)에 접속하고, 앱 스토어가 제공하는 다양한 무선 통신 방식들을 지원하는 라디오 어플리케이션들의 리스트에서 원하는 라디오 어플리케이션을 선택하고, 해당 라디오 어플리케이션을 포함하는 라디오 어플리케이션 패키지를 다운로드 받을 수 있다.
- [67] 다양한 무선 방식은 LTE(Long Term Evolution), WCDMA(Wideband Code Division Multiple Access), WiMAX (Worldwide Interoperability for Microwave Access), GSM(Global System for Mobile Communications), RFID(Radio-Frequency Identification) 등을 포함할 수 있다. 사용자는 자신의 단말에 복수의 라디오 어플리케이션을 다운로드 받아 설치해두고 상황에 따라 필요한 라디오 어플리케이션을 자유롭게 실행할 수 있다.
- [68]
- [69] 도 2는 본 발명의 일 실시예에 따른 단말 장치의 소프트웨어 아키텍처를 설명하기 위한 블록도이다.
- [70] 도 2를 참조하면, 본 실시예에 따른 단말 장치의 라디오 소프트웨어 아키텍처는 어플리케이션 프로세서(AP, Application Processor)(110) 상에서 동작하는 어플리케이션 프로세서 계층과 라디오 프로세서(RP, Radio Processor)(610) 상에서 동작하는 라디오 프로세서 계층으로 구성될 수 있다. 라디오 프로세서와 라디오 플랫폼의 조합은 라디오 컴퓨터(120)로 지칭될 수 있다.
- [71] 즉, 도 2는 후술되는 라디오 컨트롤 프레임워크(RCF, Radio Control Framework)(126)가 어플리케이션 프로세서 실행 부분과 라디오 프로세서 실행 부분으로 분리되어 두 개의 프로세서 상에서 동작하는 소프트웨어 아키텍처 환경을 예시한다. 물론, 라디오 컨트롤 프레임워크(126)는 라디오 프로세서나 라디오 운영체제(OS, Operating System)(121)에서 실행되도록 구현될 수 있다.
- [72] 어플리케이션 프로세서(110) 상에는 구글(Google)의 안드로이드(Android) OS와 애플(Apple)의 iOS와 같은 비 실시간(Non-Real Time) 운영체제(OS, Operating Systems)가 동작하거나, 라디오 프로세서 상에는 이하에서 라디오 OS(121)라 지칭되는 실시간 운영체제(Real Time OS)가 동작할 수 있다. 명확한 구별을 위하여 어플리케이션 프로세서 계층에서 동작하는 비 실시간 운영체제를 '운영체제(OS)', 라디오 프로세서 계층에서 동작하는 실시간 운영체제는 '실시간 운영체제(Radio OS)'로 구분하여 명명하기로 한다.
- [73] 어플리케이션 프로세서 계층, 라디오 프로세서 계층 및 라디오 컨트롤 프레임워크(126)를 구성하는 구성요소들에 대하여 좀 더 상세히 설명하면 다음과 같다.

- [74]
- [75] 어플리케이션 프로세서
- [76] 어플리케이션 프로세서(110)는 도 2에 도시한 바와 같이, 드라이버(111), 운영체제(OS)(112) 및 통신 서비스 계층(113) 등의 구성요소들을 포함할 수 있다.
- [77] 드라이버(111)는 주어진 운영체제(112) 상에서 하드웨어 디바이스들을 구동한다. 하드웨어 디바이스들은 카메라, 스피커 등을 포함할 수 있다.
- [78] 운영체제(112)는 안드로이드, iOS와 같이 통상적인 모바일 디바이스에서 동작하는 비 실시간 운영체제(Non-Real Time OS)를 포함할 수 있다. 라디오 컨트롤 프레임워크(126)가 어플리케이션 프로세서(110)와 라디오 프로세서(610)에서 동작하는 구성일 때, 운영체제(112) 상에는 라디오 컨트롤 프레임워크(126)의 어플리케이션 프로세서 계층 실행 부분이 존재할 수 있다.
- [79] 통신 서비스 계층(113)은 후술되는 3가지 서비스들 중 적어도 일부를 라디오 컨트롤 프레임워크(126)에 제공할 수 있다.
- [80] 첫 번째 서비스는, 관리(Administrative)에 관련된 서비스로서 라디오 어플리케이션의 인스톨/언인스톨, 인스턴스(Instance)의 생성/삭제 그리고 인스톨, 인스턴스, 활동 등의 상태(Status)에 대한 라디오 어플리케이션의 목록 획득에 관련된 서비스이다.
- [81] 두 번째 서비스는, 접속 제어에 관련된 서비스로서 라디오 어플리케이션의 실행/비실행, 데이터 흐름 생성, 네트워크 할당 생성 그리고 각 인스톨, 인스턴스, 활동 등의 상태에 대한 라디오 어플리케이션의 목록 획득에 관련된 서비스이다.
- [82] 마지막으로 세 번째 서비스는, 데이터 흐름에 관련된 서비스로서 사용자 데이터를 보내고 받는데 관련된 서비스이다.
- [83] 상술된 세가지 서비스들 중 적어도 일부 서비스를 제공하기 위한 통신 서비스 계층(113) 구성의 하나의 예로서, 통신 서비스 계층은 관리자(Administrator) 어플리케이션, 이동성 정책(Mobility Policy) 매니저, 네트워킹 스택(Networking Stack) 및 모니터(Monitor) 중 적어도 일부를 포함하는 적어도 하나 이상의 어플리케이션으로 구현될 수 있다. 네트워킹 스택은 통신 서비스 계층에서 동작하는 프로토콜 스택(Protocol Stack)을 포함할 수 있다.
- [84] 한편, 통신 서비스 계층(113)은 상술된 구성요소들 중 일부만 포함할 수도 있고, 상술된 구성요소들 이외의 추가적인 구성요소를 포함할 수도 있다. 또한, 통신 서비스 계층(113) 내에는 적어도 둘 이상의 상술된 구성요소들의 기능이 통합된 구성요소들이 포함될 수도 있다. 또한, 상술한 구성요소들은 통신 서비스 계층(113)이 수행하여야 하는 서비스들을 지원하기 위하여 통신 서비스 계층(113)이 구비하여야 하는 구성요소들의 일 예일뿐이다. 즉, 통신 서비스 계층(113)은 통신 서비스 계층(113)이 수행하는 역할에 의해서 정의되며, 상술된 구성요소들의 예시에 의해서 통신 서비스 계층(113)의 구성이 한정되는 것은 아니다.
- [85] 라디오 컨트롤 프레임워크(126)가 어플리케이션 프로세서(110)와 라디오

프로세서(610)에서 동작하는 구성에서, 본 실시예에 따른 단말 장치(10)로의 배포, 설치 및 실행 방법의 대상이 되는 라디오 어플리케이션들은 각각 어플리케이션 프로세서 계층 실행 부분과 라디오 프로세서 계층 실행 부분들로 구성될 수 있다. 라디오 어플리케이션의 어플리케이션 프로세서 계층 실행 부분인 라디오 컨트롤러(RC, Radio Controller)는 통신 서비스 계층(Communication Service Layer)(113)의 모니터에게 상황 정보(Context Information)를 보내거나 통신 서비스 계층(113)의 네트워킹 스택과 데이터를 주고 받는 역할을 수행할 수 있다.

[86]

[87]     라디오 컴퓨터

[88]     라디오 프로세서(610)는 도 2에 도시한 바와 같이 라디오 운영체제(121), 라디오 플랫폼 드라이버(122) 및 라디오 플랫폼(123)을 포함한다.

[89]     라디오 운영체제(OS)(121)는 실시간 운영체제이다. 라디오 컨트롤 프레임워크(126)가 어플리케이션 프로세서(110)와 라디오 프로세서(610)에서 동작하는 구성일 때, 라디오 OS(121) 상에는 라디오 컨트롤 프레임워크(126)의 라디오 프로세서 실행 부분이 존재할 수 있다.

[90]     라디오 플랫폼 드라이버(Radio Platform Driver)(122)는 일반적인 하드웨어 드라이버와 같이 하드웨어 라디오 플랫폼을 인식하기 위해서 라디오 OS(121)에 의해서 요구되는 구성요소이다.

[91]     라디오 컨트롤 프레임워크(126)가 라디오 프로세서(610)에서만 동작하는 구성(도 3 참조)이면, 본 실시예에 따른 단말 장치(10)로의 배포, 설치 및 실행의 대상이 되는 재구성가능 라디오 어플리케이션들은 라디오 프로세서 계층에서 동작할 수 있다.

[92]     각각의 라디오 어플리케이션의 라디오 컨트롤러(RC, Radio Controller)는 통신 서비스 계층(Communication Service Layer)(113)의 모니터에게 상황 정보(Context Information)를 보내거나 통신 서비스 계층(113)의 네트워킹 스택과 데이터를 주고 받는 역할을 수행한다.

[93]     전술한 라디오 프로세서(610)는 라디오 플랫폼(123) 또는 라디오 플랫폼 하드웨어와 함께 라디오 컴퓨터(120)를 구성할 수 있다. 여기서, 라디오 플랫폼 하드웨어(Radio Platform Hardware)는 일반적으로 라디오 프로세서의 프로그래머블 하드웨어(123-1)와 베이스밴드 가속기(들)(123-2)로 구성될 수 있다. 표준 기능 블록(들)(517-1~M)을 위해서 준비되는 베이스밴드 가속기(123-2)는 종종 ASIC(Application-Specific Integrated Circuit)의 형태로 제공될 수 있다. 또한, 라디오 플랫폼(123)은 RF 송수신기(RF Transceiver)(123-3) 및 안테나(123-4)를 포함할 수 있다.

[94]     전술한 통신 서비스 계층(113)과 컨트롤 라디오 컨트롤 프레임워크(126) 사이에는 멀티라디오 인터페이스(MURI, Multi Radio Interface)가 배치될 수 있고, 라디오 컨트롤 프레임워크(126)와 통합 라디오 어플리케이션(125) 사이에는

통합 라디오 어플리케이션 인터페이스(URAI, Unified Radio Application Interface)가 배치될 수 있다. 또한, 통합 라디오 어플리케이션(125)과 RF 송수신기(123-3) 사이에는 재구성가능 RF 인터페이스(RRFI, Reconfigurable Radio Frequency Interface)가 배치될 수 있다.

- [95] 라디오 어플리케이션은 모바일 단말의 통신을 가능하게 하는 어플리케이션으로써 라디오 어플리케이션 패키지(RAP, Radio Application Package)(510) 형태로 배포될 수 있다. 라디오 어플리케이션 패키지(510)는 기능 블록(FB, Function Block), 파이프라인 구성 메타데이터(Metadata)(513), 라디오 컨트롤러 코드(RC code)(512) 및 라디오 라이브러리(Radio Library)(514)의 구성요소들을 포함할 수 있다.
- [96] 라디오 라이브러리(514)는 표준 기능 블록(517-1)이 실행코드 형태로 배포되는 경우에 라디오 어플리케이션 패키지(510)에 실행 코드 형태로 함께 포함되어 배포될 수 있고, 라디오 어플리케이션 패키지(510)는 어플리케이션 프로세서(110)의 OS(112)에 다운로드되고, 구성코드(Configcodes)와 라디오 라이브러리(514)는 파이프라인 구성 메타 데이터(513)를 참조하여 어플리케이션 프로세서(110)에서 라디오 프로세서(610)나 라디오 OS(121)로 로딩되는 과정을 거쳐 라디오 컴퓨터에 탑재될 수 있다.
- [97]
- [98] 재구성 가능 RF 인터페이스의 소프트웨어 아키텍처
- [99] 도 3은 본 발명의 일 실시예에 따른 재구성가능(Reconfigurable) 단말 장치의 사용 환경에서 단말 장치와 관련된 4가지 인터페이스들을 설명하기 위한 개념도이다.
- [100] 도 3을 참조하면, 본 실시예에 따른 재구성 가능한 모바일 장치(Reconfigurable Mobile Device)는 다중/다중 라디오(Multiple Radios)를 동시에 실행할 수 있고 새로운 라디오 어플리케이션 패키지(RAP, Radio Application Package)(510)에 의해 라디오들의 설정을 변경할 수 있다. 모든 라디오 어플리케이션들(RAs)은 모바일 장치의 라디오 재구성과 관련된 요구사항의 관점에서 공통 속성이나 특성을 나타낼 때 통합 라디오 어플리케이션(URAs, Unified Radio Applications)으로 지칭될 수 있다.
- [101] 다중/다중 통합 라디오 어플리케이션을 실행하기 위하여, 재구성 가능한 모바일 장치(이하, 간략히 모바일 장치라고도 한다)는 통신 서비스 계층(CSL, Communication Services Layer)(113), 라디오 컨트롤 프레임워크(RCF, Radio Control Framework)(126), 라디오 플랫폼(Radio Platform)(123) 및 이들의 상호 연결을 위한 4 세트의 인터페이스들을 포함할 수 있다.
- [102] 본 실시예에 따른 재구성가능 모바일 장치(MD, Mobile Device)와 관련된 4가지 인터페이스는 통신 서비스 계층(113)의 각 구성요소와 라디오 컨트롤 프레임워크(126)의 각 구성요소 사이의 인터페이스인 멀티라디오 인터페이스(MURI, Multi Radio Interface), 통합 라디오 어플리케이션(URA,

Unified Radio Application)(125)과 RF 송수신기(Radio Frequency Transceiver)(123-3) 간의 인터페이스인 재구성 가능 RF 인터페이스(RRFI, Reconfigurable Radio Frequency Interface), 통합 라디오 어플리케이션(125)과 라디오 컨트롤 프레임워크(126)의 각 구성요소 사이의 인터페이스인 통합 라디오 어플리케이션 인터페이스(URAI, Unified Radio Application Interface), 라디오 어플리케이션의 독립적이고 정형적인(Uniform) 생산에 관련된 인터페이스인 라디오 프로그래밍 인터페이스(RPI, Radio Programming Interface)를 포함할 수 있다.

- [103] 라디오 프로그래밍 인터페이스(RPI)는 라디오 어플리케이션 제공자와 단말 장치와의 사이 또는 라디오 앱 스토어와의 사이에 정의되는 인터페이스로서, 라디오 어플리케이션 제공자는 이러한 RPI를 참조하여 단말 장치에서 실행되는 라디오 재구성 가능 어플리케이션을 제작하여 배포할 수 있다.

[104]

[105] 라디오 컨트롤 프레임워크

- [106] 전술한 라디오 컨트롤 프레임워크(126)를 좀 더 구체적으로 설명하면 다음과 같다.

- [107] 라디오 컨트롤 프레임워크(RCF, Radio Control Framework)(126)는 라디오 어플리케이션의 동작 환경을 제공하는 구성요소이다.

- [108] 라디오 컨트롤 프레임워크(126)가 어플리케이션 프로세서(110)와 라디오 프로세서(610)에서 동작하는 구성이라면, 라디오 컨트롤 프레임워크(126)는 두 개의 그룹으로 나누어질 수 있다. 즉, 하나의 그룹은 어플리케이션 프로세서(110) 상에서 동작하며, 다른 하나의 그룹은 라디오 프로세서(610) 상에서 동작한다. 라디오 컨트롤 프레임워크(126)의 어떠한 구성요소가 실시간으로 동작(라디오 프로세서 상에서 동작)하고 어떠한 구성요소가 비 실시간으로 동작(어플리케이션 프로세서 상에서 동작)할 것인지는 각각의 벤더에 따라서 다르게 결정될 수 있다.

- [109] 라디오 컨트롤 프레임워크(RCF, Radio Control Framework)(126)는 기본적으로 다음의 5가지 구성요소들의 적어도 일부를 포함하여, 라디오 어플리케이션(들)을 관리하도록 구성될 수 있다.

- [110] 설정 매니저(CM, Configuration Manager)는 멀티라디오 단말 장치에 대한 라디오 어플리케이션의 인스톨/언인스톨, 인스턴스(Instance)의 생성/삭제와 라디오 어플리케이션들의 라디오 파라미터들에 대한 액세스 관리를 수행한다.

- [111] 라디오 연결 매니저(RCM, Radio Connection Manager)는 사용자 요구들에 따른 라디오 어플리케이션들의 활성화/비활성화 및 하나의 라디오 어플리케이션으로부터 다른 라디오 어플리케이션으로 스위칭될 수 있는 사용자 데이터 흐름의 전체적인 관리를 수행한다.

- [112] 플로우 컨트롤러(FC, Flow Controller)는 사용자 데이터 패킷의 송신과 수신 및 흐름 제어를 수행한다.

- [113] 멀티라디오 컨트롤러(MRC, Multi Radio Controller)는 라디오 어플리케이션들간의 상호 운용성 문제를 미리 감지하기 위해서, 동시에 실행되는 라디오 어플리케이션들로부터 제기되는 라디오 자원들(Radio Resources)에 대한 요구들을 스케줄링한다.
- [114] 리소스 매니저(RM, Resource Manager)는 실시간 요구사항을 충족시키면서 동시에 활성화된 라디오 어플리케이션들간에 멀티라디오 자원들을 공유하기 위한 멀티라디오 자원들의 관리한다.
- [115] 한편, 라디오 컨트롤 프레임워크(126)는 전술되는 5가지 구성요소들 중 일부만을 포함할 수도 있고, 5가지 구성요소 이외의 구성요소를 더 포함할 수도 있다. 또는, 라디오 컨트롤 프레임워크(126)는 적어도 둘 이상의 후술되는 구성요소들의 기능이 통합된 구성요소들로 구성될 수도 있다. 라디오 컨트롤 프레임워크(126)의 기능 및 역할은 후술되는 구성요소들이 수행하는 기능에 의해서 정의되며, 후술되는 예시적 구성요소들에 의해서 라디오 컨트롤 프레임워크(126)의 구성이 한정되는 것은 아니다. 즉, 라디오 컨트롤 프레임워크(126)는 후술되는 구성요소들의 기능들 중 적어도 일부를 수행하기 위한 다양한 구성을 가질 수 있다.
- [116]
- [117] 도 4는 표준 기능 블록(SFB, Standard Function Block)을 도시한 개념도이다.
- [118] 도 4를 참조하면, 재구성가능(Reconfigurable) 통신 시스템을 위한 라디오 어플리케이션 코드(Radio Application Code)를 작성하는 단계에서 사용할 수 있는 SFB(517-1~M)들의 구성요소에 관한 내용이다.
- [119] 재구성 가능한 단말기(혹은 멀티모드 단말기)에 사용 되는 라디오 어플리케이션(RA, Radio Application)은 여러 가지 통신 규격에 맞게 구성될 수 있다. RA란 멀티모드 단말기에서 동작할 수 있는 소프트웨어 코드로서, CDMA, WCDMA, HSPA, LTE, LTE-A, Wi-Fi 등 현재 존재하는 모든 이동통신 표준규격을 따를 수 있도록 제작된 소프트웨어일 수 있다. 이런 RA를 제작하는 단계에서 사용될 수 있는 표준 라이브러리인 레퍼런스 라디오 라이브러리(Reference Radio Library)(220)가 존재하며 이 레퍼런스 라디오 라이브러리(220)는 SFB(517-1~M)을 포함할 수 있다.
- [120] 위에서 언급한 SFB(517-1~M)는 여러 가지 통신규격을 만족하기 위하여 작성된 여러 개의 RA에서, 공통적으로 사용되는 일반적인 기능들이고, 또한 어떤 라디오 컴퓨터(멀티모드 단말기)에서도 실행할 수 있게 보장하기 위해 엘리먼트리 표준 기능 블록(eSFB, Elementary SFB)의 조합으로 구현될 수 있다.
- [121] eSFB란, 레퍼런스 라디오 라이브러리(Reference Radio Library)(220)에 포함되어 있으며, SFB(517-1~M)의 일종일 수 있다. eSFB의 특징은 더 이상 작은 단위로 나눌 수 없다는 것이다. 즉 SFB들도 여러 가지의 단위(Granularity)가 존재 할 것인데(크게는 LTE, 작게는 Adder, Multiplier 등등), eSFB는 그 중에서 가장 작은 단위의 SFB들을 지칭할 수 있다. 그리고 위에 언급했듯이, 도 4에서 보는 바와

- 같이 eSFB를 제외한 나머지 SFB들은 모두 eSFB의 조합으로 표현될 수 있다.
- [122] eSFB는 사칙연산 혹은 비트연산 혹은 논리연산 등등 가장 기본적인 연산들로 구성 될 수 있다. eSFB는 SFB를 구성하는 요소로서 라디오 가상 머신(RVM, Radio Virtual Machine)(515)의 APE(460)에 추상화(Abstract)될 수 있는 가장 작은 단일 객체를 의미할 수 있다.
- [123] 레퍼런스 라디오 라이브러리(Reference Radio Library)(220)에 포함되는 SFB(517-1~M)는 일정한 기준에 의하여 선정될 수 있다.
- [124]
- [125] 도 5는 라디오 가상 머신(RVM, Radio Virtual Machine) 처리에 관한 다이어그램(Diagram)을 도시한 개념도이다.
- [126] 도 5를 참조하면, RVM(515)은 구성코드(Configcode)를 실행할 수 있는 추상적 머신일 수 있다. 또한 RVM은 하드웨어와 독립적일 수 있다. RVM의 구현은 특정 라디오 컴퓨터를 대상으로 할 수 있다. 그리고, RVM은 실행 코드로 구성코드를 편집하기 위한 Just-in-time(JIT)과 Ahead-of-time(AOT) 방법을 제공하는 백-엔드(Back-End) 컴파일러를 포함할 수 있다.
- [127] RVM(515)은, 데이터 플로우 차트 형식으로 표시되는 특정 알고리즘을 실행하는 추상적 머신일 수 있다. 다시 말해서, RVM(515)은 APEs(460) 및 데이터 오브젝트(DO, Data Object)(470)의 특정 데이터 플로우 차트 내에서 모든 연산자 및 토큰을 각각 대체한 결과일 수 있다.
- [128] 각 APE(460)는 대체 오퍼레이터 식별자에 의해 표시된 계산을 실행한다. 이 계산은 표준 라디오 라이브러리(Radio Library)로부터 각 APE(460)로 수신될 수 있다. APE(460)는 특정 데이터 플로우 차트에서 동작에 대응하는 연산 리소스를 추상화할 수 있다.
- [129] 데이터 오브젝트(470)는 메모리의 리소스를 추상화한다. 즉, 데이터 오브젝트(470)는 라디오 처리 과정 중에 사용되는 데이터를 저장하기 위한 추상적인 메모리일 수 있다. 표준 라디오 라이브러리는 모든 표준 기능 블록(SFB, Standard Functional Block)의 규범에 대한 설명을 포함할 수 있다.
- [130] 라디오 라이브러리에 포함 된 계산은 라디오 라이브러리(514)가 프론트-엔드 또는 백-엔드 편집에 각각 사용되는지 여부에 따라 규범 설명이나 SFBs의 네이티브 구현의 관점에서 표현될 수 있다.
- [131] 사용자 정의 기능 블록(UDFB, User Defined Functional Block)(516)은 SFB들의 결합을 통해 생성될 수 있다. UDFBs(516)은 RVM(515)에서 실행되는 데이터 흐름도로서 표현될 수 있다.
- [132] 이와 달리, UDFBs(516)은 독립형 모듈 또는 독립형 기능으로 구현될 수 있다. UDFBs(516)은 라디오 응용 프로그램 패키지의 일부일 수 있다. RVM(515)은 일부 데이터 오브젝트(470) 초기화 후 즉시 작동하기 시작할 수 있다. 모든 APEs(460)은 비동기적으로, 동시에 계산을 실행할 수 있다. 모든 입력 데이터 오브젝트(470)가 가득찰 경우, 해당 APEs(460)은 할당 연산자를 실행할 수 있다.

- [133] APEs(460)은 데이터 오브젝트(470)에 "읽기", "읽기-지우기", "쓰기" 연산자를 이용하여 접근할 수 있다. 데이터 오브젝트(470)로부터 입력 데이터를 읽은 후, 출력 데이터 오브젝트(470)가 비어있는 경우, APE(460)는 할당 연산자를 실행하고, 다음 APE(460)는 처리된 데이터를 기록할 수 있다. 가득 차있는 출력 데이터 오브젝트(470)는 쓰기에 해당하는 작동을 막을 수 있다. RVM(515)은 모든 APEs(460)이 비활성화인 상태에 도달할 때까지 계산을 실행할 수 있다.
- [134] 이 상태에서, 비활성 연산자를 활성화 시킬 수 있는 충분히 가득 차 있는 데이터 오브젝트(470)가 없을 수 있다. 예를 들어, 연산자 "N"입출력 데이터 오브젝트(470)와 " $K < N$ " 입력 데이터 오브젝트(470)는 가득 차 있을 수 있다. 결과적으로, APE(460)는 다시 활성화 되지 않을 수 있다. 계산의 결과는 비활성 연산자를 활성화 할 수 없는 가득 찬 데이터 오브젝트(470)일 수 있다.
- [135] 출력 데이터 오브젝트(470)는 후속 연산자를 위한 입력 데이터 오브젝트(470)가 될 수 있다. 그런 다음, 입력 데이터 오브젝트(470)는 후속 연산자를 활성화 시킬 수 있다. APE(460)로부터 수신한 상태 또는 작동은 다른 APE(460)의 상태에 독립적일 수 있다.
- [136] 특정 RVM은 APE 풀(Abstract Processing Element Pool)(211), 데이터 오브젝트 풀(Data Object Pool)(212) 등의 추상화 리소스(Abstract Resources)에서 제공되는 리소스를 RVM 프로그램을 통해 처리할 수 있다. 또한, 특정 RVM은 라디오 라이브러리(Radio Library)(514) 등의 작업 기초(Operation Basis)로 제공되는 기능성(Functionality) RVM 프로그램을 통해 처리할 수 있다.
- [137] RVM 프로그램은 초기 데이터 오브젝트(Initialized Data Object)를 읽고 FIR(Finite Impulse Response) 및/또는 IFFT(Inverse Fast Fourier Transform)를 수행하는 필터 또는 추상화 처리 엘리먼트(APE, Abstract Processing Element)(460)를 통해 초기 데이터 오브젝트를 처리한 후 데이터 오브젝트(DO, Data Object)(470)로 기록하고, FIR 처리된 데이터 오브젝트(470)와 IFFT 처리된 데이터 오브젝트(470)의 상관관계에 따라 최종 처리 결과를 기록(추상화)할 수 있다.
- [138] 이러한 특정 RVM에 의하면, 라디오 어플리케이션은 서로 다른 분할 정도(Granularity)의 모듈들을 포함할 수 있다. 이러한 모듈들은 라디오 라이브러리(514)로부터 얻어지거나 라디오 라이브러리 엘리먼트들로 구성될 수 있다. 또한, 동시 작업 및 데이터 구동 실행이 가능하고, 정적 또는 동적 어플리케이션 확장이 가능하다.
- [139]
- [140] 본 실시예에 따른 라디오 가상 머신(515)은 RVM 계층구조(Hierarchy)를 가진다. 즉, 라디오 가상 머신(515)은 Dos(470), APEs(460), ASF(440), CU(430) 및 eRVM(490)을 포함한다. RVM(515)은 eRVM(490)을 포함하여 RVM 계층구조를 구성한다.
- [141] 전술한 eRVM(490)은 APE(460)와 같이 기능할 수 있다. 도 6의 eRVM의 경우와

유사하게 컨트롤 유닛(CU)은 ASF(440)를 통해 APE(s)(460)에 대응하는 데이터 오브젝트(들)(DO(s))(470)에 연결될 수 있다. 물론, eRVM(490)은 RVM(515)의 복수의 APEs(460) 중 적어도 일부를 대체할 수 있다. 즉, eRVM(490)이 APEs(460) 중 일부만을 대체하는 경우, 데이터(Data)는 eRVM(490)뿐 아니라 APE(460)에도 연결될 수 있다.

[142] RVM(515) 내에 eRVM(490)을 가지는 전형적인 실시예로는, 구성코드가 표준 기능 블록들(SFBs)에 더하여 사용자 정의 기능 블록(UDFB)을 포함하는 경우이다. 구성코드 내 모든 표준 기능 블록은 라디오 라이브러리(514)를 이용하여 구현될 수 있고, 사용자 정의 기능 블록은 eRVM을 이용하여 독립적으로 구현될 수 있다.

[143] 그러므로, 본 실시예에서는 도 6에 도시한 바와 같이 UDFB(516)가 있는 경우에 그 UDFB(516)를 RVM 계층구조에 포함된 eRVM(490)을 이용하여 처리할 수 있다. UDFB(516)는 eRVM Configcodes를 통해 Basic Operations(410)에 다운로드될 수 있다.

[144]

[145] 도 6은 본 발명의 실시예에 따른 엘리먼트리 RVM(eRVM)을 도시한 개념도이다.

[146] 도 6을 참조하면, 본 실시예에 따른 엘리먼트리 라디오 가상 머신(Elementary RVM)(400)은, 데이터 오브젝트(DO, Data Objects)(470), APEs(460), 추상화 스위치 구조(ASF, Abstract Switch Fabric)(440), 및 컨트롤 유닛(CU, Control Unit)(430)을 포함한다. eRVM(400)은 다른 eRVM 또는 RVM(490)을 포함하지 않을 수 있다.

[147] 기본 작동은 특정 라디오 어플리케이션 패키지(RAP, Radio Application Package)(510)에서 정의된 UDFB(516) 및 표준 기능 블록(SFB, Standard Functional Block)(517-1~M)으로서 표준 라디오 라이브러리에서 수신한 연산자를 포함할 수 있다.

[148] 프로그램 메모리(420)는 eRVM(400)의 구성을 결정하는 구성코드를 포함할 수 있다. 컨트롤 유닛(430)은 프로그램 메모리(420)에 저장된 복호화 구성코드를 기초로 APEs(460), Dos(470), 추상화 스위치 구조(ASF, Abstract Switch Fabric)(440)를 위한 초기화 및 설정 지시를 생성할 수 있다.

[149] ASF(440)는 CU(430)의 신호에 따라 APEs(460) 및 데이터 오브젝트(470)들을 연결할 수 있다. 하나의 데이터 오브젝트(470)는 여러 개의 APEs(460)와 연결될 수 있다. 하나의 APE(460)는 다수의 데이터 오브젝트(470)들과 연결될 수 있다. 다른 eRVMs로부터의 데이터 오브젝트는 외부 데이터 포트(480)를 통해서 ASF(440)와 연결될 수 있다.

[150] 도 6에 도시된 바와 같이, eRVM(400)에서의 기본 동작은 라디오 라이브러리(514)또는 UDFB 세트(Set)(450)에서 다운로드된 동작으로 구성될 수 있다. 기본 동작은 오직 SFB(517-1~M) 또는 UDFB(516)를 포함할 수 있다. eRVM은

플랫폼에 독립적일 수 있다. 기본 동작이 라디오 라이브러리(514)에 대응되기 때문에, 각 APE(460)에 로드된 SFB의 입상(Granularity)은 서로 모두 다를 수 있다.

- [151]     컨트롤 유닛(430)은 Configcode(IR 혹은 Source Code 형태의 RA Code)에 따라 데이터 오브젝트(470), APE(460), ASF(440)의 행동양식을 결정해주는 기능을 수행한다. 컨트롤 유닛(430)에 의해 APE(460)가 데이터에 접속되는 시점이 결정되고, 각 APE(460)의 활성 타이밍이 결정될 수 있다. 이를 위해, 컨트롤 유닛(430)은 구성코드(Configcode) 내 메타데이터(513)로부터 타이밍 정보를 획득할 수 있다. 데이터 입력이 수행될 때, APE(460)는 자동으로 초기화될 수 있다. 각 APE(460)에 구비된 동작의 분할 정도는 구성코드를 실행하는 하드웨어 플랫폼에 따라 모두 다를 수 있다.
- [152]     전술한 데이터 오브젝트(DO, Data Object)(470)는 메모리를 추상화(Abstract)해주는 추상화 리소스(Abstract Resource)이다. 다시 말해, 송수신 데이터(Data)를 처리하는 과정에서 필요한 중간 데이터를 저장(Keeping)하기 위한 추상적 메모리 개념이라고 할 수 있다.
- [153]     추상화 처리 엘리먼트(APE, Abstract Processing Element)(460)는 단말 장치의 플랫폼(Platform)에 따라 서로 다른 하드웨어 리소스(Hardware Resource)를 추상화해주는 추상화 리소스이다.
- [154]     다시 말해, 미리 정해진 플랫폼으로 결정되는 플랫폼 특정한(Platform Specific) 하드웨어 리소스에서 라디오 라이브러리(514) 또는 UDFB 세트(Set)(450)에서 다운로드된 기본 작업(Basic Operations)(410) 즉, 표준 기능 블록(SFB)과 사용자 정의 기능 블록(UDFB)이 처리(Processing)할 수 있도록 하는 추상적 계산 엘리먼트(Computational Element) 개념이다. 추상화 스위치 구조(Abstract Switch Fabric)(440)는 데이터 오브젝트(470)와 APE(460)를 연결해주는 추상적 스위치 개념이다.
- [155]     기본 작업(Basic Operations)(410)은 해당 RVM에서 프로세싱될 수 있는 기능 블록(Functional Block(s))을 지칭할 수 있다. 다시 말해, 엘리먼트리 라디오 가상 머신(eRVM)(400)에서는 SFB(s)와 UDFB를 뜻하고(도 6 참조), 라디오 가상 머신(RVM)(515)에서도 SFB와 UDFB를 의미할 수 있다(도 5 참조).
- [156]     프로그램 메모리(Program Memory)(420)는 구성코드(Configcode) 즉, 중간 표현(IR, Intermediate Representation) 혹은 소스 코드(Source Code) 형태의 라디오 어플리케이션 코드(RA Code)를 저장할 수 있는 메모리를 의미할 수 있다.
- [157]     한편, eRVM 내 기본 작업은 레퍼런스 라디오 라이브러리(220) 및 UDFB 세트(Set)(450)에서 다운로드된 작업들로 구성될 수 있다. 기본 작업이 레퍼런스 라디오 라이브러리(220)에서만 SFBs를 받는 경우, 소정 RAP 내 UDFBs 각각은 대응 eRVM 또는 RVM 계층에 매핑될 수 있다.
- [158]     도 7은 데이터 오브젝트 및 데이터 오브젝트의 인터페이스를 도시한 개념도이다.

- [159] 도 7을 참조하면, eRVM의 데이터 경로는 아래의 블록으로 구성되어 있을 수 있다.
- [160] -데이터 오브젝트
- [161] -APEs
- [162] -ASF
- [163] 각 데이터 오브젝트(470)는 독특한 숫자를 가지고 있을 수 있다. 이러한 목적을 위해 데이터 오브젝트(470)는 데이터 오브젝트 1, 데이터 오브젝트 2, ..., 데이터 오브젝트 N과 같이 표현될 수 있다. N은 충분히 큰 숫자일 수 있다.
- [164] 도 7을 참조하면, 각 데이터 오브젝트(470)는 초기화 필드는 특정 초기화 절차에 따라 데이터 오브젝트를 초기화하는 구성(구현에 따라)과 세트 필드는 데이터 오브젝트 아이디(ID), 접속 시간, 크기 등 데이터 오브젝트 속성을 설정하는 명령과 같은 설정 구성으로 구성되어 있을 수 있다.
- [165] 데이터 오브젝트(470)는 APEs(460)와 ASF 인터페이스를 통해 통신하는 구성은 데이터 오브젝트(470)가 가득 차 있는지 또는 비어있는지를 지시하는 데이터 상태 신호 및 APEs(460)와 송수신하는 데이터를 쓰거나 읽기 위한 데이터 오브젝트(470)와 송수신하는 데이터 라인으로 구성될 수 있다.
- [166] 상태 인터페이스는 데이터 오브젝트(470)의 상태정보를 CU(430)에 제공할 수 있으며, 상태 인터페이스는 데이터 오브젝트(470)가 가득 차 있는지 또는 비어있는지를 설명하는 가득참부 또는 비어있음부 및 APE(460)가 데이터 오브젝트(470)와 동작할 때 실패의 원인을 설명하는 예외부로 구성될 수 있다.
- [167]
- [168] 도 8은 APE 및 APE의 인터페이스를 도시한 개념도이다.
- [169] 각 APEs(460)는 독특한 숫자를 가지고 있을 수 있다. 이러한 목적을 위해 APEs(460)는 APEs 1, APEs 2, ..., APEs N과 같이 표현될 수 있다. N은 충분히 큰 숫자일 수 있다.
- [170] 도 8을 참조하면, APEs(460)는 op 코드 작동을 기본 작동들로부터 가져오는 구성의 초기화 필드 및 데이터 포트 번호, 포트 타입, 실행 비용 및 시간 같은 APEs를 설정하는 명령과 같은 설정 구성의 세트 필드로 구성될 수 있다.
- [171] APE 포트는 APE(460)와 ASF(440)를 연결되어 있을 수 있다. 그리고 APE 포트는 데이터 인터페이스를 포함할 수 있다. 데이터 인터페이스는 데이터 오브젝트가 가득 차 있는지 또는 비어있는지를 지시하는 데이터 상태 신호, ASF(440)를 통해서 데이터를 쓰거나 읽기 위한 데이터 라인으로 구성될 수 있다.
- [172] 상태 인터페이스는 APEs(460)의 상태정보를 CU(430)에 제공할 수 있으며, 상태 인터페이스는 APE(460)의 상태를 활성화 또는 비활성화 같은 표시로 설명할 수 있는 활성화부 또는 비활성화부, APE(460)가 에러가 발생할 때 실패의 원인을 설명하는 예외부로 구성될 수 있다.
- [173] APE(460)는 입력 데이터 오브젝트(470)가 소비되고, 입력 데이터 오브젝트(470)가 처리될 때 활성화 될 수 있다. APE(460)는 APE(460)에 관련된

모든 데이터를 처리 한 후, 즉시 CU(430)에 대응 지시를 비활 성 상태로 변경할 수 있다.

- [174] ASF(440)는 도 9와 같이 APEs(460) 및 데이터 오브젝트(470)와 연결되어 있을 수 있다. 하나의 데이터 오브젝트(470)는 여러 개의 APEs(460)와 연결될 수 있다. 하나의 APEs(460) 또한 여러 개의 데이터 오브젝트(470)와 연결될 수 있다.

[175]

- [176] 도 9는 추상적 스위치 구조(Abstract Switch Fabric)를 도시한 개념도이다.

- [177] 도 9를 참조하면, ASF(440)는 데이터 오브젝트(470) 및 APEs(460)와 포트를 통하여 연결되어 있을 수 있으며, 포트는 ASF(440)와 데이터 오브젝트(470)를 인터페이스 라인들을 통하여 연결하는 내부 데이터 포트(441), ASF(440)와 다른 eRVMs 또는 RVMs로부터의 데이터 오브젝트를 연결하는 외부 데이터 포트(443), ASF(440)와 APEs(460)를 인터페이스 라인들을 통하여 연결하는 처리 포트(442)로 구성될 수 있다.

- [178] ASF(440)의 각 연결부는 APE 경계가 있는 포트와 함께 데이터 오브젝트의 경계 있는 포트를 ASF(440)와 연결 시킬 수 있다. 각 연결부는 데이터 오브젝트 포트, 즉 데이터 상태, 데이터와 같은 동일한 인터페이스 라인을 가질 수 있다.

- [179] 연결부는 그들이 포트를 대응하여 나타나는 경우, 포트들 사이의 인터페이스 값을 전달할 수 있다. CU(430)는 인터페이스에 의해 ASF(440)를 설정할 수 있다. 인터페이스는 데이터 오브젝트(470) 및 APEs(460)와 포트를 연결시키는 초기화부, 각 포트들 사이에 포트와 연결부를 생성하는 설정부로 구성될 수 있다.

[180]

- [181] 도 10은 RVM 계층 구조를 도시한 개념도이다.

- [182] 도 10은 APEs(460), 데이터 오브젝트(470), 기본 작업(410), 프로그램 메모리(420), 제어 장치(CU)(430), 추상적 스위치 구조(ASF)(440) 및 eRVMs, RVM(490)으로 구성된 RVM 계층 구조를 나타낼 수 있다. RVM은 다른 eRVM(s) 및 RVM(s)를 포함할 수 있다.

- [183] 도 10의 CU(430)는 ASF(440)를 통해 데이터 오브젝트(470)를 APE(s)(460)와 대응되도록 연결할 수 있다. 그러나 이 경우 데이터는 APE(460) 뿐만 아니라 eRVM 및 RVM(490)에 접속될 수도 있다. 기본적인 연산들은 레퍼런스 라디오 라이브러리(220)로부터만 SFBs를 받고, 소정 RAP(510) 내 각 UDFB(516)는 대응 eRVM/RVM에 매핑될 수 있다. 그리고 RVM 계층은 다운로드된 UDFBs에 대응하는 eRVM/RVM을 포함할 수 있다. 각 UDFB(516)는 SFBs의 조합으로서 표현될 수 있다. 어떤 UDFB(516)은 SFB의 조합으로 표현될 수 있다.

- [184] RVM(515)은 수직 및 수평으로 확장이 가능할 수 있다. 수직 스케일링에 관해서는, 각 eRVM은 RVM 계층 구조를 생성하기 위해 정확히 특정 데이터 플로우 차트, 즉, 특정 알고리즘을 포함하기 때문에, APE(460)는 다른 특정 데이터 플로우 차트를 실행하는 또 다른 eRVM 및 RVM(490)를 포함할 수 있다.

- [185] 수평 확장에 관해서는, 여러 라디오 가상 머신들(RVMs)은 동일 레벨에서 배치될 수 있다. 이러한 가로 배열된 RVMs은 독립이 요구될 수 있다. 독립이 요구된다는 것은 완전히 독립적인 처리가 병렬적으로 실행되어야 하는 것을 의미할 수 있다.
- [186]
- [187] 도 11은 eRVM의 수평 확장을 도시한 개념도이다.
- [188] 도 11을 참조하면, 유형이 없을 수 있다. 데이터 오브젝트(470)는 바이트 및 비트 안에서 약간의 크기가 있는 토큰들일 수 있다. 가득 찬 데이터 오브젝트(470)는 비트 필드로 취급될 수 있다. 데이터 오브젝트 구조는 초기화 APE에 의해 인식될 수 있다.
- [189] 런타임 데이터
- [190] 다음과 같은 런타임 데이터가 고려될 수 있다.
- [191] - 구성코드 카운터 등록: 양의 32 비트 정수를 등록할 수 있다.
- [192] - 상수 데이터 개체: 동적 비트 필드는 작업 실행 중에만 존재할 수 있다. 적 비트 필드는 어떤 외부 에이전트에 의해 초기화 될 수 있다.
- [193] - 중간 데이터 오브젝트: 동적 비트 필드는 작업 실행 중에만 존재할 수 있다. 동적 비트 필드는 APE 작동에 대응 결과 값으로서 생성된 중간 값에 의해 초기화 될 수 있다.
- [194] 산술은 고정되어 있지 않을 수 있다. 다만 산술은 기본 작동 세트로부터의 동작에 의해 정의될 수 있다.
- [195] 두 가지 유형의 예외가 있을 수 있다.
- [196] 데이터 오브젝트(470)에 의해 생성되는 경우, 표 1을 참조할 수 있다.
- [197] [표1]

| Value | Semantic                          |
|-------|-----------------------------------|
| 00    | No exception                      |
| 01    | Operation with size > DO size     |
| 10    | Conflicting writes                |
| 11    | Conflicting read-erase operations |

- [198] 표 1은 데이터 오브젝트(470)의 예외를 나타낸 표이다.
- [199] APEs 및 (e)RVMs에 의해 생성되는 경우, 표 2를 참조할 수 있다.

[200] [표2]

| Exception | Semantic                                                                                  |
|-----------|-------------------------------------------------------------------------------------------|
| 00        | No exception                                                                              |
| 01        | Change CC flow                                                                            |
| 10        | Arithmetic overflow/underflow                                                             |
| 11        | Incorrect operation (operation is carried out on data where the operation is not defined) |

[201] 표 2는 APEs(460) 및 (e)RVMs(490)의 예외를 나타낸 표이다. 표 2에서 CC는 Configcodes Count를 나타낸다.

[202]

[203] 제어, 동기화 및 실행

[204] 제어는 오직 데이터 기반일 수 있다.

[205] APE(460)에 의한 실행이 아래의 경우에 수행될 수 있다:

[206] - 모든 입력 데이터 객체는 구성되고, 단말 동작이 가득 차는 경우

[207] - 모든 입력 데이터 객체는 구성되고, 모든 입력 데이터 객체 중 일부는 비단말 동작이 가득 차는 경우

[208] - APE가 구성되는 경우

[209] - APE가 구성된 데이터 오브젝트와 연결되는 경우.

[210]

[211] APE(460)가 구성된 후, APE(460)가 작업을 수행하는 동안, APE(460)의 일부 출력 데이터 오브젝트(470)가 비어 있을 수도 있다. 실행 작업은 순간적인 것이 아니고, 일부 한정된 시간을 가지고 있을 수 있다. 모든 데이터 경로에 있는 작동은 동시 및 비동기일 수 있다.

[212] CU(430)는 직접 작업 실행을 관리하지 않을 수 있다.

[213] CU(430)는 아래의 동작을 수행할 수 있다:

[214] - 구성코드 실행 전에 모든 데이터 경로 요소를 구성하고,

[215] - 작업 실행 동안 모든 데이터 경로 요소로부터 상태 신호를 수신하고,

[216] - 각 구성코드의 끝을 감지하고,

[217] - 작업의 마지막 구성코드를 감지하고,

[218] - 예외의 경우 작업을 정지하는 동작.

[219]

[220] 메모리와 작업

[221] 메모리는 평평하고 무한할 수 있다.

[222] 그것의 구현은 본 문서의 범위 밖이다.

[223] 어떤 RVM도 메모리로 접근할 수 있다.

[224] 데이터 오브젝트(470)는 그들의 구성 동안 메모리에서 할당될 수 있다.

메모리는 다수의 병렬 읽기, 쓰기 작동을 허용할 수 있다. 충돌 작동은 예외를 강제할 수 있다. 데이터 오브젝트 메모리가 할당되는 것보다 더 많은 데이터를 기록하려는 시도는 예외를 강제할 수 있다. 특정 메모리 동작 동안 전체 데이터 오브젝트(470)는 소비될 수 있다.

[225]

[226] 실행 코드 배포를 위한 RVM

[227] 이 경우, RVM은 실행을 제어하는 구성요소만 포함해야 한다.

[228]

[229] 도 12는 RVM 입력 파일을 생성하기 위한 프로세싱 방법을 도시한 흐름도이다.

[230] 도 12를 참조하면, 라디오 가상 머신(515)을 위한 구성코드(Configcodes for RVM)에 관한 설명이 기재되어 있다. 이하에서는 RA(Radio Application) 코드 배포(Distribution)의 설계기간 동안 SWIR(Software Intermediate Representation)의 프론트-엔드 컴파일 결과로서 구성코드가 어떻게 생성되는지에 대해 설명한다.

[231] 구성코드는 벤더의 선택에 따라 바이너리 또는 XML(eXtensible Markup Language)로 생성된다. 라디오 가상 머신(515)은 실행 가능한 코드를 생성하기 위해 구성코드를 백-엔드 컴파일러를 위한 적절한 형태로 변환한다.

[232]

[233] 구성코드의 생성

[234] 이하에서는, 구성코드들이 어떻게 XML 또는 바이너리 형태로 생성되는지 설명할 것이며, 이후 XML 또는 바이너리 형태는 RVM 입력 파일로 지칭될 수 있다. 도 12는 대응하는 SWIR로부터 구성코드를 생성하는 프로세싱 단계들을 도시한다.

[235] 도 12에 도시된 구성코드 생성 프로세싱은 주어진 RA 코드의 알고리즘으로 이루어진 SWIR로부터 시작한다. SWIR은 고급-에벨 언어, 예를 들어, C, C++, 파스칼(Pascal) 등으로 생성된 주어진 RA 코드의 병렬화(parallelizing) 절차를 통해 생성된다. SWIR의 동작은 앞서 설명한 데이터-유도된 메커니즘에 기초한다. SWIR을 대응 구성코드로 변환하는 방법은 아래의 3가지 단계를 포함할 수 있다.

[236]

[237] 단계 1: SWIR 파일을 파싱하는 단계(Parsing SWIR file)

[238] 입력: SWIR 파일(S710)

[239] 출력: 터미널 객체들 및 터미널 객체들을 연결하는 링크들로 구성된 IR(Intermediate Representation) 객체들(S730)

[240] 동작: SWIR 파일을 파싱하고 섹션샌드(Sectionsand) 링크들로 구성된 IR 객체들을 생성한다(S720).

[241]

[242] 단계 2: IR 객체를 구성코드 객체로 매핑하는 단계

[243] 입력: IR 객체들의 리스트(S730)

- [244] 출력: 각각이, 터미널 오퍼레이터, APE(Abstract Processing Element), 데이터 오브젝트(Data Object) 및 스위치에 의해 결정된 하나의 구성, 그리고 구현된 데이터 플로우 차트를 특정하는 OpCode를 갖는 텍스트 파일을 나타내는 구성코드 객체들(S760)
- [245] 동작: 각 객체를 IR 데이터 포맷으로부터 구성코드 포맷으로 변환한다(S740). 이때, 태스크의 구성코드의 모든 구성들(Configurations)은 APE, 데이터 오브젝트 및 스위치를 위한 파라미터들에 의해 결정된다.
- [246] 각각의 구현된 기능(Function)의 명칭과 대응되는 OpCode 간에 매칭이 되도록 OpCode를 생성한다(S750).
- [247]
- [248] 단계 3: RVM을 위한 입력 파일을 생성하는 단계
- [249] 입력: 구성코드 객체(S760)
- [250] 출력: RVM 입력 파일(S780)
- [251] 동작: 구성코드 객체들로부터 RVM 입력 파일을 생성한다(S770).
- [252]
- [253] 도 13은 구성코드를 위한 바이너리 포맷을 도시한 개념도이다.
- [254] 도 13을 참조하면, RVM 입력 파일의 형태에는 여러 가지가 있을 수 있으며, 본 발명에서는 특히 XML 또는 바이너리 포맷을 염두에 둔다. XML 또는 바이너리 포맷으로 정의되는 구성코드의 예는 앞서 설명한 바 있다.
- [255]
- [256] 구성코드를 위한 바이너리 포맷(Binary Format)
- [257] 이하에서는 구성코드를 위한 바이너리 포맷이 설명된다. 각 태스크는 적어도 하나의 구성코드를 포함한다. 각각의 구성코드는 컨트롤 구성코드, 데이터 오브젝트 구성코드, APE 구성코드, ASF 구성코드, 및 부가적인 넥스트 구성코드 주소 오프셋(NCAO, Next Configcode Address Offset)을 포함한다.
- [258] NCAO 필드는 if NAF = 1인 경우 증가된다. 구성코드의 바이너리 형태는 도 13에 나타나 있으며, 아래의 요소들을 포함한다. 컨트롤 섹션은 태스크와 관련한 일반적인 정보를 제공하고 아래의 요소들을 포함한다.
- [259] LCF: 1 bit, LCF = 1 은 태스크 내에서 마지막 구성코드임을 나타냄
- [260] NAF: 1 bit, NAF = 1 은 NCAO 필드가 이 구성코드에 대해 증가함을 나타냄
- [261] Task\_ID: 8 bits, 는 이 태스크의 자동으로 생성된 식별자임
- [262] RPI 버전: 8 bits, 은 지원되는 RPI의 버전 넘버임
- [263] Reference\_ID: 8 bits, 는 레퍼런스 라디오 라이브러리의 SFB 식별자임
- [264] 구현 버전(Implementation Version): 8 bits, 은 구현된 태스크의 버전 넘버임
- [265] Developer\_ID: 16 bits, 는 현재 태스크를 만든 개발자의 식별자임
- [266] Creation\_Date: 16 bits, 태스크를 생성한 날짜임
- [267]
- [268] 데이터 오브젝트 섹션은 데이터 오브젝트와 관련한 일반적인 정보를 제공하고

아래의 요소들을 포함한다.

- [269] N\_DO: 8 bits, 해당 구성코드와 연관된 데이터 오브젝트 들의 개수임
- [270] N\_DO 특정 데이터 오브젝트 구성 필드들(N\_DO Particular DO Configuration Fields):
- [271] DO\_config 는 특정한 데이터 오브젝트 구성코드임
- [272] ASF\_config 는 데이터 오브젝트 필드가 없는 ASF 구성코드임
- [273]
- [274] APE 섹션은 APE 구성과 관련한 일반적인 정보를 제공하고 아래의 요소들을 포함한다.
- [275] N\_APE: 16 bits, 은 해당 구성코드의 연관된 APE의 개수들임
- [276] N\_APE 특정 APE 구성 필드들 (N\_APE Particular APE Configuration Fields)
- [277] APE 구성 필드는 특정 APE 구성코드임
- [278]
- [279] 도 14는 특정 데이터 오브젝트의 구성코드 포맷을 도시한 개념도이다.
- [280] 도 14를 참조하면, 특정 데이터 오브젝트 구성코드의 포맷은 두 부분을 포함하는데, 하나는 데이터 오브젝트 세트 코드(Set Code)의 고정 길이 필드이고, 다른 하나는 데이터 오브젝트 초기화 코드(Init Code)의 가변 길이 필드이다. 도 15에서 이러한 구성을 잘 보여준다.
- [281]
- [282] 도 15는 데이터 오브젝트 세트 코드의 포맷을 도시한 개념도이다.
- [283] 도 15를 참조하면, 도 15에 도시된 바와 같이 데이터 오브젝트 세트 코드는 아래의 요소들을 포함할 수 있다.
- [284] - 사이즈(Size): 데이터 오브젝트 사이즈를 바이트로 나타내는 양의 정수
- [285] - 액세스 타임(Access Time): 액세스 타임을 ns 단위로 나타내는 양의 정수
- [286] 데이터 오브젝트 속성들을 설정하는 세트 명령(Set Instruction) 즉, Set(YYYY, ZZZZ)이 생성된다. 여기서, "YYYY"는 바이트 단위의 사이즈이고, "ZZZZ" ns 단위의 액세스 타임이다.
- [287]
- [288] 도 16은 데이터 오브젝트 초기화 코드의 포맷을 도시한 개념도이다.
- [289] 도 16에 도시된 바와 같이 데이터 오브젝트 초기화 코드(Init Code)는 아래의 요소들을 포함할 수 있다.
- [290] 길이(Length): 1 byte, 필드의 가변 파트 길이를 바이트로 나타냄
- [291] 가변 길이 필드(Variable Length Field): 256 바이트까지 나타낼 수 있으며 즉시 값들(Immediate Values)을 가져옴
- [292]
- [293] 도 17은 특정 추상적 처리 추상화 처리 엘리먼트 구성코드(APE Configcode)의 포맷을 도시한 개념도이다.
- [294] 도 17을 참조하면, Init 필드는 (구현예에 의존하는) 특정 초기화 절차에 따라

데이터 오브젝트(470)를 초기화하며 Length 0, 즉, Init(XXXX), (여기서, XXXX는 비트 필드)이면 초기화 이후 데이터 오브젝트(470)를 full로 만든다. Init 필드는 Length = 0인 경우 비어 있을 수 있으며, 이 경우 데이터 오브젝트(470)는 비어 있다.

- [295] 도 17에 도시된 바와 같이 추상화 처리 엘리먼트 구성코드(APE Configcode)는 아래의 요소들을 포함할 수 있다.
- [296] APE\_ID: 2 bytes, APE의 개수임
- [297] Op Code: 20 bits로 기본 작업(Basic Operations)(410)으로부터의 오퍼레이션 코드임
- [298] T: 1 bit 플래그로, 동적 동작인 T = 1일 때 APE는 해당 동작의 완료 직후 비활성화(Inactive) 상태이며, 정적 동작인 T = 0일 때 APE는 해당 동작이 완료된 이후라도 활성화(Active) 상태임
- [299] NN: 3 bits, 포트의 개수
- [300] 비용: 2 bytes, 실행 비용 값
- [301] 시간: 2 bytes, 시간 제한 값(Time Constraint Value)
- [302] 포트 액세스 타입: 포트 당 2 bits, 액세스 타입 {r, re, w, rew }을 설명함
- [303]
- [304] 도 18은 ASF Configcode의 포맷을 도시한 개념도이다.
- [305] 도 18을 참조하면, APE(460)는 Init 필드, Set 필드 및 APE 포트 식별자 (APE\_Port\_ID) 필드를 갖는 명령어(Instruction) APE\_config(APE\_Port\_ID, Init, Set)에 의해 구성된다.
- [306] APE\_Port\_ID 필드는 식별자 APE\_Port\_ID(XXXX.YY, PORT\_TYPE)를 가지며, 여기서, XXXX = APE\_ID, YY는 APE 포트의 개수, PORT\_TYPE = {r, re, w, rew}이다.
- [307] Init 필드는 기본 작업(Basic Operations)(410) 세트 및 포트 구성으로부터 오퍼레이션 식별자를 가져온다. 구성 절차 동안, APE(460)는 Init(op)를 이용하여 베이직 오퍼레이션으로부터 Op 오퍼레이션으로 바운드된다. 오퍼레이션의 입력 및 출력 파라미터들은 대응하는 액세스 타입의 APE 포트들로 바운드된다.
- [308] Set필드는 Set(NN, PP, XXXX, YYYY, ZZZZ)를 이용해 APE 속성들을 설정하는데, 여기서, NN은 포트의 개수(십진수), PP는 필드 정의 포트 타입, XXXX는 실행 비용(십진수)을 나타내며, YYYY = 0이면 비용이 없음을, ZZZZ는 ms 단위의 시간(십진수)을, ZZZZ = 0이면 시간 제약이 없음을 나타낸다.
- [309] APE\_configuration 절차가 성공적으로 완료되면 APE(460)는,
- [310] 활성화 상태로 전환하며, 이를 CU(430)로 알려주고,
- [311] DS(Data Status) 및 Data 라인들에서 값들을 예측한다.
- [312] APE(460) 예외 코드들이 [표 3]에 나타나 있다.

[313] [표3]

| 예외사항(Exception) | 시맨틱(Semantic)                |
|-----------------|------------------------------|
| 00              | 예외 없음(No exception)          |
| 01              | CC(Configcodes Count) 플로우 변환 |
| 10              | 산술적                          |

[314] 표 3은 APE(460)의 예외사항들을 나타낸 표이다.

[315]

[316] 도 18은 ASF\_config 코드의 포맷을 도시한다.

[317] 앞서 설명한 바와 같이 포트들은 ASF(440)를 데이터 오브젝트(470)들 또는 APE(460)들과 연결한다. ASF(440)의 커넥터들이 데이터 오브젝트(470)로 바운드되는 포트들을 APE(460)로 바운드되는 포트들과 연결시킨다.

[318] 각 커넥터는 포트들과 동일한 인터페이스 라인들(DA, Data)을 가진다. 커넥터들은 해당 포트들에 나타날 때 포트들 간에 인터페이스 값들을 전달한다.

[319] 도 18에 도시된 ASF\_config 코드는 아래의 요소들을 포함한다.

[320] DO: 1 byte 필드로 데이터 오브젝트 넘버를 나타냄

[321] N: 6 bits로, 데이터 오브젝트 연결된 APE들의 개수를 나타냄

[322] APE\_K.P: 12 bits로 K는 0부터 N-1까지 변화함

[323] APE\_K: 8-bit로 APE 넘버를 나타냄

[324] P: 4-bit로 포트 넘버를 나타냄.

[325]

[326] ASF(440)는 Set 필드 및 Init 필드를 갖는 ASF\_config (Set, Init) 명령어에 의해 구성된다.

[327] Set 필드는 Set(DO, N)을 이용해 ASF 속성들을 설정하는데, 여기서 데이터 오브젝트(DO)(470)는 데이터 오브젝트(470)의 넘버이고, N은 데이터 오브젝트에 연결된 APE(460)들의 개수이다.

[328] Init 필드는 Init(APE\_K.P)를 이용해 ASF(440)를 초기화하는데, APE\_K(K는 0부터 N-1까지 변화)는 APE 번호를, P는 포트 번호를 나타낸다.

[329] 구성 후에는 커넥터들이 즉시 인터페이스 값들을 데이터 오브젝트(470)들 및 APE(460)들로 전달한다.

[330]

[331] 도 19는 구성코드들로 구성되는 XML 엘리먼트들을 도시하는 블록도이다.

[332] 도 19를 참조하면, 구성코드를 위한 XML 스키마가 설명된다. 각 태스크(Task)는 적어도 하나의 구성코드를 포함한다. XML 엘리먼트들은 아래와 같다.

- [333]     태스크(Task); 구성 코드(Configcode); DOconfig; DO; ASF;  
 [334]     Virtual; APEconfig; APE; Complex; FOR; SWITCH; IF; WHILE; Port  
 [335]  
 [336]     [표 4] 내지 [표 17] 는 도 19에 도시된 각 요소의 속성들을 나타낸다.  
 [337]     엘리먼트/Task  
 [338]     모 엘리먼트(Parent Element): 문서 루트(Document Root)  
 [339]     자(Children):  
 [340]     <Configcode>  
 [341]     <Task> 는 xml의 루트 요소이다.  
 [342]     [표4]

| 속성             | 설명                              |
|----------------|---------------------------------|
| TaskID         | 태스크 식별자                         |
| RPIVersion     | RPI의 버전                         |
| DeveloperID    | 개발자 식별자                         |
| CreationDate   | 태스크 생성 날짜                       |
| ReferenceID    | 레퍼런스 라디오 라이브러리의 SFB 식별자         |
| TaskImpVersion | 태스크 구현 버전                       |
| num            | 태스크 내포함된 구성들(configurations)의 수 |

- [343]     표 4는 Task의 속성들을 나타낸 표이다.  
 [344]     엘리먼트 /Task/Configcode  
 [345]     -모(Parent): <Task>  
 [346]     -자(Children):  
 [347]     <DOconfig>  
 [348]     <APEconfig>  
 [349]     <Configcode>는 태스크 내 하나의 구성을 위한 태그이다.  
 [350]     [표5]

| 속성   | 설명                                                 |
|------|----------------------------------------------------|
| LCF  | 최후 구성 플래그(Last Config Flag)로 태스크 내에서 마지막 구성코드를 식별함 |
| ID   | 자동 생성된 식별자                                         |
| NDO  | 구성 내 데이터 객체의 수                                     |
| NAPE | 구성 내 APE의 수                                        |

- [351]     표 5는 Configcode의 속성들을 나타낸 표이다.  
 [352]     엘리먼트 /Task/Configcode/DOconfig  
 [353]     -모(Parent): <Configcode>  
 [354]     -자(Children):

- [355] <DO>  
 [356] <Virtual/>  
 [357] <DOconfig> 데이터 객체 구성을 위한 태그이다.  
 [358]  
 [359] 요소 /Task/Configcode/DOconfig/DO  
 [360] -모(Parent): <DOconfig>  
 [361] -자(Children):  
 [362] <ASF>  
 [363] <DO> 는 하나의 데이터 객체의 구성을 위한 태그이다.  
 [364] [표6]

| 속성     | 설명              |
|--------|-----------------|
| ID     | 자동 생성된 식별자      |
| Size   | 데이터 객체의 크기      |
| AccT   | 데이터 객체의 액세스 타임  |
| Length | 객체 내 초기 데이터의 길이 |
| Data   | 초기 데이터          |

- [365] 표 6은 데이터 오브젝트(470)의 속성들을 나타낸 표이다.  
 [366] 요소 /Task/Configcode/DOconfig/DO/ASF  
 [367] -모(Parent): <DO>  
 [368] -자(Children):  
 [369] <Port/>  
 [370] <ASF>는 DO를 위한 ASF(Abstract Switch Fabric)의 구성을 위한 태그이다.  
 [371] [표7]

| 속성     | 설명                 |
|--------|--------------------|
| APENum | 데이터 객체에 연결된 APE의 수 |

- [372] 표 7은 ASF(440)의 속성들을 나타낸 표이다.  
 [373] 엘리먼트 /Task/Configcode/DOconfig/ASF/Port  
 [374] -모(Parent): <ASF>  
 [375] -자(Children): None  
 [376] <Port>는 스위치 포트를 위한 태그이다.  
 [377]

[378] [표8]

| 속성     | 설명        |
|--------|-----------|
| APEid  | APE 식별자.  |
| PortId | Port 식별자. |

[379] 표 8은 Port의 속성들을 나타낸 표이다.

[380] 엘리먼트 /Task/Configcode/DOconfig/Virtual

[381] -모(Parent): &lt;DOconfig&gt;

[382] -자(Children): None

[383] &lt;Virtual&gt; 는 가상 데이터 객체를 위한 태그이다.

[384]

[385] [표9]

| 속성         | 설명                                |
|------------|-----------------------------------|
| ID         | 자동 생성된 식별자                        |
| Type       | 데이터 객체의 사이즈                       |
| PortRef    | VDO가 연결된 메인 제어 오퍼레이터의 포트          |
| InPortRef  | VDO를 위한 입력 포트가 될, 메인 제어 오퍼레이터의 포트 |
| OutPortRef | VDO를 위한 출력 포트가 될, 메인 제어 오퍼레이터의 포트 |

[386] 표 9는 Virtual의 속성들을 나타낸 표이다.

[387] 엘리먼트 /Task/Configcode/APEconfig

[388] -모(Parent): &lt;Configcode&gt;

[389] -자(Children):

[390] &lt;APE&gt;

[391] &lt;Complex&gt;

[392] &lt;Switch&gt;

[393] &lt;IF&gt;

[394] &lt;FOR&gt;

[395] &lt;WHILE&gt;

[396] &lt;APEconfig &gt; 추상 물리적 엘리먼트 구성들을 위한 태그이다.

[397]

[398] 엘리먼트 /Task/Configcode/APEconfig/APE

[399] -모(Parent): &lt;APEconfig&gt;

[400] -자(Children):

[401] &lt;Port&gt;

[402] &lt;APE&gt;는 하나의 기능적 APE의 구성을 위한 태그이다.

[403]

[404] [표10]

| 속성      | 설명                            |
|---------|-------------------------------|
| ID      | 자동 생성된 식별자                    |
| OpCode  | 구현된 기능(function)의 OpCode      |
| T       | 동적 오퍼레이션에 대한 플래그              |
| Cost    | 실행 비용 값(Execution cost value) |
| Time    | 시간 제한 값(Time constrain value) |
| PortNum | APE의 스위치 포트들의 개수              |

[405] 표 10은 APE(460)의 속성들을 나타낸 표이다.

[406] 엘리먼트 /Task/Configcode/APEconfig/Complex

[407] -모(Parent): &lt;APEconfig&gt;

[408] -자(Children):

[409] &lt;Port&gt;

[410] &lt;Complex&gt; 콤플렉스 APE의 구성을 위한 태그이다.

[411]

[412] [표11]

| 속성      | 설명                            |
|---------|-------------------------------|
| ID      | 자동 생성된 식별자.                   |
| Body    | 바디 구성의 ID                     |
| T       | 동적 오퍼레이션에 대한 플래그              |
| Cost    | 실행 비용 값(Execution cost value) |
| Time    | 시간 제한 값(Time constrain value) |
| PortNum | APE의 스위치 포트들의 개수              |

[413] 표 11은 Complex의 속성들을 나타낸 표이다.

[414] 엘리먼트 /Task/Configcode/APEconfig/FOR

[415] -모(Parent): &lt;APEconfig&gt;

[416] -자(Children):

[417] &lt;Port&gt;

[418] &lt;FOR&gt;은 FOR-cycle의 구성을 위한 태그이다.

[419]

[420] [표12]

| 속성      | 설명                            |
|---------|-------------------------------|
| ID      | 자동 생성된 식별자                    |
| Body    | 싸이클 바디 구성의 ID                 |
| T       | 동적 오퍼레이션에 대한 플래그              |
| Cost    | 실행 비용 값(Execution cost value) |
| Time    | 시간 제한 값(Time constrain value) |
| PortNum | APE의 스위치 포트들의 개수              |

[421] 표 12는 FOR의 속성들을 나타낸 표이다.

[422] 엘리먼트 /Task/Configcode/APEconfig/WHILE

[423] -모(Parent): &lt;APEconfig&gt;

[424] -자(Children):

[425] &lt;Port&gt;

[426] &lt;WHILE&gt;은 While-cycle의 구성을 위한 태그이다.

[427]

[428] [표13]

| 속성      | 설명                            |
|---------|-------------------------------|
| ID      | 자동 생성된 식별자.                   |
| Body    | 싸이클 바디 구성의 ID                 |
| T       | 동적 오퍼레이션에 대한 플래그              |
| Cost    | 실행 비용 값(Execution cost value) |
| Time    | 시간 제한 값(Time constrain value) |
| PortNum | APE의 스위치 포트들의 개수              |

[429] 표 13은 WHILE의 속성들을 나타낸 표이다.

[430] 엘리먼트 /Task/Configcode/APEconfig/IF

[431] -모(Parent): &lt;APEconfig&gt;

[432] -자(Children):

[433] &lt;Port&gt;

[434] &lt;IF&gt; 는 IF 오퍼레이터의 구성을 위한 태그이다.

[435]

[436] [표 14]

| 속성      | 설명                            |
|---------|-------------------------------|
| ID      | 자동 생성된 식별자                    |
| Then    | body-then 구성의 ID              |
| Else    | body-else 구성의 ID              |
| T       | 동적 오퍼레이션에 대한 플래그              |
| Cost    | 실행 비용 값(Execution cost value) |
| Time    | 시간 제한 값(Time constrain value) |
| PortNum | APE의 스위치 포트들의 개수              |

[437] 표 14는 IF의 속성들을 나타낸 표이다.

[438] 엘리먼트 /Task/Configcode/APEconfig/Switch

[439] -모(Parent): &lt;APEconfig&gt;

[440] -자(Children):

[441] &lt;Port&gt;

[442] &lt;Case&gt;

[443] &lt;Switch&gt; 는 Switch-Case 오퍼레이터의 구성을 위한 태그이다.

[444]

[445] [표 15]

| 속성      | 설명                            |
|---------|-------------------------------|
| ID      | 자동 생성된 식별자                    |
| T       | 동적 오퍼레이션에 대한 플래그              |
| Cost    | 실행 비용 값(Execution cost value) |
| Time    | 시간 제한 값(Time constrain value) |
| PortNum | APE의 스위치 포트들의 개수              |

[446] 표 15는 Switch의 속성들을 나타낸 표이다.

[447] 엘리먼트/task/configcode/APEconfig/APEComplexFor/WHILE/IF/Switch/Port

[448] -모(Parent): &lt;APEComplexFor/WHILE/IF/Switch&gt;

[449] -자(Children): None

[450] &lt;Port&gt;는 ASF 포트를 위한 태그이다.

[451]

[452] [표16]

| 속성      | 설명      |
|---------|---------|
| ID      | 포트 식별자. |
| AccType | 액세스 타입  |

[453] 표 16은 Port의 속성들을 나타낸 표이다.

[454] 엘리먼트 /Task/Configcode/APEconfig/Switch/Case

[455] -모(Parent): &lt;Switch&gt;

[456] -자(Children): None

[457] &lt;Port&gt; 는 Switch 오퍼레이터 케이스 부분을 위한 태그이다.

[458]

[459] [표17]

| 속성        | 설명                                     |
|-----------|----------------------------------------|
| Body      | 케이스-바디 구성(case-body configuration)의 ID |
| Condition | 케이스 조건                                 |

[460] 표 17은 port의 속성들을 나타낸 표이다.

[461] 도 20은 도 2의 단말 장치의 라디오 컴퓨터의 소프트웨어 아키텍처를 설명하기 위한 블록도이다.

[462] 도 20을 참조하면, 라디오 컴퓨터는 모바일 장치에 통신 능력(Communication Capabilities)을 제공하며, 그 소프트웨어 아키텍처는 다음의 구성요소들을 포함하여 구성될 수 있다.

[463] - 라디오 OS

[464] - 라디오 컨트롤 프레임워크의 라디오 프로세서 실행 부분

[465] - 쉐도우 라디오 플랫폼(Shadow Radio Platform)이 라디오 가상 머신(RVM, Radio Virtual Machine)일 경우, 라디오 가상 머신의 구현(Implementation)

[466] - 쉐도우 라디오 플랫폼이 라디오 가상 머신일 경우 라디오 라이브러리(Radio Lib)의 네이티브 구현(Native Implementation)

[467] - 라디오 어플리케이션(RA)들의 구성코드들(Configuration Code; Configcodes), 구성코드는 타겟 라디오 플랫폼의 실행 코드 혹은 플랫폼 독립적인 중간표현(IR: Intermediate Representation) 형태로 제공될 수 있다.

[468] 쉐도우 라디오 플랫폼이 라디오 가상 머신일 경우, Configcode는 라디오 가상 머신(515)에 의해서 해석(Interpreted)되고, 쉐도우 라디오 플랫폼이 타겟 라디오 플랫폼일 경우, Configcode는 해당 타겟 라디오 플랫폼에서 직접 실행될 수 있는 실행 가능 코드에 해당된다.

[469] RCF(126)와 그 인터페이스들인 MURI(Multi Radio Interface) 및 URAI(Unified Radio Application Interfaces)는 상기 설명한 바와 같다.

- [470] 웨도우 라디오 플랫폼은 라디오 가상 머신(515) 또는 타겟 라디오 플랫폼일 수 있다.
- [471] 만약 웨도우 라디오 플랫폼이 타겟 라디오 플랫폼과 동일하다면, 프론트-엔드 컴파일러(Front-End Compiler)가 타겟 플랫폼을 위한 실행가능 코드를 생성할 것이며 Configcodes는 그러한 특정 플랫폼을 위한 실행가능 코드와 동등하다.
- [472] 라디오 가상 머신(515)은 Configcodes를 실행할 수 있는 추상화 머신이며, 어떠한 하드웨어에 대해서도 독립적이다. Configcode는 특정 라디오 가상 머신을 통해서 타겟 플랫폼상에서 실행된다. 따라서, 라디오 가상머신(515)은 Configcodes를 실행 코드들로 컴파일하기 위한 JIT(Just-in-time) 또는 AOT(Ahead-of-time) 방법을 제공하는 백-엔드(Back-end) 컴파일러를 포함한다.
- [473] 라디오 라이브러리(514)는 연산 기반(Computational Basis)을 표현하는 기능 블록들로 구성된다. 라디오 어플리케이션은 이러한 상호 연결된 기능 블록들의 집합으로서 표현될 수 있다. 라디오 라이브러리(514)의 기능 블록들은 정규화된 언어(Normative Language)로서 표현된다. 라디오 라이브러리(514)의 네이티브 구현은 타겟 플랫폼을 위한 라이브러리의 기능 블록들의 실행 가능 코드들을 제공한다. 라디오 라이브러리(514)는 확장 가능하다.
- [474]
- [475] 통합 라디오 어플리케이션의 동작 구조
- [476] 통합 라디오 어플리케이션(125)의 동작 구조는 두 가지 다른 경우로 표현될 수 있다. 첫 번째는 라디오 어플리케이션 Configcodes가 타겟 플랫폼상에서 실행 가능한 코드인 경우(도 7을 통하여 예시)이고, 두 번째는 라디오 어플리케이션 Configcodes가 주어진 모바일 디바이스에서 백-엔드 컴파일되는 중간 표현(IR: Intermediate Representation) 코드인 경우(도 8을 통하여 예시)이다.
- [477] 도 21은 도 2의 단말 장치의 통합 라디오 어플리케이션(125)의 동작 구조의 일례를 설명하기 위한 계층 구조도이다.
- [478] 도 21을 참조하면, 주어진 라디오 어플리케이션을 수행하기 위해 필요한 라디오 라이브러리(514)와 사용자 정의 기능 블록들(UDFB, User Defined Function Block)(516)이 이미 라디오 어플리케이션의 실행 가능 Configcode들에 포함될 수 있다.
- [479] 한편, 주어진 라디오 어플리케이션(들)을 수행하기 위해 필요한 사용자 정의 기능 블록(516)들은 라디오 어플리케이션의 Configcode들에 포함되어 있고 라디오 가상 머신(515)에 의해서 백-엔드 컴파일되어야 한다(도 8 참조). 이 경우에는, 라디오 어플리케이션 Configcodes에 라디오 라이브러리(514)가 포함될 수 없기 때문에 라디오 라이브러리(514)의 네이티브 구현이 주어진 모바일 디바이스 내에 별도로 준비되어야만 한다. 일반적으로 라디오 라이브러리(514)의 네이티브 구현은 프로그래머블 하드웨어(123-1)에 의해서 제공되는데 라디오 라이브러리(514)는 프로그래머블 하드웨어(123-1) 상에서 구현되는 표준 기능 블록(SFB, Standard Function Block)(517-1~M)들을 포함하기

때문이다.

- [480] 하드웨어 가속기(123-2)들을 사용하지 않고 구현될 수 있는 라디오 라이브러리(네이티브 구현)은 표준 기능 블록들의 속도 향상과 가속기들과 프로그램 코드를 결합하여 다른 표준 기능 블록들을 생성하기 위해서 필요하다.
- [481] 라디오 어플리케이션 Configcodes가 실행 가능 코드이거나 중간 표현 코드이든지 어느 경우에나, 표준 기능 블록들은 라디오 하드웨어 추상화 계층(HAL)(124)을 통하여 하드웨어 가속기(123-2)들에 의해서 지원된다.
- [482] 이는 하드웨어 로직에 의해서 구현된 표준 기능 블록들이 주어진 라디오 어플리케이션 코드들에 의해서 호출될 때마다, 라디오 어플리케이션 Configcodes가 실행 가능 코드이든지 중간 표현 코드이든 지간에, 라디오 HAL(124)을 거쳐서 대응되는 하드웨어 가속기(123-2)들 상에서 직접 구현되어야 함을 의미한다.
- [483] 후술되는 바와 같이, 라디오 HAL(124)은 또한 사용자 정의 기능 블록 라이브러리들을 위해서 준비된 인터페이스들을 위한 하드웨어 추상화를 포함할 수 있다.
- [484] 표준 기능 블록(517-1~M)들은 많은 라디오 어플리케이션들에서 공통적으로 사용되는 기능 블록들 예컨대, FFT(Fast Fourier Transform) 및/또는 주어진 라디오 플랫폼에서 특수 목적 가속기를 사용하여 매우 효율적으로 구현되어야만 하는 어떠한 기능 블록들(예컨대, 터보 코더)이 될 수 있다.
- [485]
- [486] 도 22는 도 2의 단말 장치의 라디오 플랫폼의 기능 블록 라이브러리 구현들을 설명하기 위한 개념도이다.
- [487] 도 22를 참조하면, 통합 라디오 어플리케이션의 동작 구조는 다음의 구성 요소들을 포함한다.
- [488] - 통합 라디오 어플리케이션(125)은 표준 기능 블록들(517-1, 517-3, 517-M)과 사용자 정의 기능 블록들(517-2, 516)을 주어진 라디오 어플리케이션 패키지(RAP, Radio Application Package)(510)의 메타데이터(513)의 내용에 대응하여 포함한다.
- [489] - 라디오 라이브러리(네이티브 구현)(514)는, 하드웨어 로직을 이용하여 구현되는 표준 기능 블록들이 라디오 HAL(124)에 의해서 지원되는 반면, 프로그래머블 하드웨어(123-1) 상에서 실행되는 표준 기능 블록들의 Configcodes를 포함한다.
- [490] - 라디오 가상 머신(515)은 모바일 장치의 라디오 특성에 영향을 미치는 소프트웨어를 위한 제어된 실행 환경이다. 이를 이용하면, 재구성 소프트웨어, 즉 라디오 어플리케이션들이 라디오 가상 머신(515)에 로드되는 상황을 가정할 수 있다.
- [491] - '사용자 정의 기능 블록 집합'(450)은 일반적으로 라디오 어플리케이션 프로바이더(Radio Application Provider)에 의해서 제공되며, 주어진 라디오

어플리케이션 패키지(510)에서 사용되는 모든 사용자 정의 기능 블록들을 포함한다. 사용자 정의 기능 블록(516)은 라디오 어플리케이션 패키지(510)에 메타데이터(Metadata) (513) 및 라디오 컨트롤러 코드(512)와 함께 포함된다. 일반적으로 사용자 정의 기능 블록(516)은 표준 기능 블록의 수정 및/또는 확장 버전이기 때문에, 사용자 정의 기능 블록(516)은 표준 기능 블록 라이브러리에 대해서 의존성을 가질 수 있다.

- [492] - 라디오 HAL(124)은 라디오 플랫폼(123)을 추상화한다. 라디오 HAL(124)은 하드웨어 가속기(Hardware Accelerator)(123-2)를 이용하여 구현되는 표준 기능 블록들이 대응 하드웨어 가속기(123-2)들 상에서 직접 실행될 수 있도록 하는 점에 주의하여야 한다.
- [493] - 라디오 플랫폼 드라이버(122)는 라디오 OS(121)가 라디오 플랫폼(123)을 인식하도록 한다.
- [494] - 라디오 플랫폼(123)은 일반적으로 프로그래머블 하드웨어와 하드웨어 가속기들을 모두 포함하여 구성될 수 있다.
- [495] 한편, 전술한 '사용자 정의 기능 블록 집합(UDFB Set)'(450)은 주어진 라디오 어플리케이션에 의해서 사용되는 모든 사용자 정의 기능 블록들을 포함한다. 어떠한 표준 기능 블록이라도 그것을 적절한 표준 기능 블록들 즉, 대체될 표준 기능 블록들의 수정 및/또는 확장된 버전으로 대체하는 것에 의해서 수정 및/또는 확장 가능하다는 점이 중요하다.
- [496] 따라서, 어떤 사용자 기능 블록들은 표준 기능 블록의 확장을 위해서 좋은 후보가 될 수 있으며, 그것들이 추후 표준 기능 블록으로서 추가될 수 있음을 의미한다. 이러한 경우에, 사용자 기능 블록들이 표준 기능 블록으로서 추가된 후에는 정규 표준 기능 블록들로서 규정될 것이다.
- [497] 또한, '사용자 정의 기능 블록 집합(UDFB Set)'(450)은 라디오 플랫폼 벤더 대신 라디오 어플리케이션의 제공자(즉, 3rd Party)에 의해서 제공될 수 있기 때문에, 라디오 컨트롤 프레임워크(126)가 모든 사용자 기능 블록의 이벤트 및/또는 커맨드들의 기본적인 제어를 수행할 수 있도록 하기 위해서, 'Start', 'Stop', 'Pause', 'Get Port' 및 'Initialize'와 같은 제어 인터페이스들의 표준 집합이 대응되는 사용자 기능 블록들을 위해서 규정되어야만 할 수 있다.
- [498] 사용자 정의 기능 블록(516)들에 대한 제어 인터페이스의 표준 집합 규정은 별도로 주어질 수 있다. 도 8과 도 9에 나타낸 라디오 플랫폼(123)은 프로그래머블 하드웨어(123-1)와 하드웨어 가속기(123-2)들을 각 기능 블록들을 구현하기 위해서 포함할 수 있다.
- [499]
- [500] 도 23은 도 2의 단말 장치에서 라디오 플랫폼의 기능 블록 라이브러리 구현들을 설명하기 위한 개념도이다.
- [501] 본 실시예에서는 라디오 프로세서와 다양한 종류의 주변 디바이스들로 구성된 주어진 라디오 플랫폼(123)으로 이루어진 라디오 컴퓨터(120)의 기능 블록

구현들이 예시된다.

- [502] 도 23을 참조하면, 라디오 프로세서(610)상에 구현되는, 예컨대 하드웨어 가속기(123-2)를 이용하여 구현되는 표준 기능 블록들(예컨대, FFT, Turbo Decoder, MIMO Decoder 등)은 대응되는 하드웨어 가속기(123-2) 상에서 높은 성능과 저전력 소모를 위해서 직접적으로 실행될 수 있다. 이러한 표준 기능 블록들은 가속기 상에서의 실행을 위해서 라디오 HAL(Hardware Abstraction Layer)(124)에 의해서 지원된다.
- [503] 이는 가속기 상에서 실행되는 각 표준 기능 블록들이 라디오 어플리케이션에서 호출될 때 대응되는 가속기(들)에서 라디오 HAL(124)을 거쳐 직접적으로 실행되는 것을 의미한다. 유사하게, 예를 들어 비트 뒤집기(Bit-reverse), 곱셈(Multiply) 및 축적(Accumulation) 등과 같은 코어 프로세서상에서 실행되는 각 표준 기능 블록들은 라디오 어플리케이션에서 호출될 때, 주어진 라디오 프로세서(610)(예컨대, ARM with Neon) 상에서 수행된다.
- [504] 결과적으로, 라디오 프로세서(610) 상에서 필요한 실행 코드들은 다음 두 부분으로 구성된다. 하나의 부분은 프로그래머블 하드웨어(123-1) 상에서 수행되는 표준 기능 블록들의 실행코드이며, 다른 부분은 가속기들 상에서 수행되는 표준 기능 블록들을 위한 라디오 HAL 코드들이다.
- [505] 이는 다음과 같이 요약될 수 있다. {C: 표준 기능 블록들의 구현을 위해 필요한 라디오 프로세서 상의 실행 코드들}={A: 프로그래머블 하드웨어 상에서 수행되는 표준 기능 블록들을 위한 실행 코드들}+{B: 가속기들 상에서 수행되는 표준 기능 블록들을 위한 라디오 HAL 코드들}. 즉,  $C=A+B$ 이며 A와 B의 분할은 각 벤더들에 의해서 결정될 수 있다.
- [506] 이는 또한 다음을 암시한다. {표준 기능 블록들}={프로그래머블 하드웨어 상에서 구현되는 표준 기능 블록들}과 {하드웨어 가속기들 상에서 구현되는 표준 기능 블록들}의 합집합이며, {프로그래머블 하드웨어 상에서 구현되는 표준 기능 블록들}과 {하드웨어 가속기(Hardware Accelerator) 상에서 구현되는 표준 기능 블록들}의 교집합은 공집합이 된다.
- [507] 한편, 앞서 언급된 바와 같이, 사용자 정의 기능 블록(516)은 표준 인터페이스를 이용하여 기술되어야 한다. 도 9에 도시한 바와 같이, 사용자 정의 기능 블록(516)의 표준 인터페이스는 프로그래머블 하드웨어(123-1) 상에 구현되는 표준기능블록들 및 하드웨어 가속기(Hardware Accelerator)(123-2) 상에서 구현되는 표준 기능 블록들 중 하나 또는 양자와 연관되어야 함에 주목하여야 한다.
- [508] 표준 인터페이스들을 두 그룹 즉, 프로그래머블 하드웨어(123-1) 상에서 수행되는 표준 기능 블록들(517-1~M)에 대응되는 그룹과, 하드웨어 가속기(123-2) 상에서 수행되는 표준 기능 블록들에 대응되는 그룹으로 분류한 이유는 각 카테고리가 나름의 장점과 단점을 가지기 때문이다. 후자는 일반적으로 하드웨어 로직으로 구현되므로, 전력 소모, 속도가 필요한 동작 및

아마도 비용 효율성에서 유리하다.

- [509] 반면, 전자는 일반적으로 마이크로프로세서 상에서 수행되기 때문에 유연성(Flexibility)에서 유리하다. 성능의 측면에서 프로그래머블 디바이스들이 하드웨어 디바이스들에 비하여 경쟁력을 가질 때까지는 초기 단계에서는 하드웨어 가속기(123-2)들이 상대적으로 더 광범위하게 사용될 것이 예상된다.
- [510] 장기적인 관점에서 반도체 기술이 점점 더 진화할수록, 프로그래머블 하드웨어 의존적인 표준 기능 블록들이 주변기기 의존적인(Peripheral Dependent) 표준 기능 블록들에 비하여 점진적으로 더 우세해질 것이며, 명령어 셋 아키텍처(ISA: Instruction Set Architecture) 레벨의 가속 기능으로 구현될 수 있다.
- [511] 본 명세서에서 예시된 복수의 표준 기능 블록들은 데이터의 분할의 정도(Granularity)를 나타내는 것으로서 설명의 목적으로 예시된 것일 뿐이다.

[512]

[513] 라디오 어플리케이션 패키지의 구성

- [514] 이하에서는, 본 발명에 따른 단말 장치로의 라디오 어플리케이션의 배포를 위한 라디오 어플리케이션 패키지(RAP: Radio Application Package)(510)의 구성을 설명한다.

[515]

- [516] 도 24는 도 2의 단말 장치에서 다운로드할 수 있는 라디오 어플리케이션 패키지의 구성예를 설명하기 위한 블록도이다.

- [517] 도 24를 참조하면, 본 실시예에 따른 라디오 어플리케이션 패키지(510)의 적어도 하나의 라디오 어플리케이션은 기능 블록들과 라디오 컨트롤러로 구성될 수 있다. 즉, 라디오 어플리케이션 패키지(510)는 사용자 정의 기능 블록 코드(511), 라디오 라이브러리와 라디오 컨트롤러 코드(512)를 포함할 수 있다. 그리고, 라디오 어플리케이션의 배포를 위한 라디오 어플리케이션 패키지(510)는 사용자 정의 기능 블록 코드(511) 및 라디오 컨트롤러 코드(512)에 더하여 파이프라인 구성 메타 데이터(Pipeline Configuration Meta-data)(513)를 추가로 포함할 수 있다.

- [518] 라디오 컨트롤러 코드(512)는 상술된 소프트웨어 아키텍처 환경에 따라서 라디오 프로세서(610)와 어플리케이션 프로세서(110) 중 어느 프로세서의 실행 코드의 형태로 라디오 어플리케이션 패키지(510)에 포함될 지가 결정된다. 즉, 라디오 컨트롤 프레임워크(126)가 어플리케이션 프로세서(110) 실행 부분과 라디오 프로세서(610) 실행 부분으로 분리된 경우는 라디오 컨트롤러 코드(512)는 어플리케이션 프로세서(110)에서 실행되는 코드로 구성될 수 있으며, 라디오 컨트롤 프레임워크(126)가 라디오 프로세서(610)에서만 실행되는 경우에는 라디오 컨트롤러 코드(512)는 라디오 프로세서(610)에 실행되는 코드로 구성될 수 있다.

- [519] 한편, 사용자 정의 기능 블록 코드(511)는, 앞서 언급된 바와 같이, 어느 경우이든 라디오 프로세서(610)에서 실행되는 코드로서 라디오

프로세서(610)에서 실행 가능한 실행 코드, 소스코드, 중간표현 형태의 코드로서 라디오 어플리케이션 패키지(510)에 포함될 수 있다.

[520] 파이프라인(Pipeline)이란 라디오 어플리케이션의 송신 또는 수신 기능을 구현하기 위한 라디오 어플리케이션의 라디오 컨트롤러, 사용자 정의 기능 블록들(516) 및 표준 기능 블록들(517-1~M)의 조합과 그 연결관계를 의미하는 것으로서, 파이프라인 구성 메타 데이터(513)에 의해서 정의될 수 있다.

[521] 또한, 표준 기능 블록 코드가 라디오 프로세서(610)의 코어에서 실행 가능한 실행 코드 형태로 구성된다면 라디오 어플리케이션 패키지(530)는 실행코드 형태 즉 라디오 프로세서(610)의 코어가 실행 가능한 실행 코드의 라디오 라이브러리(534)를 추가적으로 포함할 수 있다.

[522] 라디오 어플리케이션 패키지(510)는 공급자에 의한 표준 베이스밴드 API 헤더(520)를 포함하도록 생성되어 서버(530)으로 업로드되고, 서버(530)로부터 사용자의 단말 장치(10)의 어플리케이션 프로세서 계층의 운영체제(OS)(112)에 다운로드되고, 사용자 정의 기능 블록 코드(512)와 라디오 라이브러리(514)는 파이프라인 구성 메타 데이터(513)를 참조하여 어플리케이션 프로세서(110)에서 라디오 프로세서(610)로 로딩되는 과정을 거쳐 라디오 프로세서 계층의 라디오 운영체제(121) 즉 실시간 운영체제에 로딩될 수 있다.

[523]

[524] 도 25는 도 2의 단말 장치의 아키텍처를 개략적으로 나타낸 블록도이다.

[525] 도 25를 참조하면, 본 실시예에 따른 단말 장치는 베이스밴드 프로세싱을 위한 라디오 가상 머신(RVM, Radio Virtual Machine)(515)과 ASIC(Application Specific Integrated Circuit) 타입의 베이스밴드 구현 집적회로(Hard-wired)(842), 및 RF 프론트엔드 또는 RF 트랜시버 체인(RF Front-end / RF Transceiver Chain)(810)을 포함한다. 라디오 가상 머신 소프트웨어 성분(RVM Software Component)(841)은 RF 및 RVM 프로텍션 클래스들을 선택할 수 있다. 그리고 베이스밴드 프로세싱(Base-Band Processing)(840) 장치는 어플리케이션 프로세서(Application Processor) 혹은 MAC(Layer 2) 및 그 상위 계층 프로세싱(Higher Layer Processing) 장치(850)와 연결될 수 있다.

[526] 또한, 단말 장치는 RF 프론트-엔드 또는 RF 트랜시버 체인(810)과 베이스밴드 프로세싱 장치(840) 사이에 아날로그-디지털 변환기(820) 및 디지털-아날로그 변환기(830)를 구비할 수 있다. 여기서, 아날로그-디지털 변환기(820)와 디지털-아날로그 변환기(830)는 RRFI에서 특정되지 않는 DigRF 등의 기존의 인터페이스 표준에 의해 구현될 수 있다.

[527] RF 프론트-엔드 또는 RF 트랜시버 체인(810)과 RVM 소프트웨어 성분(841)이나 베이스밴드 구현 집적회로(842)는 본 실시예의 재구성가능 무선주파수 인터페이스(RRFI)에 의해 연결될 수 있다.

[528]

[529] 라디오 라이브러리(514)는 표준 기준 블록(SFB)의 전체 세트를 포함한다(ETSI

- EN 303 095). 라디오 라이브러리(514)는 다음과 같이 2가지로 분류된다.
- [530] 레퍼런스 라디오 라이브러리(220)는 각 표준 기능 블록(SFB)의 정규화된 설명을 제공한다. 네이티브 라디오 라이브러리(230)는 타겟 플랫폼 하드웨어를 나타내는 각 표준 기능 블록(SFB)의 플랫폼-특정된 설명을 제공한다.
- [531] 라디오 라이브러리(514)에 포함된 계산 규범 설명이나 프론트-엔드 또는 백-엔드 컴파일, 각각 사용되는 라디오 라이브러리에 따라 표준 기능 블록(SFB)들의 네이티브 구현 측면에서 표현된다. 레퍼런스 라디오 라이브러리(220) 및 네이티브 라이브러리(230)는 아래에 자세히 설명한다.
- [532] 레퍼런스 라디오 라이브러리(220)는 모든 표준 기능 블록(SFB)들의 정규화된 설명을 포함한다. 레퍼런스 라디오 라이브러리(220)는 상술한 라디오 가상 머신(RVM)(515)의 기본작업(410)에서 다운로드 하는 각 연산자에 대한 정보를 포함한다. 각 연산자는 ETSI EN 303 095에 정의된 표준 기능 블록(SFB)에 대응된다.
- [533] 레퍼런스 라디오 라이브러리(220)는 각각 모든 표준 기능 블록(SFB)의 정규화된 설명을 포함한다.
- [534] 각 표준 기능 블록(SFB)은 라디오 컴퓨팅에 사용되는 연산자에 대응된다. 예를 들면 FFT, IFFT, 컨볼루션(Convolution) 등. 각 표준 기능 블록(SFB)의 정규화된 설명은 제 3파티의 공급자가 참조하도록 예를 들어, C, C++, C# 등 높은 수준의 언어에서 주어질 수 있다.
- [535] 정규화된 설명은 RA코드를 만드는 동안 제 3자가 사용하도록 주어진 표준 기능 블록(SFB)의 각 기능에 필요한 비용 등 기타정보(Other Information)를 포함한다. 표준 기능 블록(SFB)들의 전형적인 후보들은 ETSI TS 103 146-4의 Annex B에 주어졌다.
- [536] 표준 기능 블록(SFB)의 대부분 기본 레벨은 엘리먼트리 표준 기능 블록(elementary SFB, eSFB)로서 표현된다. eSFB는 터미널 라이브러리 요소이며, 이러한 엘리먼트리 표준 기능 블록(eSFB)은 더 미세한 수준의 엘리먼트로 나누어질 수 없다. 표 18은 엘리먼트리 표준 기능 블록(eSFB)들로 정의된 연산자들을 보여준다.

[537] [표 18]

| Name           | Input | Output | Description                                                                                                 |
|----------------|-------|--------|-------------------------------------------------------------------------------------------------------------|
| NOP            | 1     | 1      | No operation, coping input to output.                                                                       |
| Copy operator  | 1     | 2      | Coping 1 input to 2 outputs.                                                                                |
| Filter         | 2     | 1      | 1 predicate input and 1 data input, if predicate input is true than coping data input to output.            |
| Multiplexer    | 2     | 1      | Any input triggers coping this input to output.                                                             |
| Demultiplexer  | 2     | 2      | If predicate is true than copy input to the 1 <sup>st</sup> output else to the 2 <sup>nd</sup> .            |
| Return         | 1     | 1      | Stop configuration execution with corresponding code sent to output.                                        |
| Add            | 2     | 1      | Returns the sum of two inputs.                                                                              |
| Subtract       | 2     | 1      | Subtract the second input from the first input.                                                             |
| Multiply       | 2     | 1      | Returns the multiple of two inputs.                                                                         |
| Divide         | 2     | 1      | Divides one input value by the other input.                                                                 |
| Logical AND    | 2     | 1      | Returns true if both inputs can be converted to true; otherwise, returns false.                             |
| Logical OR     | 2     | 1      | Returns true if either input can be converted to true; if both can be converted to false, returns false.    |
| Logical NOT    | 1     | 1      | Returns false if input can be converted to true; otherwise, returns true.                                   |
| Bitwise AND    | 2     | 1      | Returns a one in each bit position for which the corresponding bits of both inputs are ones.                |
| Bitwise OR     | 2     | 1      | Returns a one in each bit position for which the corresponding bits of either or both inputs are ones.      |
| Bitwise XOR    | 2     | 1      | Returns a one in each bit position for which the corresponding bits of either but not both inputs are ones. |
| Bitwise NOT    | 1     | 1      | Inverts the bits of its input.                                                                              |
| Left shift     | 1     | 1      | Shifts a in binary representation b (< 32) bits to the left, shifting in zeroes from the right.             |
| Right shift    | 1     | 1      | Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off.                  |
| Preincrement   | 1     | 1      | Increment input by 1, then use the new value of input in the current operation.                             |
| Postincrements | 1     | 1      | Use the current value of input in the current operation, then increment input by 1.                         |
| Predecrement   | 1     | 1      | Decrement input by 1, then use the new value of input in the current operation.                             |
| Postdecrements | 1     | 1      | Use the current value of input in the current operation, then decrement input by 1.                         |

[538] 표 18은 eSFB 리스트를 나타낸 표이다.

[539] 도 26은 본 실시예에 따른 표준 기능 블록들을 결정하는 방법에 대한 흐름도이다.

[540] 본 실시예에 따른 표준 기능 블록들의 결정 방법은, 새로운 SFB 후보가 제안되면(S910), 소위 라디오 라이브러리 위원회(RLA, Radio Library Authority)가

후보 SFB를 새로운 SFB로서 등록할 것인지를 결정할 수 있다(S930). RLA에서의 주요 요인(Key Factor)은 소정의 라디오 라이브러리에서 사용되는 후보 기능 블록(FB)이 얼마나 중요한지 및/또는 얼마나 자주 사용되는지가 고려될 수 있다.

[541] 도 26에 나타내 바와 같이, SFB 후보로 고려되기 위해, 후보 FB는 레퍼런스 코드의 형태로 표현될 수 있다. 레퍼런스 코드는 eSFB들의 적정 조합으로 구성될 수 있다. 재사용 가능성(Reusability)은 새로운 SFB를 결정하는 주요 요인(Key Factor)이므로, 도 26에 도시한 FB 재사용 가능성 인덱스(FBRI)와 같은 평가 지수가 후보 FB의 하드웨어 및 소프트웨어 관련 인자들 모두의 관점에서 RLA에 의해 설정될 수 있다(S920). 하드웨어 차원(HD, Hardware Dimension) 및 소프트웨어 차원(SD, Software Dimension)은 후보 FB의 하드웨어 관련 인자와 소프트웨어 관련 인자(Factor)와 각각 관련되어 있다. HD에 대한 평가에서, 후보 FB는 성능 향상 정도, 하드웨어 구현 난이도, 시장성(Time to Market) 등에 대하여 평가될 수 있다. SD에 대한 평가에서, 후보 FB는 소프트웨어 정확성과 완벽성, IPR(Intellectual Property Rights) 라이선스 필요성, FB 알고리즘의 복잡성 및/또는 재사용성 등의 항목에서 평가될 수 있다.

[542] 전술한 표준 기능 블록들(SFBs)의 기본적인 후보들을 나타내면 아래의 표 19 내지 표 23과 같다.

[543] [표19]

| Classification     | Data type      | Range                                           | Byte  |
|--------------------|----------------|-------------------------------------------------|-------|
| integer            | char           | -128~127                                        | 1(8)  |
|                    | unsigned char  | 0~255                                           | 1(8)  |
|                    | short          | -32768~32767                                    | 2(16) |
|                    | int            | -2147483648~2147483647                          | 4(32) |
|                    | long           | -2147483648~2147483647                          | 4(32) |
|                    | unsigned short | 0~65535                                         | 2(16) |
|                    | unsigned int   | 0~4294967295                                    | 4(32) |
|                    | unsigned long  | 0~4294967295                                    | 4(32) |
| real-floating type | float          | $8.4 \times 10^{-37} \sim 3.4 \times 10^{39}$   | 4(32) |
|                    | double         | $2.2 \times 10^{-309} \sim 1.8 \times 10^{309}$ | 8(64) |

[544] [표 20]

| Name                          |         | Syntax | Number of operands | Description       |
|-------------------------------|---------|--------|--------------------|-------------------|
| Basic assignment              |         | a = b  | 2                  | Allocate b to a   |
| Addition                      |         | a + b  | 2                  | Add a to b        |
| Subtraction                   |         | a - b  | 2                  | Subtract b from a |
| Multiplication                |         | a * b  | 2                  | Multiply a with b |
| Division                      |         | a / b  | 2                  | Divide b from a   |
| Modulo<br>(integer remainder) |         | a % b  | 2                  | Modulo b from a   |
| Increment                     | Prefix  | ++a    | 1                  |                   |
|                               | Postfix | a++    | 1                  |                   |
| Decrement                     | Prefix  | --a    | 1                  |                   |
|                               | Postfix | a--    | 1                  |                   |

[545] [표 21]

| Name                     | Syntax | Number of operands | Description |
|--------------------------|--------|--------------------|-------------|
| Equal to                 | a == b | 2                  |             |
| Not equal to             | a != b | 2                  |             |
| Greater than             | a > b  | 2                  |             |
| Less than                | a < b  | 2                  |             |
| Greater than or equal to | a >= b | 2                  |             |
| Less than or equal to    | a <= b | 2                  |             |

[546] [표 22]

| Name                      | Syntax | Number of operands | Description |
|---------------------------|--------|--------------------|-------------|
| Logical negation<br>(NOT) | !a     | 1                  |             |
| Logical AND               | a && b | 2                  |             |
| Logical OR                | a    b | 2                  |             |

[547] [표 23]

| Name                | Syntax | Number of operands | Description |
|---------------------|--------|--------------------|-------------|
| Bitwise NOT         | ~a     | 1                  |             |
| Bitwise AND         | a & b  | 2                  |             |
| Bitwise OR          | a   b  | 2                  |             |
| Bitwise XOR         | a ^ b  | 2                  |             |
| Bitwise left shift  | a << b | 2                  |             |
| Bitwise right shift | a >> b | 2                  |             |

- [548]     표 19 내지 표 23에 나타낸 바와 같이, SFBs의 기본 후보들은 정수(Integer), 실부동소수점 타입(Real-Floating Type) 등이나, 산술 연산에 관련된 후보들, 비교 연산에 관련된 후보들, 로직 연산에 관련된 후보들, 비트 연산(Bitwise Operations)에 관련된 후보들을 포함할 수 있다.
- [549]
- [550]     도 27은 본 실시예에 따른 방법에 있어서 네이티브 라디오 라이브러리를 생성하는 과정을 나타낸 흐름도이다.
- [551]     본 실시예에 따른 네이티브 라디오 라이브러리(Native Radio Library)(230)는 레퍼런스 라디오 라이브러리(220)의 각 표준 기능 블록(SFB)이 타겟 하드웨어 플랫폼에서 구현되는 방법에 대한 정보를 포함한다. 정보는 전용 하드웨어 가속기(123-2) 및 타겟 하드웨어 플랫폼에서 제공하는 프로그래밍 가능한 장치를 사용하여 레퍼런스 라디오 라이브러리(220)에 명시적으로 각 표준 기능 블록(SFB) 구현을 지정한다. 네이티브 라디오 라이브러리(230)는 구성코드(Configcodes)의 백-엔드 컴파일을 위해 사용될 수 있다.
- [552]     도 27에 나타낸 바와 같이, 네이티브 라디오 라이브러리(230)는 레퍼런스 라디오 라이브러리(220)에 기초하여 수행될 수 있다. 레퍼런스 라디오 라이브러리(220)는 개방성, 표준 및 공중 이용가능성 및 하드웨어(H/W) 라이브러리(250)일 수 있다. 하드웨어 라이브러리(250)는 타겟 하드웨어 플랫폼을 탑재될 수 있다. 네이티브 라디오 라이브러리(230)의 엘리먼트들은 레퍼런스 라디오 라이브러리(220) 엘리먼트들을 하드웨어 라이브러리(250) 엘리먼트들에 맵핑하여 생성될 수 있다(S1000).
- [553]     표 24는 XML과 바이너리(Binary) 간의 맵핑을 보여준다.

[554] [표 24]

| XML schema     | Binary format          | Description                                                                          |
|----------------|------------------------|--------------------------------------------------------------------------------------|
| TaskID         | Task_ID                | Task identifier.                                                                     |
| RPIVersion     | RPI version            | Version of RPI.                                                                      |
| DeveloperID    | Developer_ID           | Developer identifier.                                                                |
| CreationDate   | Creation_Date          | Date of task creation.                                                               |
| ReferenceID    | Reference_ID           | SFB identifier of reference Radio Library.                                           |
| TaskImpVersion | Implementation version | Task implementation version.                                                         |
| Num            | NAF                    | Contain number of configurations in the task.                                        |
| LCF            | LCF                    | Last Config Flag. Identify last configcode in the task                               |
| NDO            | N_DO                   | Number of data objects in the configuration.                                         |
| NAPE           | N_APE                  | Number of APE in the configuration.                                                  |
| ID(DO)         | DO_ID                  | Automatically generated identifier                                                   |
| Size           | SIZE                   | Size of data object.                                                                 |
| AccT           | ACCESS TIME            | Accesses time of data object.                                                        |
| Length         | LENGTH                 | Length of initial data in object.                                                    |
| Data           | VARIABLE LENGTH FIELD  | Initial data.                                                                        |
| APENum         | N                      | Number of APE connected to data object.                                              |
| APEid          | APE_ID                 | APE identifier.                                                                      |
| PortId         | APE_Port_ID            | Port identifier.                                                                     |
| Type           | Not applicable         | Size of data object.                                                                 |
| PortRef        | Not applicable         | Port of the main control operator, to which Virtual DO is connected.                 |
| InPortRef      | Not applicable         | Port of the main control operator, which will be the input port for the Virtual DO.  |
| OutPortRef     | Not applicable         | Port of the main control operator, which will be the output port for the Virtual DO. |
| ID(APE)        | APE_ID                 | Automatically generated identifier.                                                  |
| OpCode         | op code                | OpCode of implemented function.                                                      |
| T              | T                      | Flag for dynamic operations.                                                         |
| Cost           | cost                   | Execution cost value.                                                                |
| Time           | time                   | Time constrain value.                                                                |
| PortNum        | NN                     | Number of Switch ports for APE                                                       |
| AccType        | port access type       | Accesses type.                                                                       |

[555] 표 24를 참조하면, 바이너리 포맷에 대한 XML 스키마(Schema)를 맵핑할 수 있다.

[556] 전술한 실시예들에 있어서, 재구성 가능한 모바일 장치의 정보 모델 및 프로토콜에 있어서, RPI(Radio Programming Interface)의 맵핑 관계를 나타내면 다음의 표 25와 같다.

[557] [표 25]

| Entity/Component/Unit       | System Requirements [i.6] | Comments                                                                                                                                                                                                                                    |
|-----------------------------|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Radio Programming Interface | R-FUNC-MDR-04             | The reconfigurable MD shall provide a suitable interface which conveys structural and behavioural information of RAs for the reconfigurable MD reconfiguration. The requirement is described in clause 6.4.4 of ETSI EN 302 969 [i.6].      |
| Radio Virtual Machine       | R-FUNC-MDR-13             | Radio Application(s), SFB(s) or UDFB(s) shall be executed on a suitably configured Radio Virtual Machine, including the application of a suitable protection class. The requirement is described in clause 6.4.13 of ETSI EN 302 969 [i.6]. |
|                             | R-FUNC-MDR-14             | A Radio Virtual Machine may consist of several smaller Radio Virtual Machines. The requirement is described in clause 6.4.14 of ETSI EN 302 969 [i.6].                                                                                      |
|                             | R-FUNC-MDR-15             | A Radio Application shall select a suitable Radio Virtual Machine Protection Class. The requirement is described in clause 6.4.15 of ETSI EN 302 969 [i.6].                                                                                 |
| Radio Library               | R-FUNC-FB-06              | A library extension shall be supported. The requirement is described in clause 6.3.6 of ETSI EN 302 969 [i.6].                                                                                                                              |

[558] 표 25에서 재구성 가능한 모바일 장치의 시스템 요구사항은 ETSI EN 302 969 문서에 따를 수 있다.

[559] 전술한 실시예들의 라디오 어플리케이션 실행 방법은 이를 구현한 프로그램이나 소프트웨어 형태로 컴퓨터 판독 가능 매체(기록매체)에 저장되도록 구현될 수 있다. 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하는 형태로 구현될 수 있다. 컴퓨터 판독 가능 매체에 기록되는 프로그램은 본 발명을 위해 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수 있다.

[560] 또한, 컴퓨터 판독 가능 매체는 롬(Rom), 램(Ram), 캐시 메모리, 플래시 메모리(Flash Memory) 등과 같이 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치를 포함할 수 있다. 프로그램 명령은 컴파일러(Compiler)에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터(Interpreter) 등을 사용해서 컴퓨터에 의해 실행될 수 있는 고급 언어 코드를 포함할 수 있다. 하드웨어 장치는 전술한 실시예의 라디오 어플리케이션 실행 방법을 수행하기 위해 적어도 하나의 소프트웨어 모듈로 작동하도록 구성될 수 있으며, 그 역도 마찬가지일 수 있다.

[561] 상기에서는 본 발명의 바람직한 실시예를 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

## 청구범위

- [청구항 1] 라디오 컴퓨터 또는 라디오 프로세서 및 어플리케이션 프로세서를 구비하는 하드웨어 독립적인 라디오 어플리케이션을 실행하는 단말 장치로서,  
메모리를 추상화해주는 추상화 리소스인 적어도 하나의 데이터 오브젝트(DO, Data Object);  
상기 단말 장치의 플랫폼에 따라 서로 다른 하드웨어 리소스를 추상화해주는 적어도 하나의 추상화 처리 엘리먼트(APE, Abstract Processing Element);  
구성코드에 따라서, 복수의 데이터 오브젝트(DO, Data Objects)들 및 복수의 APE들의 행동 양식을 결정하는 컨트롤 유닛(CU, Control Unit)을 포함한 라디오 가상 머신(RVM, Radio Virtual Machine)을 포함하는,  
라디오 어플리케이션을 실행하는 단말 장치.
- [청구항 2] 청구항 1에 있어서,  
상기 적어도 하나의 DO 및 상기 적어도 하나의 APE를 연결하는 추상화 스위치 구조(ASF, Abstract Switch Fabric)를 추가로 포함하고,  
상기 컨트롤 유닛은 상기 구성코드에 따라서, 상기 ASF의 행동 양식을 추가로 결정하는, 라디오 어플리케이션을 실행하는 단말 장치.
- [청구항 3] 청구항 1에 있어서,  
상기 RVM은 상기 RVM에 포함된 적어도 하나의 APE가 적어도 하나의 엘리먼트리 라디오 가상 머신(eRVM, elementary Radio Virtual Machine)으로 대체된 계층적 구조를 가지는, 라디오 어플리케이션을 실행하는 단말 장치.
- [청구항 4] 청구항 1에 있어서,  
상기 RVM은,  
상기 구성코드를 저장하는 프로그램 메모리(Program Memory)를 추가로 포함하는, 라디오 어플리케이션을 실행하는 단말 장치.
- [청구항 5] 청구항 2에 있어서,  
상기 DO는,  
상기 DO를 초기화하는 초기화 필드;  
상기 DO의 속성을 설정하는 세트 필드;  
상기 DO와 상기 복수의 APE들과 통신하는 ASF 인터페이스; 및  
상기 DO의 상태정보를 CU에 제공하는 상태 인터페이스를 포함하는,  
라디오 어플리케이션을 실행하는 단말 장치.
- [청구항 6] 청구항 5에 있어서,  
상기 ASF 인터페이스는,  
상기 DO가 가득 차 있는지 또는 비어있는지를 지시하는 데이터 상태

신호; 및

상기 DO가 상기 복수의 APE들과 데이터를 송수신하는 데이터 라인을 포함하고,

상기 상태 인터페이스는,

상기 DO가 가득 차 있는지 또는 비어 있는지를 설명하는

가득참·비어있음부; 및

복수의 APE들이 상기 DO와 동작할 때 실패의 원인을 설명하는 예외부를 포함하는, 라디오 어플리케이션을 실행하는 단말 장치.

[청구항 7]

청구항 2에 있어서,

상기 APE는,

작동 코드를 기본 작업(Basic Operations)으로부터 가져오는 초기화 필드;

상기 APE를 설정하는 명령을 설정하는 세트 필드;

상기 ASF와 연결하기 위한 데이터 인터페이스를 포함하는 APE포트; 및

상기 APE의 상태정보를 상기 컨트롤 유닛에 제공하는 상태 인터페이스를 포함하는, 라디오 어플리케이션을 실행하는 단말 장치.

[청구항 8]

청구항 7에 있어서,

상기 데이터 인터페이스는,

DO가 가득 차 있는지 또는 비어 있는지를 지시하는 데이터 상태 신호; 및

상기 ASF를 통해서 데이터를 쓰거나 읽기 위한 데이터 라인을 포함하고,

상기 상태 인터페이스는,

상기 APE의 상태를 활성화 또는 비활성화 같은 표시로 설명하는

활성화·비활성화부; 및

상기 APE가 에러가 발생할 때 실패의 원인을 설명하는 예외부를

포함하는, 라디오 어플리케이션을 실행하는 단말 장치.

[청구항 9]

청구항 2에 있어서,

상기 ASF는,

상기 ASF와 상기 복수의 DO들을 상기 복수의 DO들의 ASF 인터페이스 라인을 통하여 연결하는 내부 데이터 포트;

상기 ASF와 상기 복수의 APE들을 상기 복수의 APE들의 데이터 인터페이스 라인들을 통하여 연결하는 처리 포트;

상기 컨트롤 유닛이 상기 ASF를 설정하는 인터페이스;

복수의 포트들 사이에 상기 ASF를 설정하는 인터페이스의 값을 전달하는 연결부; 및

상기 RVM이 수평으로 확장한다면, 상기 ASF와 다른 eRVM 또는

RVM으로부터의 복수의 DO들을 연결하는 외부 데이터 포트를 포함하는, 라디오 어플리케이션을 실행하는 단말 장치.

[청구항 10]

청구항 9에 있어서,

상기 ASF를 설정하는 상기 인터페이스는,

상기 복수의 DO들 및 상기 복수의 APE들과 포트를 연결하는 초기화부;  
 및  
 상기 복수의 포트들 사이에 연결부를 생성하는 설정부를 포함하는,  
 라디오 어플리케이션을 실행하는 단말 장치.

[청구항 11] 청구항 1에 있어서,  
 상기 RVM은 Just-in-time(JIT) 또는 Ahead-of-time(AOT) 백-엔드(Back-end) 컴파일러를 포함하는, 라디오 어플리케이션을 실행하는 단말 장치.

[청구항 12] 청구항 1에 있어서,  
 상기 구성코드는,  
 상기 라디오 어플리케이션의 사용자 정의 기능 블록들을 구성하는  
 사용자 정의 기능 블록 코드 및 상기 라디오 어플리케이션이 이용하는  
 표준 기능 블록들을 구성하는 라디오 라이브러리를 라디오 플랫폼에서  
 실행 가능한 실행 코드를 포함하는, 라디오 어플리케이션을 실행하는  
 단말 장치.

[청구항 13] 청구항 12에 있어서,  
 상기 표준 기능 블록들 중에서 하드웨어 로직으로 구현되는 표준 기능  
 블록을 실행시키는 라디오 프로세서 계층상의 라디오 HAL(Hardware  
 Abstraction Layer)을 포함하는, 라디오 어플리케이션을 실행하는 단말  
 장치.

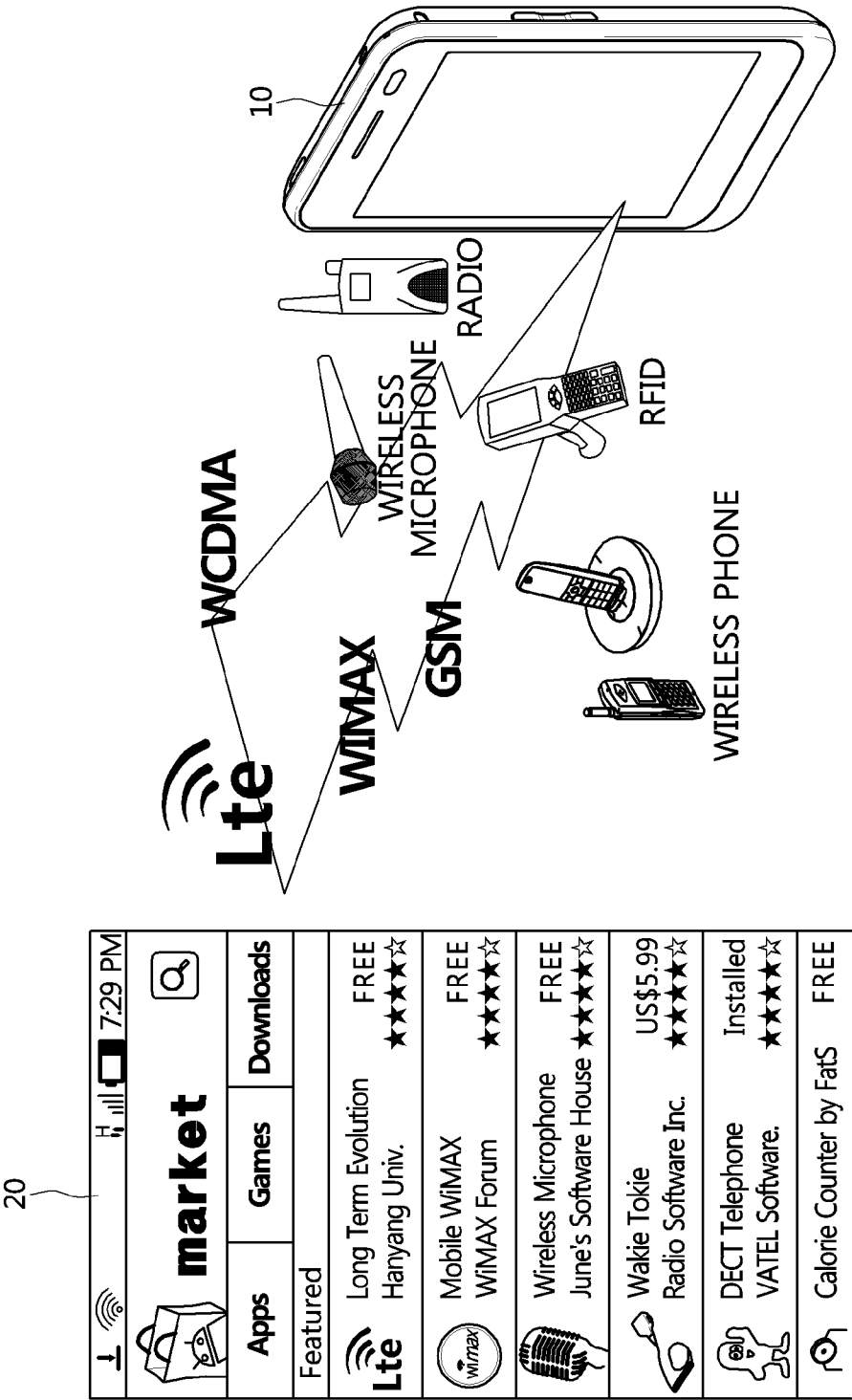
[청구항 14] 청구항 1에 있어서,  
 상기 구성코드는,  
 서버로부터 다운로드한 라디오 어플리케이션 패키지(RAP, Radio  
 Application Package)에 포함되어 있는 코드를 포함하는, 라디오  
 어플리케이션을 실행하는 단말 장치.

[청구항 15] 청구항 1에 있어서,  
 상기 라디오 어플리케이션에게 동작 환경을 제공하는 라디오 컨트롤  
 프레임워크(RCF, Radio Control Framework)를 포함하는, 라디오  
 어플리케이션을 실행하는 단말 장치.

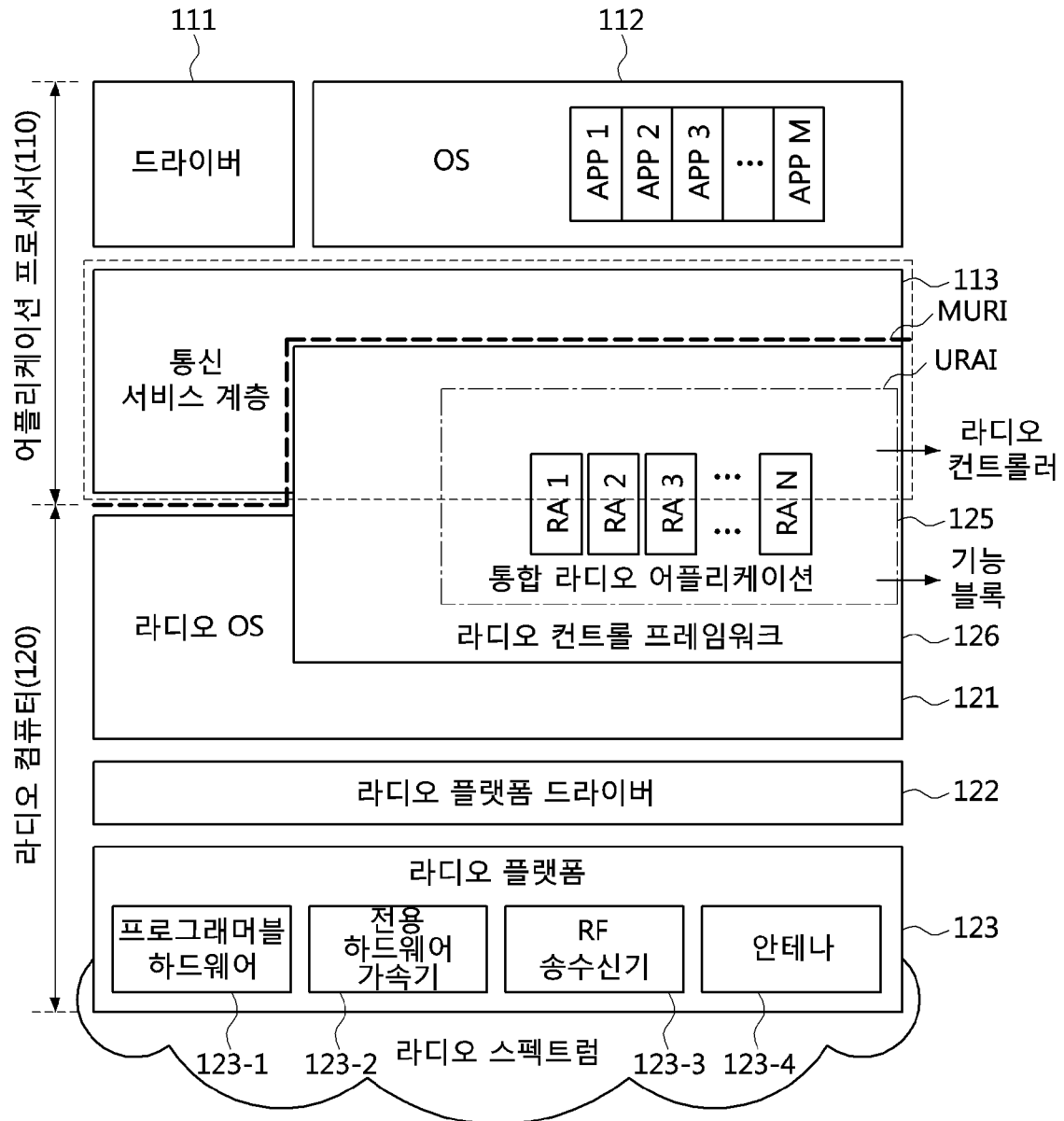
[청구항 16] 청구항 15에 있어서,  
 상기 라디오 컨트롤 프레임워크는,  
 상기 라디오 어플리케이션의 인스턴스/언인스턴스 및 인스턴스(Instance)  
 생성/삭제를 수행하고, 라디오 파라미터들을 관리하는 설정 매니저(CM,  
 Configuration Manager);  
 상기 라디오 어플리케이션의 활성화/비활성화 및 라디오 어플리케이션들  
 간의 사용자 데이터 흐름 스위치를 관리하는 라디오 연결 매니저(RCM,  
 Radio Connection Manager);  
 사용자 데이터 패킷의 송/수신 및 흐름을 제어하는 플로우 컨트롤러(FC,  
 Flow Controller);

라디오 어플리케이션들로부터의 스펙트럼 자원들에 대한 요청을 스케줄링하는 멀티 라디오 컨트롤러(MRC, Multi-Radio Controller); 및 라디오 어플리케이션들간에 라디오 자원의 공유시키는 리소스 매니저(RM, Resource Manager) 중 적어도 하나를 포함하는, 라디오 어플리케이션을 실행하는 단말 장치.

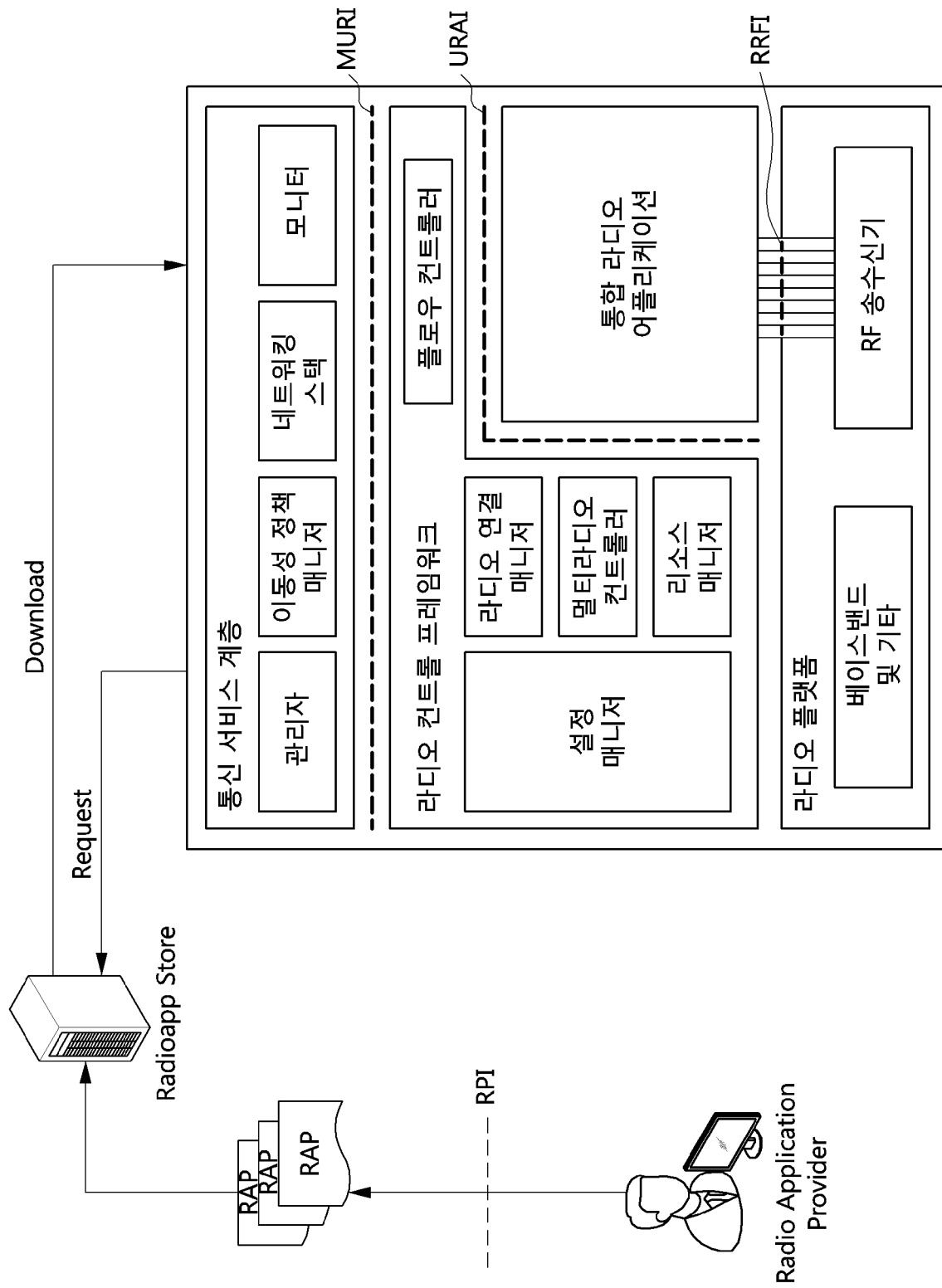
[도 1]



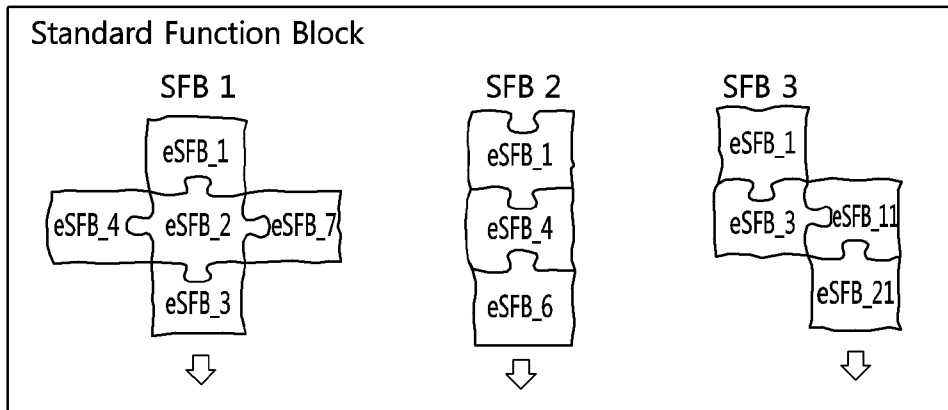
[도2]



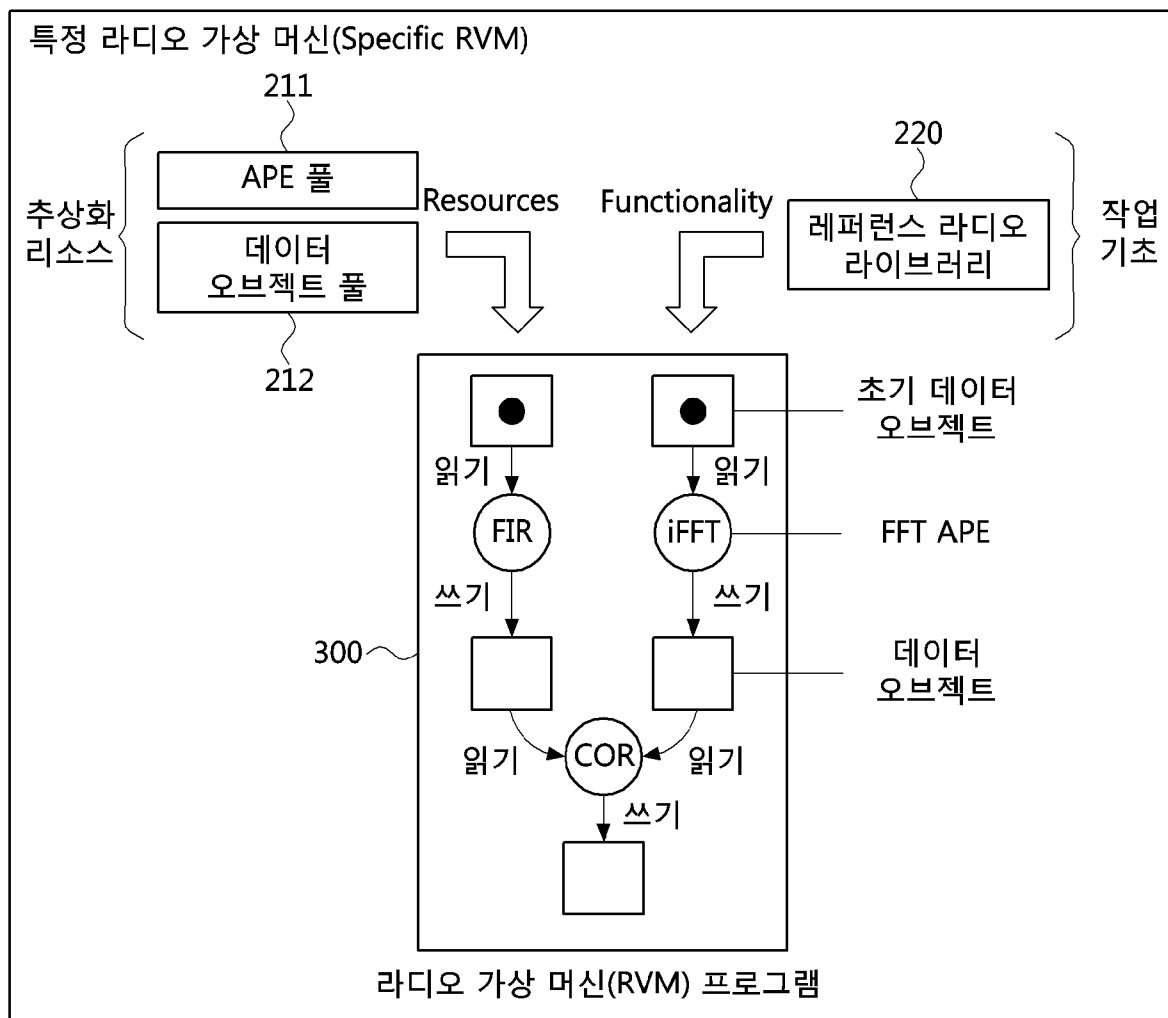
[도3]



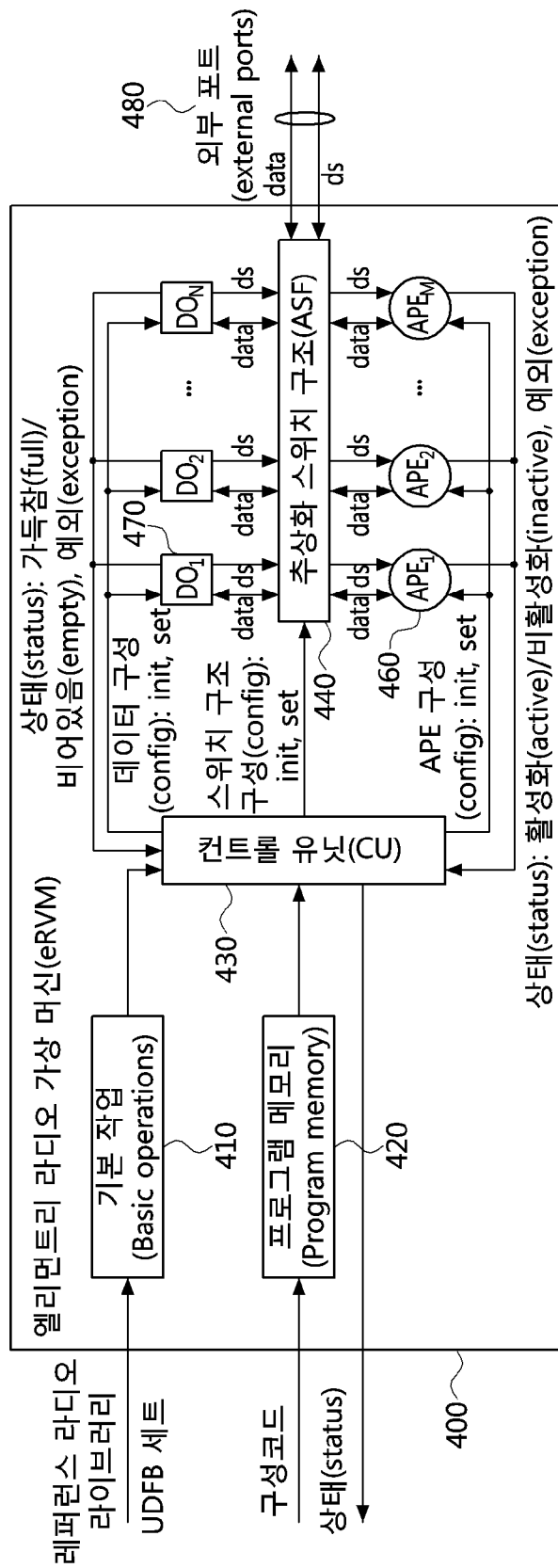
[도4]



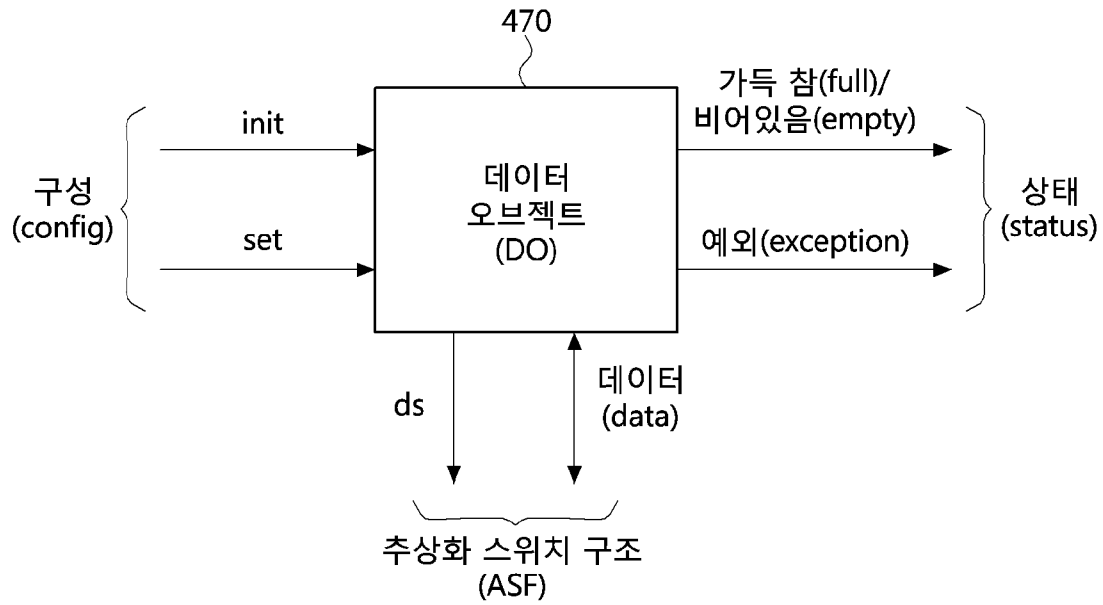
[도5]



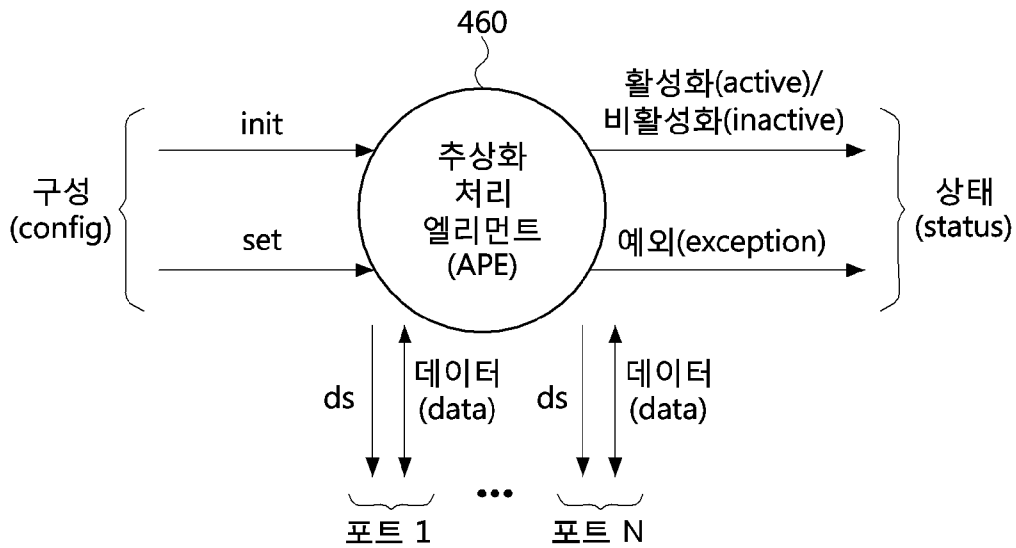
[도6]



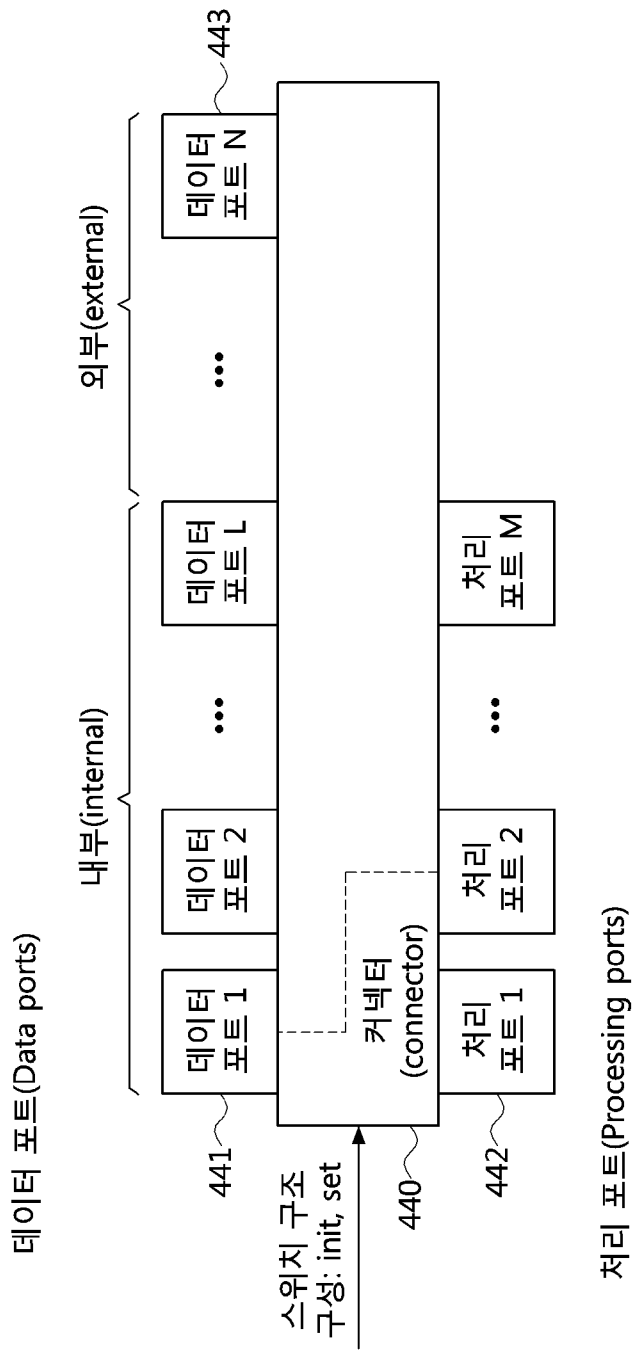
[도7]



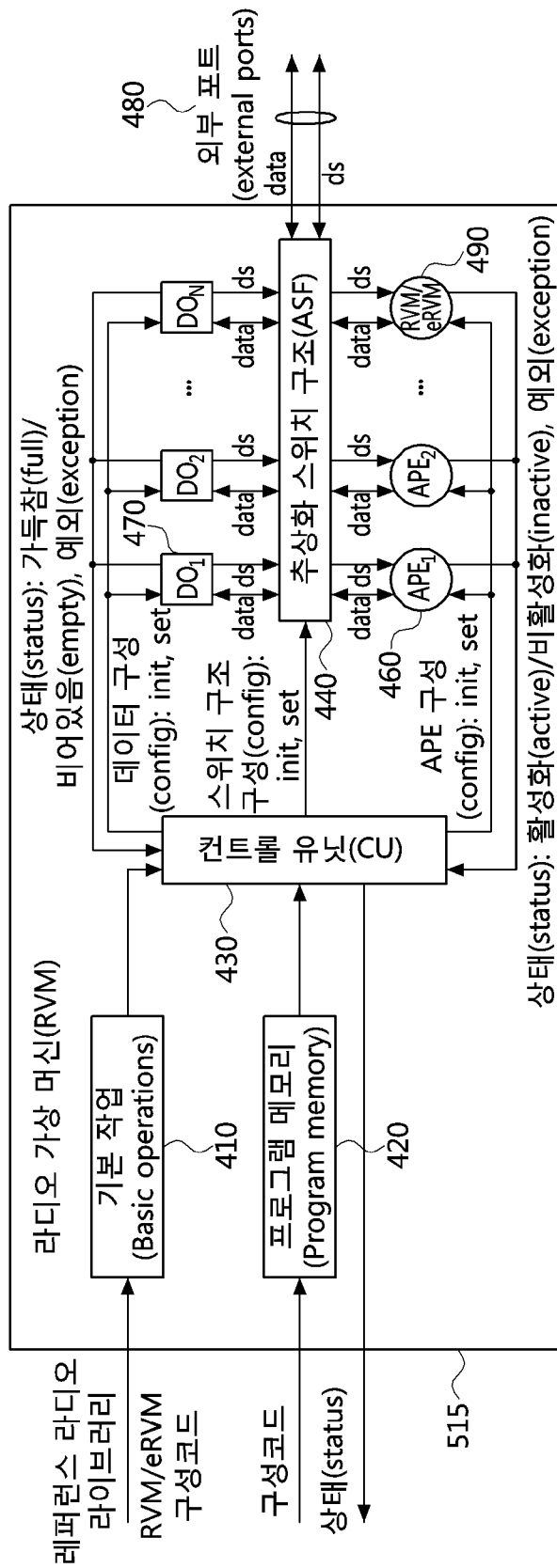
[도8]



[도9]

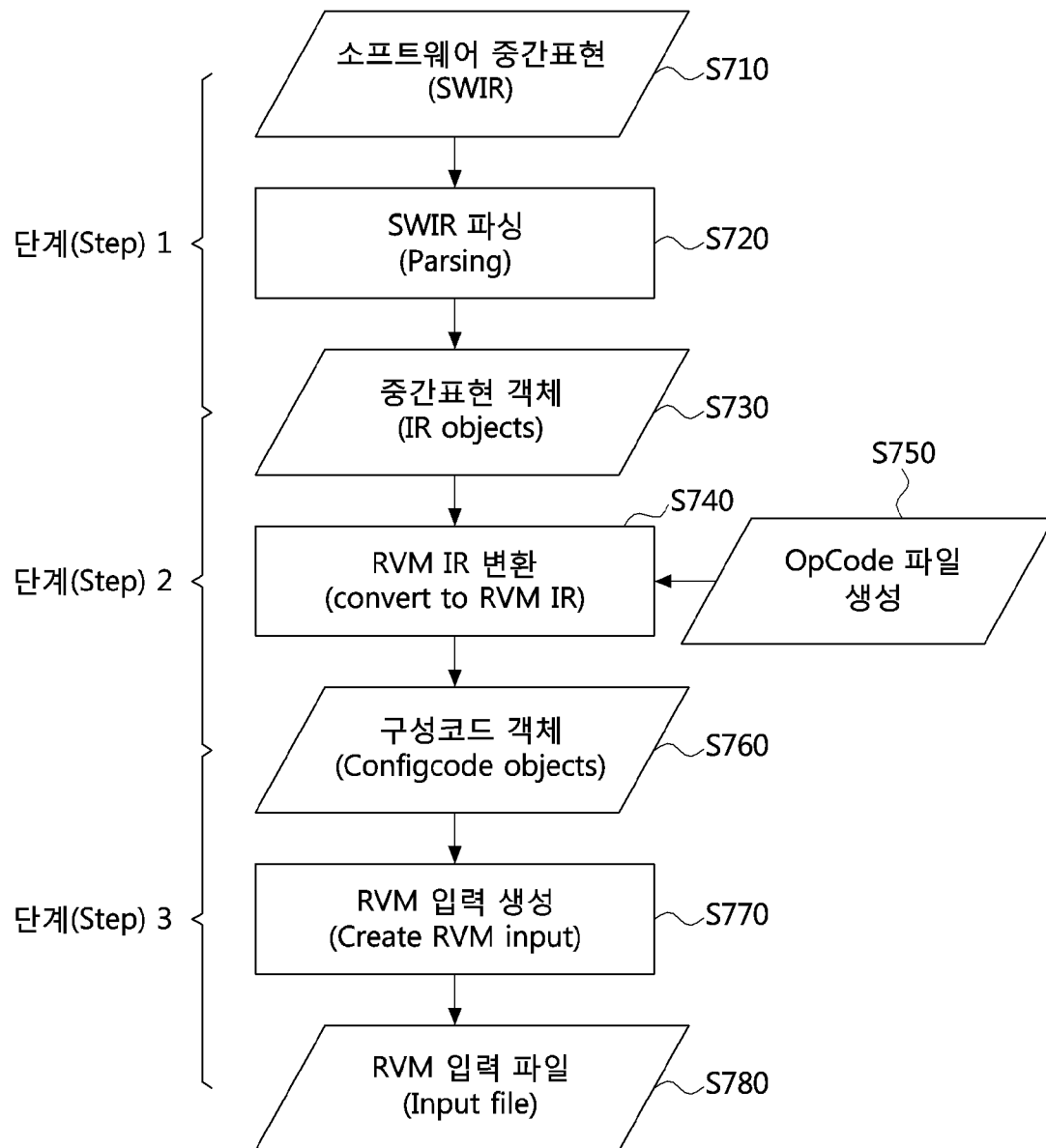


[도10]

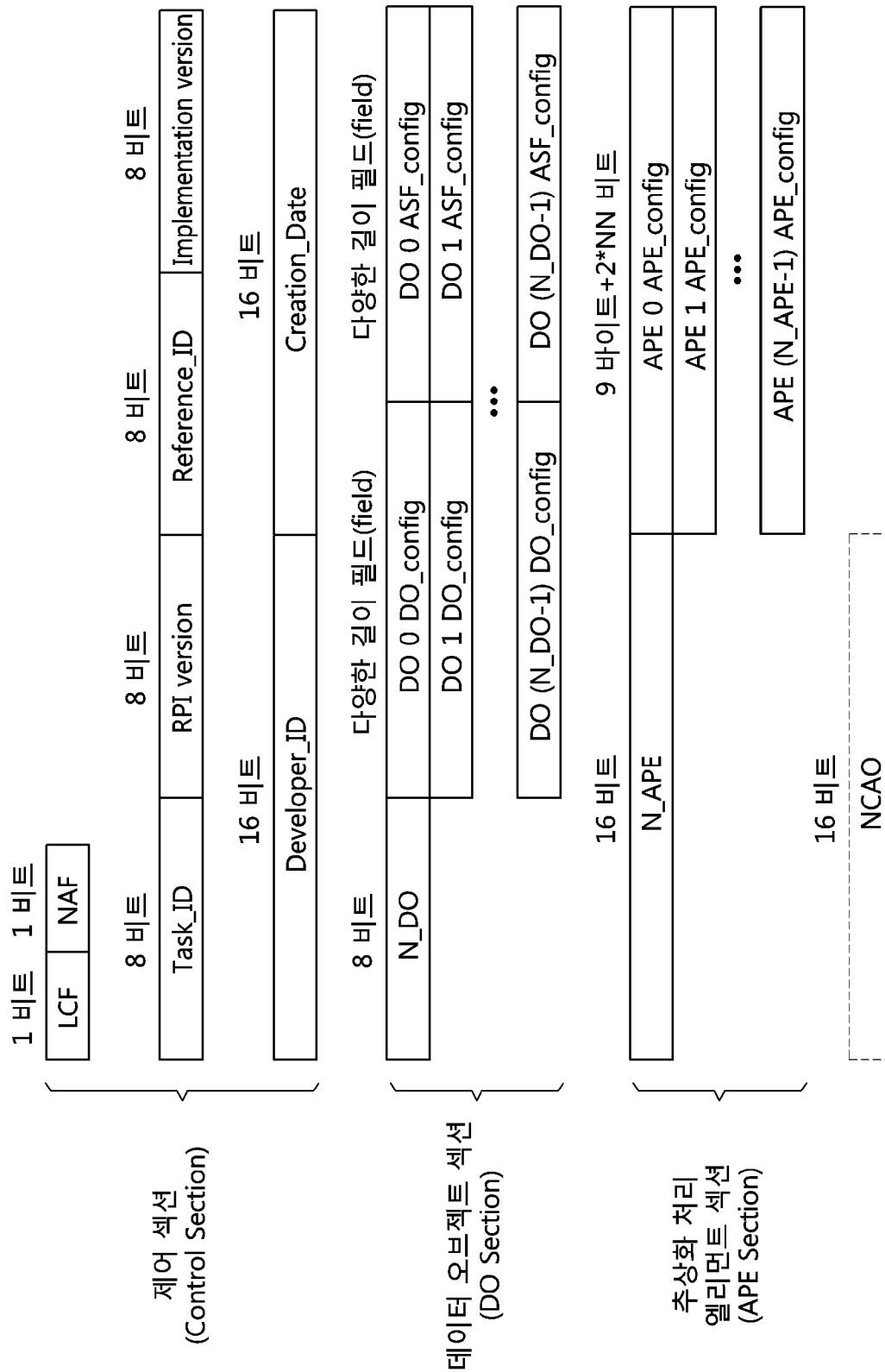




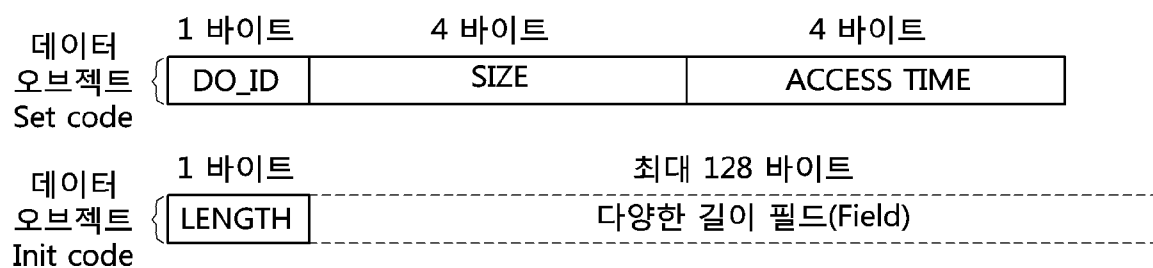
[도12]



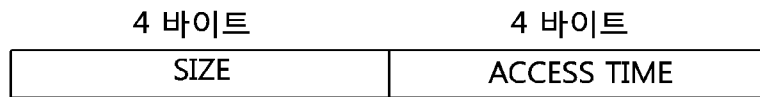
[도13]



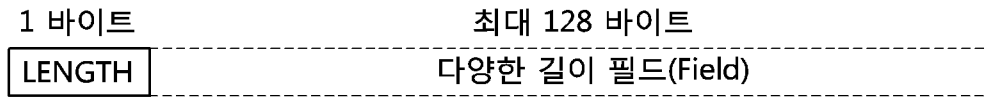
[도14]



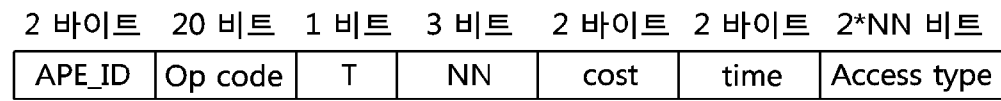
[도15]



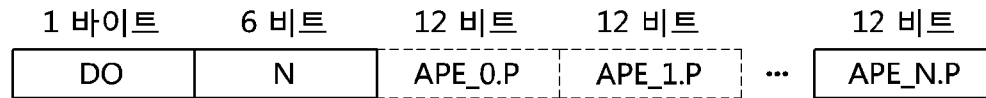
[도16]



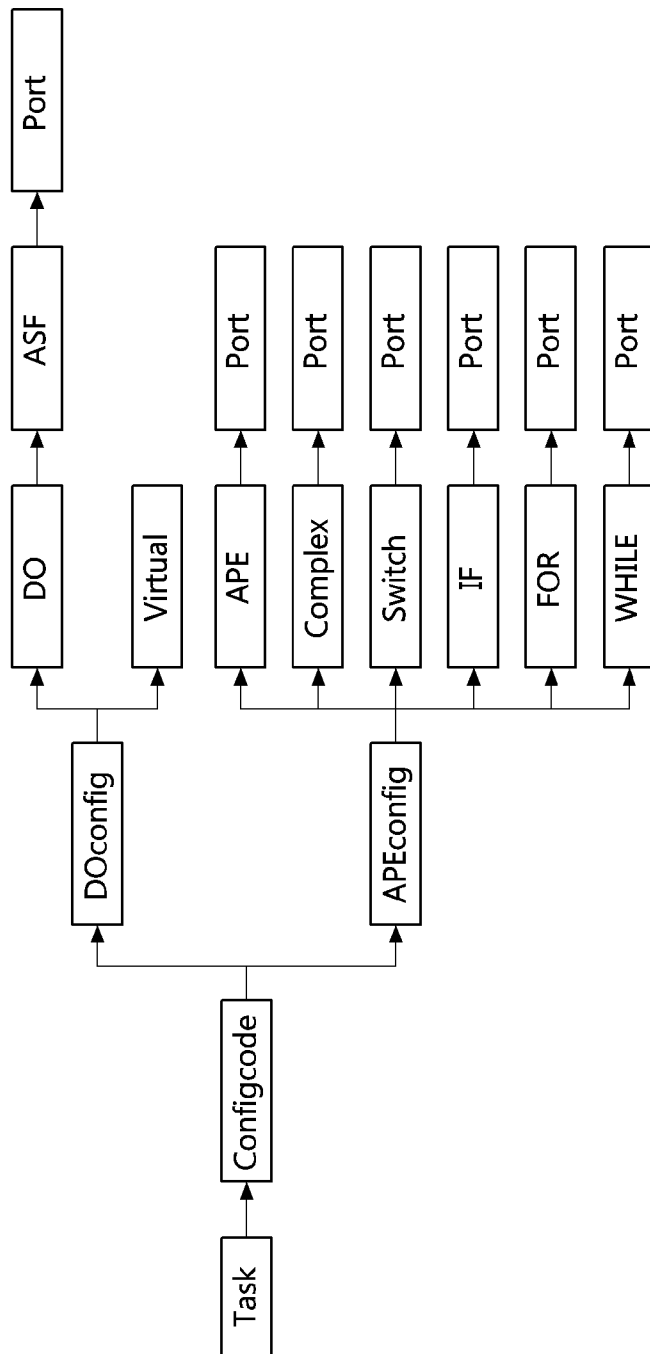
[도17]



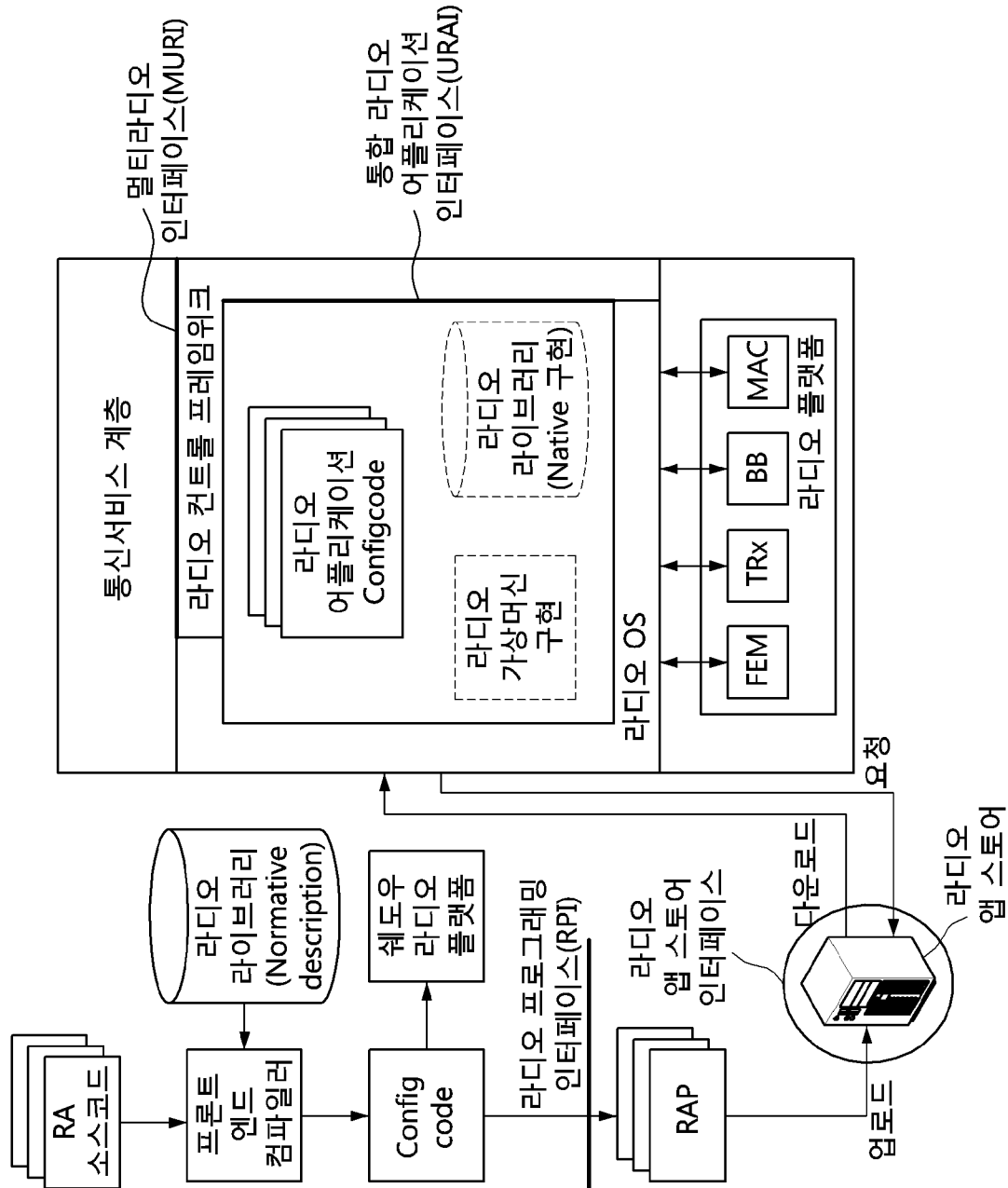
[도18]



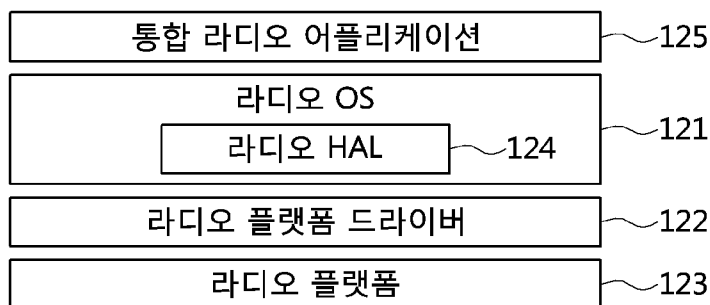
[도19]



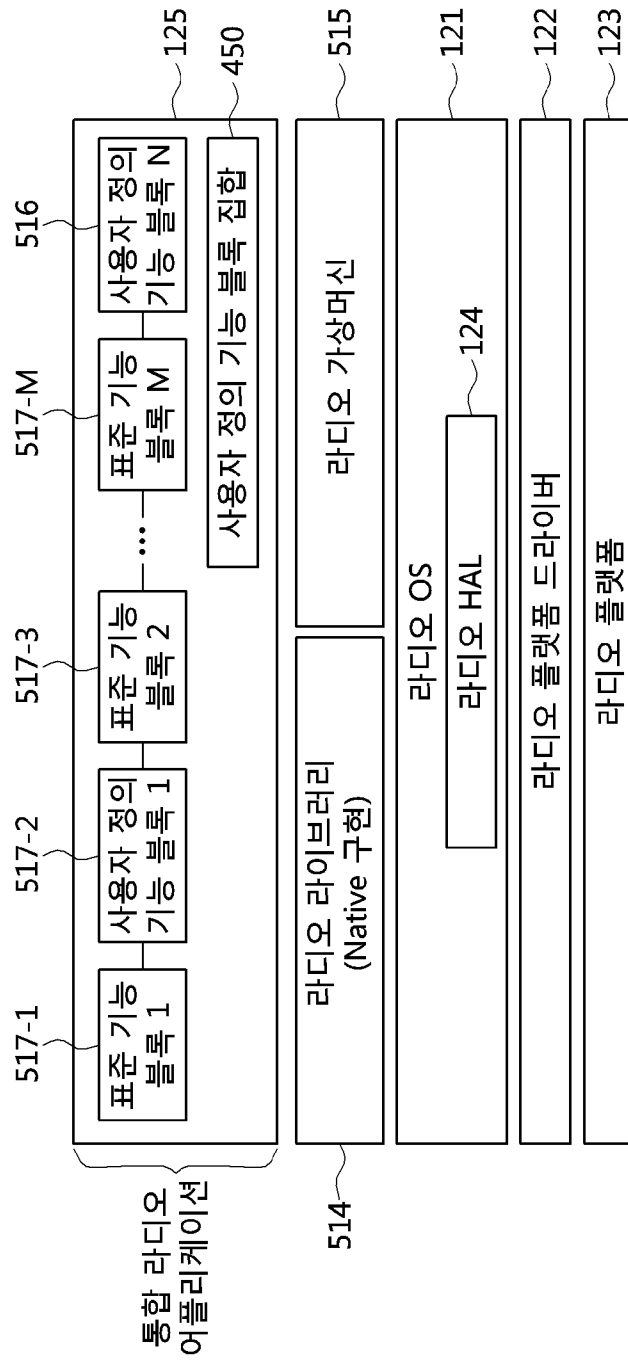
[도20]



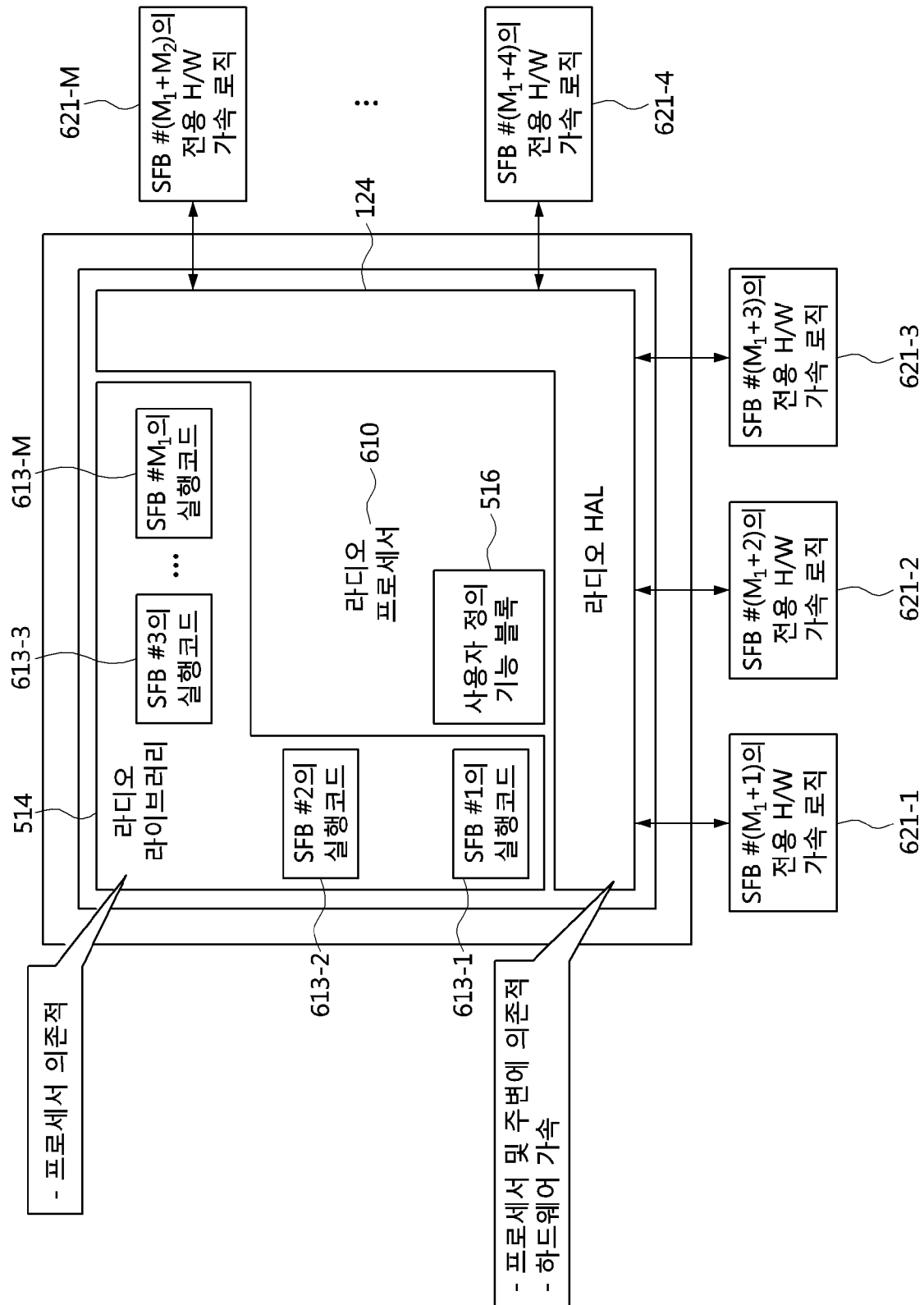
[도21]



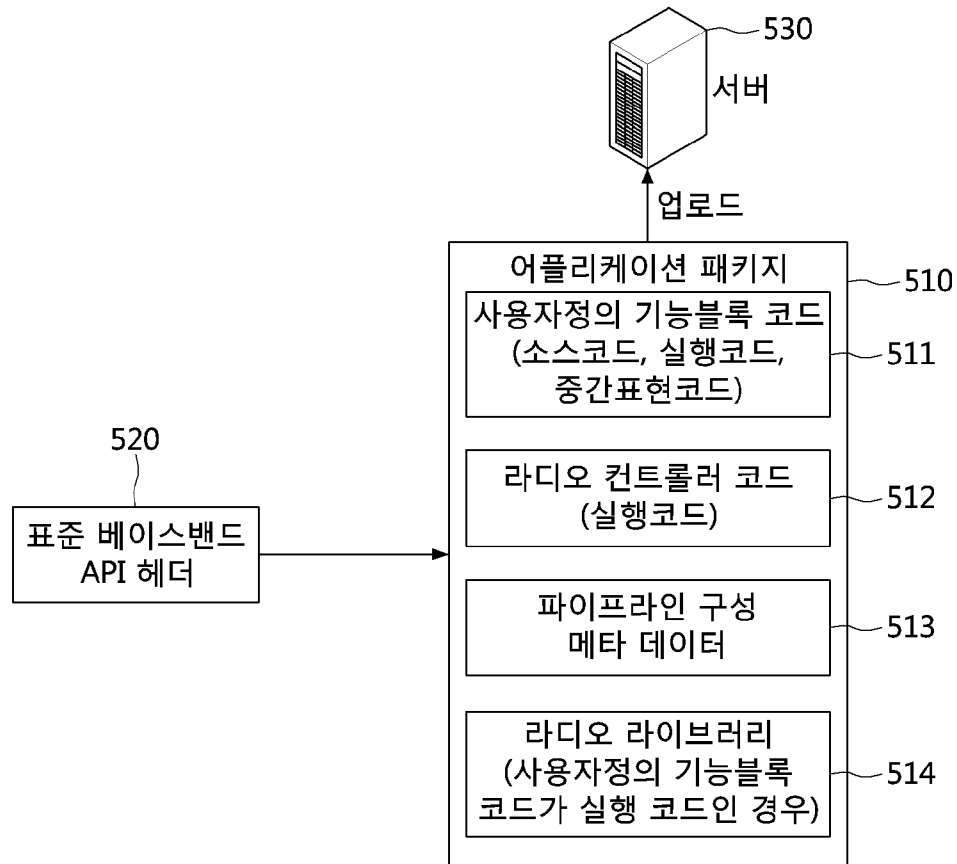
[도22]



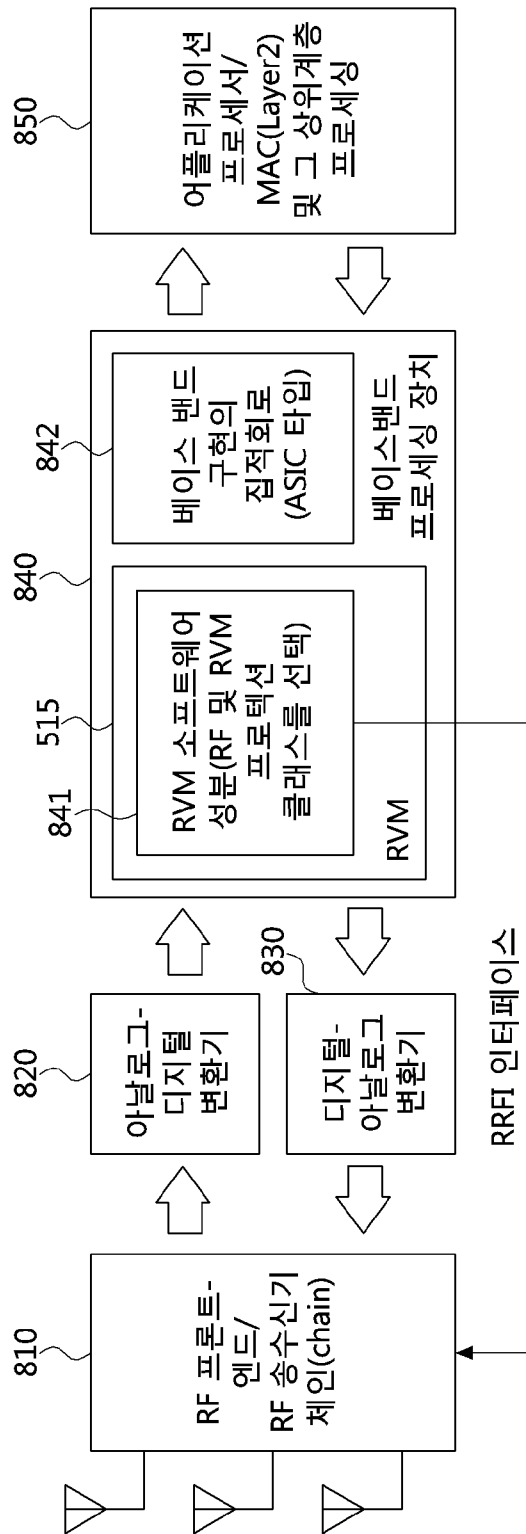
[도23]



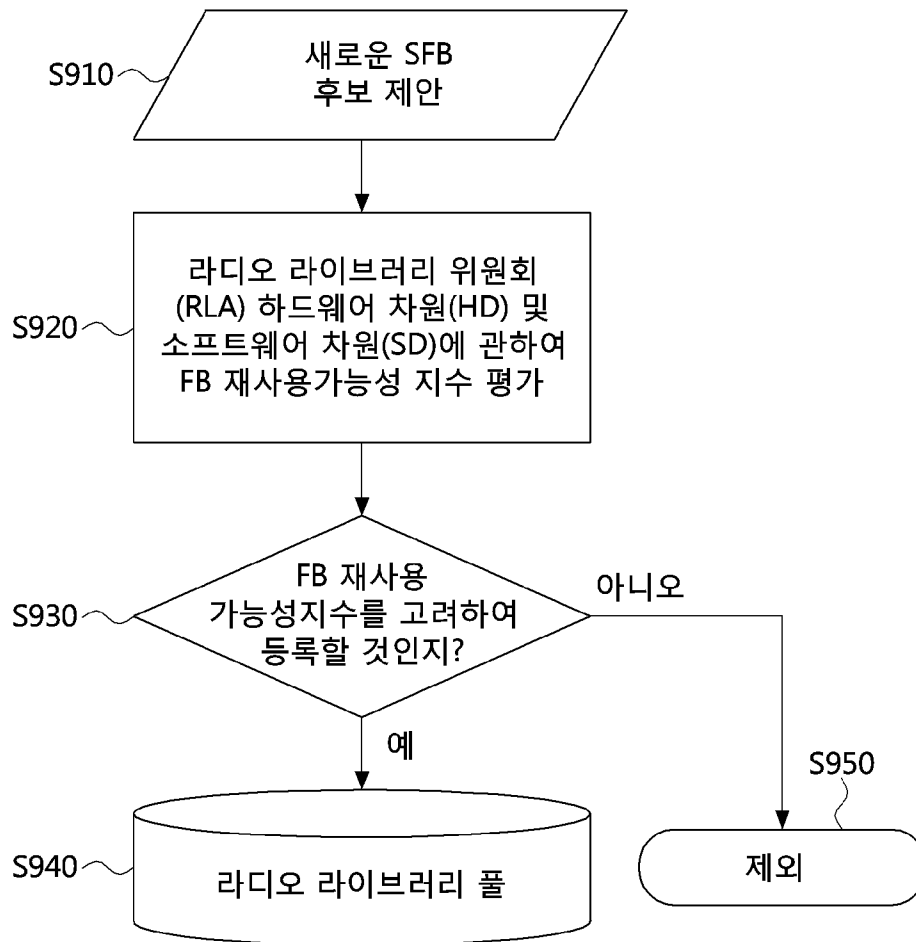
[도24]



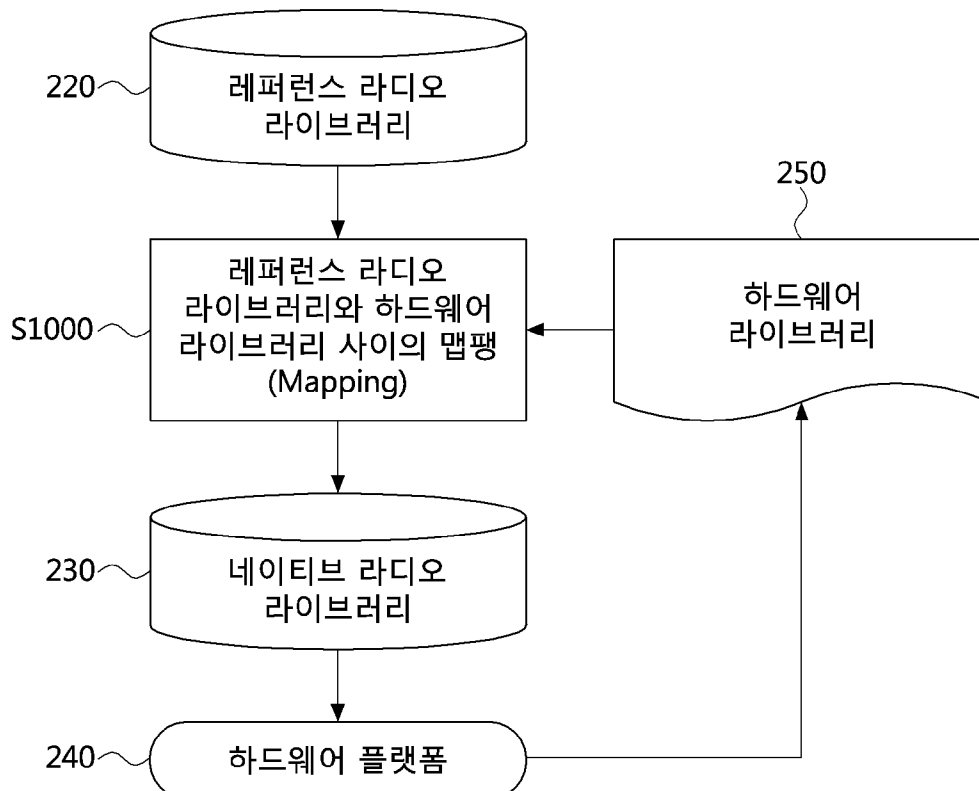
[도25]



[도26]



[도27]



## INTERNATIONAL SEARCH REPORT

International application No.

PCT/KR2017/001504

## A. CLASSIFICATION OF SUBJECT MATTER

*H04L 29/08(2006.01)i, H04W 4/00(2009.01)i, H04B 1/00(2006.01)i, G06F 9/455(2006.01)i*

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L 29/08; G06F 9/44; G06F 9/455; H04W 4/00; H04B 1/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean Utility models and applications for Utility models: IPC as above

Japanese Utility models and applications for Utility models: IPC as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS (KIPO internal) &amp; Keywords: radio virtual machine(RVM, radio virtual machine), abstract, switch, radio application

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages                                                                                                                                                                                                      | Relevant to claim No. |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| X         | IVANOV, Vladimir et al., "Radio Virtual Machine", ETSI Workshop on Future Radio Technologies-Air Interfaces, Sophia Antipolis France, 27-28 January 2016, pages 1-10<br>See pages 1, 4-6 and figures 8-10, 12.                                                                          | 1-11                  |
| Y         |                                                                                                                                                                                                                                                                                         | 12-16                 |
| Y         | KR 10-2013-0116039 A (JUCF-HYU (INDUSTRY-UNIVERSITY COOPERATION FOUNDATION HANYANG UNIVERSITY)) 22 October 2013<br>See paragraphs [0008], [0091], claims 3, 5-6 and figure 4.                                                                                                           | 12-16                 |
| A         | AHN, Chi Young et al., "Development of Standards for Software Modem-based Multi-mode Terminal for ETSI Standards", Korea Institute of Electromagnetic Engineering and Science, Electromagnetic Technology, vol. 24, no. 5, September 2013, pages 65-72<br>See pages 68-69 and figure 4. | 1-16                  |
| A         | US 2015-0178112 A1 (INTEL CORPORATION) 25 June 2015<br>See paragraphs [0025]-[0037] and figures 1-4.                                                                                                                                                                                    | 1-16                  |
| A         | ABDALLAH, Riadh Ben et al., "Virtual Machine for Software Defined Radio: Evaluating the Software VM Approach", 2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010), 16 September 2010, pages 1970-1977<br>See pages 1973-1976.                    | 1-16                  |



Further documents are listed in the continuation of Box C.



See patent family annex.

\* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&amp;" document member of the same patent family

Date of the actual completion of the international search

12 MAY 2017 (12.05.2017)

Date of mailing of the international search report

25 MAY 2017 (25.05.2017)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office  
Government Complex-Daejeon, 189 Sconsa-ro, Daejeon 302-701,  
Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

Telephone No.

**INTERNATIONAL SEARCH REPORT**  
Information on patent family members

International application No.

**PCT/KR2017/001504**

| Patent document<br>cited in search report | Publication<br>date | Patent family<br>member | Publication<br>date |
|-------------------------------------------|---------------------|-------------------------|---------------------|
| KR 10-2013-0116039 A                      | 22/10/2013          | CN 104247290 A          | 24/12/2014          |
|                                           |                     | CN 104254835 A          | 31/12/2014          |
|                                           |                     | EP 2838014 A1           | 18/02/2015          |
|                                           |                     | EP 2838209 A1           | 18/02/2015          |
|                                           |                     | JP 2015-523750 A        | 13/08/2015          |
|                                           |                     | JP 2015-525009 A        | 27/08/2015          |
|                                           |                     | JP 5977438 B2           | 24/08/2016          |
|                                           |                     | KR 10-2013-0116037 A    | 22/10/2013          |
|                                           |                     | US 2015-0052515 A1      | 19/02/2015          |
|                                           |                     | US 9350848 B2           | 24/05/2016          |
|                                           |                     | WO 2013-154380 A1       | 17/10/2013          |
|                                           |                     | WO 2013-154398 A1       | 17/10/2013          |
|                                           |                     |                         |                     |
| US 2015-0178112 A1                        | 25/06/2015          | CN 104731637 A          | 24/06/2015          |
|                                           |                     | EP 2887217 A1           | 24/06/2015          |
|                                           |                     | US 9354910 B2           | 31/05/2016          |

**A. 발명이 속하는 기술분류(국제특허분류(IPC))**

H04L 29/08(2006.01)i, H04W 4/00(2009.01)i, H04B 1/00(2006.01)i, G06F 9/455(2006.01)i

**B. 조사된 분야**

조사된 최소문헌(국제특허분류를 기재)

H04L 29/08; G06F 9/44; G06F 9/455; H04W 4/00; H04B 1/00

조사된 기술분야에 속하는 최소문헌 이외의 문헌

한국등록실용신안공보 및 한국공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

일본등록실용신안공보 및 일본공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

국제조사에 이용된 전산 데이터베이스(데이터베이스의 명칭 및 검색어(해당하는 경우))

eKOMPASS(특허청 내부 검색시스템) & 키워드: 라디오 가상 머신(RVM, radio virtual machine), 추상화, 스위치, 라디오 어플리케이션

**C. 관련 문헌**

| 카테고리* | 인용문헌명 및 관련 구절(해당하는 경우)의 기재                                                                                                                                                                                                                         | 관련 청구항 |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| X     | VLADIMIR IVANOV 등, 'Radio Virtual Machine,' ETSI Workshop on Future Radio Technologies - Air Interfaces, Sophia Antipolis France, 2016.01.27-28, 페이지 1-10<br>페이지 1, 4-6 및 도면 8-10, 12 참조.                                                          | 1-11   |
| Y     |                                                                                                                                                                                                                                                    | 12-16  |
| Y     | KR 10-2013-0116039 A (한양대학교 산학협력단) 2013.10.22<br>단락 [0008], [0091], 청구항 3, 5-6 및 도면 4 참조.                                                                                                                                                          | 12-16  |
| A     | 안치영 등, 'ETSI 표준을 위한 소프트웨어 모뎀 기반 멀티모드 단말기의 표준 개발,' 한국전자과학회, 전자과학기술, 제24권, 제5호, 2013.09, 페이지 65-72<br>페이지 68-69 및 도면 4 참조.                                                                                                                           | 1-16   |
| A     | US 2015-0178112 A1 (INTEL CORPORATION) 2015.06.25<br>단락 [0025]-[0037] 및 도면 1-4 참조.                                                                                                                                                                 | 1-16   |
| A     | RIADH BEN ABDALLAH 등, 'Virtual Machine for Software Defined Radio: Evaluating the Software VM Approach,' 2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010), 2010.09.16, 페이지 1970-1977<br>페이지 1973-1976 참조. | 1-16   |

☐ 추가 문헌이 C(계속)에 기재되어 있습니다.

☒ 대응특허에 관한 별지를 참조하십시오.

\* 인용된 문헌의 특별 카테고리:

"A" 특별히 관련이 없는 것으로 보이는 일반적인 기술수준을 정의한 문헌

"T" 국제출원일 또는 우선일 후에 공개된 문헌으로, 출원과 상충하지 않으며 발명의 기초가 되는 원리나 이론을 이해하기 위해 인용된 문헌

"E" 국제출원일보다 빠른 출원일 또는 우선일을 가지나 국제출원일 이후에 공개된 선출원 또는 특허 문헌

"X" 특별한 관련이 있는 문헌. 해당 문헌 하나만으로 청구된 발명의 신규성 또는 진보성이 없는 것으로 본다.

"L" 우선권 주장에 의문을 제기하는 문헌 또는 다른 인용문헌의 공개일 또는 다른 특별한 이유(이유를 명시)를 밝히기 위하여 인용된 문헌

"Y" 특별한 관련이 있는 문헌. 해당 문헌이 하나 이상의 다른 문헌과 조합하는 경우로 그 조합이 당업자에게 자명한 경우 청구된 발명은 진보성이 없는 것으로 본다.

"O" 구두 개시, 사용, 전시 또는 기타 수단을 언급하고 있는 문헌

"&" 동일한 대응특허문헌에 속하는 문헌

"P" 우선일 이후에 공개되었으나 국제출원일 이전에 공개된 문헌

국제조사의 실제 완료일

2017년 05월 12일 (12.05.2017)

국제조사보고서 발송일

2017년 05월 25일 (25.05.2017)

ISA/KR의 명칭 및 우편주소



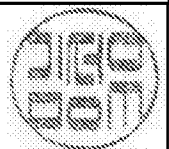
대한민국 특허청  
(35208) 대전광역시 서구 청사로 189,  
4동 (둔산동, 정부대전청사)

팩스 번호 +82-42-481-8578

심사관

김성우

전화번호 +82-42-481-3348



국제조사보고서에서  
인용된 특허문헌

공개일

대응특허문헌

공개일

|                      |            |                      |            |
|----------------------|------------|----------------------|------------|
| KR 10-2013-0116039 A | 2013/10/22 | CN 104247290 A       | 2014/12/24 |
|                      |            | CN 104254835 A       | 2014/12/31 |
|                      |            | EP 2838014 A1        | 2015/02/18 |
|                      |            | EP 2838209 A1        | 2015/02/18 |
|                      |            | JP 2015-523750 A     | 2015/08/13 |
|                      |            | JP 2015-525009 A     | 2015/08/27 |
|                      |            | JP 5977438 B2        | 2016/08/24 |
|                      |            | KR 10-2013-0116037 A | 2013/10/22 |
|                      |            | US 2015-0052515 A1   | 2015/02/19 |
|                      |            | US 9350848 B2        | 2016/05/24 |
|                      |            | WO 2013-154380 A1    | 2013/10/17 |
|                      |            | WO 2013-154398 A1    | 2013/10/17 |
| US 2015-0178112 A1   | 2015/06/25 | CN 104731637 A       | 2015/06/24 |
|                      |            | EP 2887217 A1        | 2015/06/24 |
|                      |            | US 9354910 B2        | 2016/05/31 |